THE UNIVERSITY OF MICHIGAN

Memorandum 34

GRAPHICS RAMP USER'S MANUAL

W.S. Gerstenberger

R.W. Taylor

COMCOMP: Research in Conversational Use of Computers
ORA Project 07449
F.H. Westervelt, Director

supported by:

DEPARTMENT OF DEFENSE
ADVANCED RESEARCH PROJECTS AGENCY
WASHINGTON, D.C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

July 1970

ABSTRACT

The Graphics RAMP system provides basic display and communication support for a DEC 338 display system communicating over serial, synchronous, voice-grade telephone lines to a large computer. This memorandum details the various system commands and associated modules which are particular to the 338 graphics configuration.

TABLE OF CONTENTS

1. INTRODUCTION

This manual describes the use of the Graphics RAMP System for the DEC 338 computer. This system provides a Command Language Interpreter (CLI); I/O support routines for: two teletypes, a 201A Dataphone, a PC-01 paper tape reader/punch, a GRAFACON tablet, a DF-32 disk, and a Calcomp plotter; and a graphics support package which when used in conjunction with the command language permits the construction of line or character display files. A dynamic display file loader permits 338 display files (constructed in an external machine) to be loaded from input devices. Generally, such files are constructed by the DF (Display File) routines[1] which run in MTS, the Michigan Terminal System[2], and are then transmitted on-line from the 360/67 to the 338 via a 201A Dataphone connection. Display files generated locally by commands may be intermixed (i.e., appear simultaneously on the screen) with display files generated by MTS. In addition, a third method of producing display output allows the screen to be used for alphanumeric output with each character line "scrolled" down the screen at a rate controlled by push buttons. In conjunction with the command language, the light pen may be used for drawing and to point at previously defined display files.

Generally, the intent of this manual is to provide sufficient information so that an intelligent user may use

the system either locally or in conjunction with MTS. As such, only the display routines are described in detail since it is this portion of the system which is really different from other versions of the PDP-8 RAMP System[3,4,5]. Thus, the manner in which display files are constructed is described in considerable detail. All CLI commands are described.

2. LOADING

The usual case is for the Graphics RAMP System to be loaded from the DF-32 disk using the system Disk Loader. Of course, the system may also be loaded from paper tape using the DEC Binary Loader. Assuming that the Disk Loader has been loaded from tape into core bank 2 using the Binary Loader, the following procedure will load RAMP from the disk.

(1) Set IF=2 and SR=7400

(2) Press LOAD ADD

(3) Press START

The Disk Loader will load and start program zero (RAMP) since the low-order 5 bits of the SR are zero. To inhibit the starting of RAMP (or any other disk program), set bit 6 after pressing LOAD ADD, but before pressing START. This causes the Disk Loader to halt after loading (like the Binary Loader does). The starting address for RAMP is 0200 in core bank zero.

3. TELETYPE OPERATION

Teletypes operating with the Graphics RAMP System must be run in full-duplex (FDX) mode and must be either model 33 or 35 with or without paper tape equipment. They may use odd, even, or no parity since parity is ignored by the RAMP keyboard routines.

Normally, teletypes are attached via the Bell System switched network, although an "on-line" teletype is available in the 338 machine room. The telephone numbers of the two teletype ports on the 338 and their associated logical devices numbers are:

(313) 764-7146 LDN 1

(313) 764-7147 LDN 2

The end-of-line character for teletypes is RETURN. The character-delete character (i.e., backspace character) is left-arrow; newer teletypes sometimes have underscore instead of left-arrow) which is typed as SHIFT-O. The backspace may be entered repeatedly to backspace over more than one character. Left-arrows (or underscores) will be echoed unless the backspacing has reached the beginning of the input line, i.e., there remains nothing to backspace over, in which case the backspaces are ignored. An entire input line may be completely cancelled by sending a RUBOUT character at any time before the RETURN is entered. A pound sign "#" will be echoed (unless there were no characters to delete)

followed by a carriage-return/line-feed. The end-of-file character is CTRL-Y which causes a backslash to be echoed followed by normal end-of-line processing. The backslash provides a visual indication that an end-of-file was entered to end the input line. The attention character is WRU. This character when entered will empty the input buffer and transmit an attention indication to MTS. An exclamation point "!" is echoed followed by carriage-return/line-feed. The ESC character (which may also be typed as CTRL-SHIFT-K on teletypes which lack an ESC key) causes RAMP to invert the echoing on/off switch. This character may be entered before and after passwords so that the password is not echoed.

Because of the full-duplex operation, it is permissible to type while output is in progress with the result that output continues normally until the end of the current output line, at which time the input line which has been (or is being) typed is echoed. Output will not begin again until the input line has been ended.

4. I/O DISPATCHER

The I/O Dispatcher does the initial processing of all input records from all logical devices. Provision has been made in the graphics RAMP system for each logical device to be in any one of eight modes (0-7), although only two of the eight are functional at this time. When an input record (e.g., a typed line from a teletype keyboard) is received, if its first character is not one of several special characters, then the complete input record is passed to a subprocessor according to whichever mode the input device is in. Thus, the mode of a device determines which of eight possible subprocessors will process the input record in the event that the first character is not a special character.

Currently, there are four special characters (recognized only as the first character of a record) which cause special action by the Dispatcher regardless of what mode the source device is in. The ASCII character SOH (Start of Header), which is typed CTRL-A from a teletype keyboard, causes the input record (with the initial SOH stripped off) to be sent to the Command Language Interpreter (CLI) subprocessor. This produces the same effect as having the source device in mode 0 (command mode). SOH and the other special characters STX, SOCR, and SOBF do not change the mode of the input device. The ASCII character STX (Start of Text), which is typed CTRL-B, is similar

to SOH except that it forces the input record (less the STX) to be processed by the Copy Task as if the input device were in mode 1 (copy mode) regardless of what mode it may actually be in. The SOCR (Start of Calcomp Record) character is used to direct records to the Calcomp plotter subprocessor independent of the mode of the source device. Finally, the SOBF (Start of Binary File) character is used to designate a record which contains binary display file data. Such records (less the SOBF) are forced to the Binary Display File Loader regardless of the mode of the input device. There is no mode corresponding to either the Calcomp subprocessor or the Binary Display File Loader; thus, all records destined for them must begin with SOCR and SOBF respectively, and they are not counted among the eight possible subprocessors although they are logically parallel with them. Normally, records beginning with SOCR and SOBF are received only from the 201 Data Phone (i.e., they originate in the 360), although there is no reason why they can't come from other devices.

In summary, all records which begin with SOH are sent to the CLI. Records which begin with STX are forced to the Copy Task. Those which begin with SOCR are routed to the Calcomp subprocessor; and those which start with SOBF are sent to the Binary Display File Loader. All records which begin with none of these special characters are routed according to the mode of the input device which generated the record (mode 0

for the CLI, modes 1-7 for the Copy Task). Provision has been made for temporarily overriding the recognition of these special characters. See the description of the PFX command.

The I/O Dispatcher also sets up the default sink device for each record which it dispatches. Each input device has a sink device associated with it (known as the "coupled sink") in addition to the mode information described above. The coupled sink and the mode may be changed by the "CONTROL" command. If the subprocessor produces output, it is generally sent to this coupled sink. Certain CLI commands, however, have as arguments the device number of the sink to which the output for that command is to be directed. In this case, the default "coupled sink" is not used, although it remains unchanged.

5. GRAPHICS SUPPORT

The following section details the display file structure and the system subroutines which operate on this structure. It is included for completeness, the education of other system programmers, and for the sophisticated user who wishes to write his own RAMP display commands and tasks. 338 users who rely primarily on the DF routines need not be concerned with the level of detail found here.

Most display service subroutines require that the dsiplay be available solely to them, since certain key information is often passed in system registers. Thus, the display is usually reserved by invoking the sub-task.

```
JMS*  TASK
 DC   DSPSEL
```

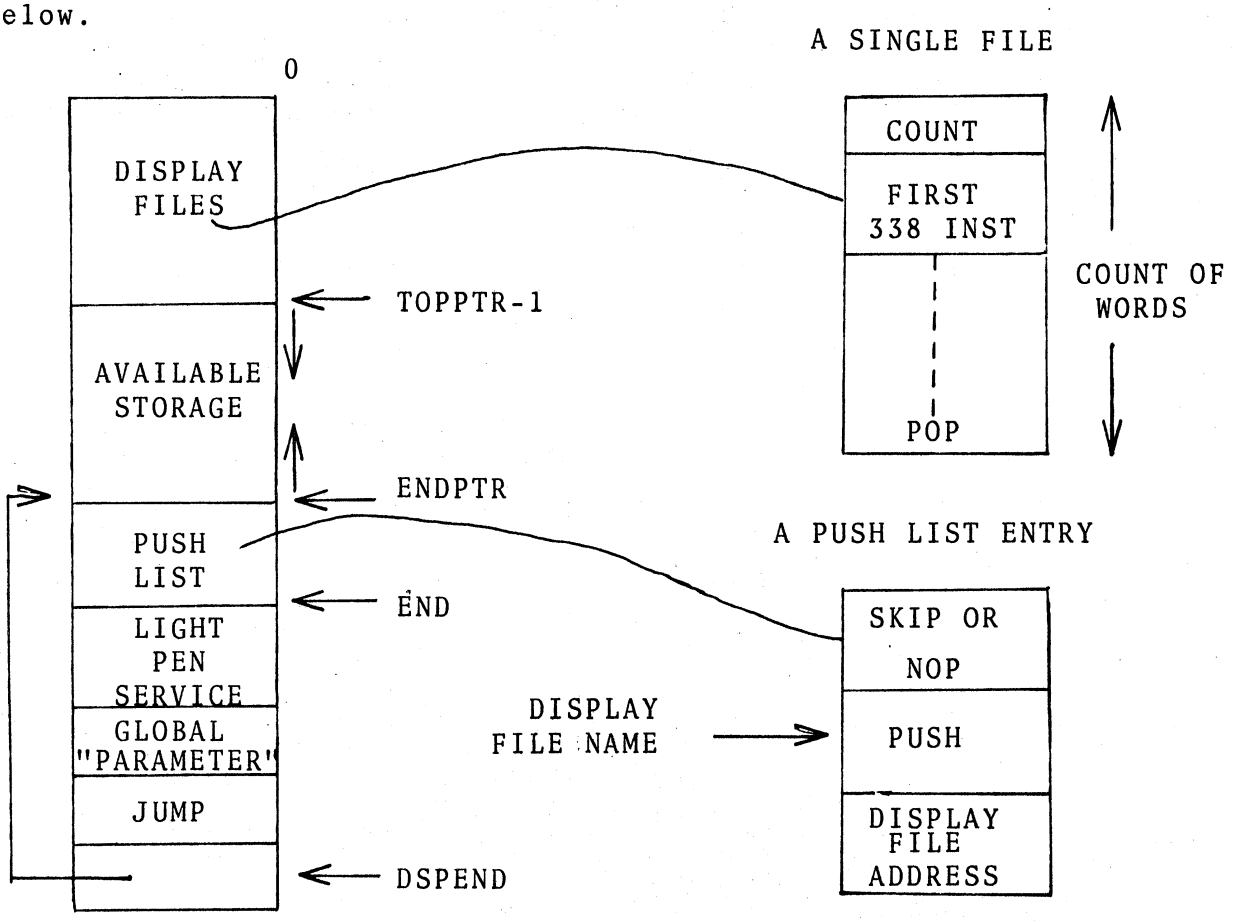
The display will be reserved upon return.

If the 338 has been reserved, then it must be explicitly released when all manipulation is finished via a call.

```
JMS*  RESTDI
```

5.1 338 Display File Structure

Display files for the 338 are stored in Corebank B of the PDP-8 core. The structure of these files is shown below.



DSPEND currently has the value 7777, but is an assembly parameter. TOPPTR is indexed by using the system routine DISPUT. ENDPTR is decremented in steps of 3 by the ENTRY subroutine. The subroutine PTRCHK disallows overlap between TOPPTR and ENDPTR. Thus the displayed picture is a collection of sub-display files, each of which has a name, which in turn is an address in the push list.

The display sequence is started at the global parameter setting word. The display then jumps to the head of the push list. Each push is then executed, causing the subfile to which it points to be executed by the display. These subfiles normally contain display instructions of the beam movement type, but they may contain further push instructions to other display files. Each subfile ends in control state with a POP, thus causing a return to the push list.

The two cases when a subfile does not contain beam-movement instructions are of some interest. The easier case is when the subfile is a character string. The subfile then contains a series of push instructions to predefined locations in Corebank C, where special character display files are stored. The other case is that of a "subroutine" display file cell. Here the push is to the second word of a special 3-word block in the push list. This block has the format

SKIP

JUMP

SUBROUTINE ADDRESS

Thus control is transferred to the subroutine file, which will return control to the calling file upon execution of a POP instruction.

5.2 System Subroutines for Servicing 338 Display Files

ERASE: calling sequence

```
***              AC = 0
JMS*   ERASE
***              AC = 0
```

Erases the display file named in the system location NAME. The display must be seized before ERASE is called. The erase algorithm works as follows:

1. Stop the display
2. Make a check that the display file name is legal
3. Erase the 3-word block at the highest level
4. Compress the storage by moving all display files with higher (numerically) addresses
5. Relocate the addresses in the "push list" to conform with the new locations of moved instructions
6. Restart but do not release the display

```
ENTRY:   calling sequence    ***              AC = 0
                             JMS*   ENTRY
                             ***              AC = 0
```

Enters a display file into the displayed loop. The file is assumed to be already present with the address of its count in system location DISBGN. (Such a pointer will be constructed automatically if the display file was constructed using routines DISOPN and DISPUT).

If system location NAME contains a non-zero display value, then its contents will be used as the "display file name", i.e., as the address of the highest level push. If NAME contains a zero value, then a new name will be created using GETNAME and returned to the user in location NAME.

The display should be seized prior to the use of subroutine ENTRY.

ENTRY uses the following algorithm

1. Check for NAME = 0

   If so, then call GETNAME

   If not, continue

2. Place a push instruction in the Corebank 1 location indicated by NAME

3. Put 1+ the contents of DISBGN in the location one past the contents of NAME
4. Update the contents of ENDPTR and DSPEND to eliminate any extraneous display NOP instructions whenever possible.

GETNAME: calling sequence ***
JMS* GETNAME
*** AC = 0, DF = CBB

Scans the push list from higher addresses to lower addresses to find a three-word entry marked as "available", i.e., contents of second word equal to zero.

Returns the address of the second word in the three-word block in system location NAME.

The display should be seized before calling GETNAME.

1. Start at location indicated by END as the highest possible 3-word block
2. Check the second word of each 3-word block until
   a. Find one equal to zero, in which case return with the found address in NAME
   b. Would go below contents of ENDPTR, in which case create a new three-word block and update ENDPTR after checking that no overflow with existing display files will occur.

```
BLANK:   calling sequence   ***              AC = 0
                            JMS*    BLANK
                            ***              AC = 0, DF = CBB
```

Places a display SKIP instruction in the location in Corebank B one previous to the contents of system location NAME.

```
UNBLANK:  calling sequence  ***                AC = 0
                            JMS*    UNBLANK
                            ***                AC = 0, DF = CBB
```

Places a display NOP instruction in the location in Corebank B one previous to the contents of system location NAME.

The display should be seized before executing either of these calls.

DISOPN:

DISPUT: calling sequence

```
***                              AC = ARG
JMS*   DISOPN,DISPUT
***                              AC=0, DF=CBA
```

Place arguments in a display file, checking for overlap with the push list.

DISOPN should be called the first time, and DISPUT for succeeding entries.

The display must be seized before calling either routine.

DISOPN

1. Save TOPPTR in DISBGN
2. Call DISPUT to put a one in the count location in Corebank B indicated by DISBGN (=TOPPTR)
3. Call DISPUT to put argument in the next location (corebank B) and index the "count" (pointed to by DISBGN)

DISPUT

1. Check for possible overflow with the push list. Hang if overflow exists
2. Place argument in location indicated by TOPPTR
3. Index location pointed to by DISBGN
4. Index TOPPTR

6. COMMANDS

The RAMP Command Language Interpreter (CLI) is called by the I/O Dispatcher as described in the previous section. The CLI performs the function of reading successive input characters from the input device (ignoring leading non-alphanumeric characters) until it finds a non-alphanumeric character or the end-of-record indication. The first and last characters of the alphanumeric string so found are then used to identify a command name. If no such command exists, an error comment is printed on the coupled sink device. Otherwise, control is passed to a particular CLI segment which reads arguments (if any) from the remainder of the input record and performs a particular function as described below. After completing this function, it returns to a common section of the CLI which assures that all of the input record has been read (i.e., extraneous characters on the end of the input line are swallowed).

When numbers occur as arguments in commands (as is the usual case), they must be separated by one or more non-numeric characters (e.g., spaces, commas, etc.). Negative arguments are given by immediately preceding the number by a minus sign.

In the command descriptions below only the underlined characters in the command name need be given--other intervening alphanumeric characters are ignored.

HELLO n

This command serves only as a check of the airworthiness of the system. A comment will be printed on logical device n. If n is omitted, the comment is defaulted to the coupled sink.

Example: HO

CONTROL n m s

This command causes the mode of device n to be changed to m and the coupled sink for device n to be changed to s. An asterisk for either m or s will leave its associated value unchanged. An asterisk for n will make the command apply to the source device.

Example: CL 1 1 4

This causes device 1 (the online teletype) to be placed in mode 1 (copy mode) with coupled sink device 4 (the 201 Dataphone).

Current Device Assignments:

0	Dummy file (similar to *DUMMY* in MTS)
1	Online teletype (764-7146)
2	Alternate teletype (764-7147)
3	Utility file #1 (used only for system checkout)
4	201 Dataphone (764-7145)
5	338 Display (output only)
6	Utility file #2
7	PC-01 High-speed paper-tape reader/punch

Current Mode Assignments:

0	Command
1	Copy
2-7	Not used (actually Copy mode)

ECHO n s

This command causes the character string s to be transmitted to device n.

Examples: EO 5 THIS WILL BE SEEN ON THE SCREEN.

EO 4 HELLO MTS

ATTENTION [n1 n2 ...]

This command causes an attention character to be asynchronously transmitted to devices n1, n2, etc. with the additional result that all further transmission to those devices is thrown away until each device is reset (see description of RESET command). If no device number is given, the coupled sink device is used.

Example: AN 4

RESET [n1 n2 ...]

This command causes an attention character to be transmitted to devices n1, n2, etc. and also resets the devices so that each is able to receive output again (if output had been previously stopped by the ATTENTION command). If the device number is omitted, the coupled sink is used.

Example: RT 4

SENSE [n1 n2 ...]

This command causes one line of sense data to be printed for n1, n2, etc. If no argument is given, sense data for the source device is printed. An asterisk may be given for n1 with the result that sense data for the source device is printed. All output (i.e., the printed sense data) is sent to the coupled sink device. The format for the sense data is:

LDN SOUDCB SINDCB SOUBCB SINBCB LA1 LA2

Each of the sense items is a four-digit octal number and the last two items (LA1 and LA2) appear only for device 4 (the 201 Dataphone)--these are the two line adapter status words, a description of which is available in the design report for the 201 interface[6]. The first item (LDN--logical device number) is the device for which this line of sense data applies. The next four words are the source and sink device control block (DCB) words and the source and sink buffer control block (BCB) pointers. Normally, the information provided by the SENSE command is used only for system debugging. However, an explanation of the important bits within each of these four words is now given.

SOUDCB (Source DCB word)

Bit	Meaning
0	Source busy (currently in the process of supplying an input record to a system subprocessor).

1-2	Either not used or explanation inappropriate here.
3-5	Mode of source device.
6-8	Not used.
9-11	Coupled sink logical device number

SINDCB (Sink DCB word)

Bit	Meaning
0	Sink busy (currently in the process of receiving output--i.e., filling an output buffer).
1-8	Either not used or explanation inappropriate here.
9-11	Device number of source from which last copy occurred to this sink.

SOUBCB (Source BCB pointer)

Pointer to buffer control block (a five-word block which describes the current status of a permanent or dynamically allocated, wrap-around buffer) for the input buffer associated with this device. If this word is zero, then no input buffer currently exists. The input buffers for devices 1, 2, and 4 are permanent and, thus, always exist.

SINBCB (Sink BCB pointer)

Pointer to buffer control block for the output buffer associated with this device. If this word is zero, then no output buffer currently exists. All output buffers are dynamically allocated and released as necessary.

Example: SE 1 2 3

ENABLE n1 n2

This command is used to load the two status words in the 201 line adapter. The octal number n1 is loaded into status register 1 (LA1) and n2 into status register 2 (LA2). A description of these registers may be found in the design report for the 201 interface[6]. This command is generally used for system checkout only, although the two examples shown below are useful.

Examples: EE 4 10

EE

The first example loads 4 into LA1 which has the effect of setting Terminal Ready (a necessary step before making a call on the 201). LA2 is loaded with 10 octal (-8_{10} mod 16) which sets the character frame size to 8 bits. The second example clears both registers (LA1 and LA2), thus hanging up the 201 if a call is in progress.

TERMINALREADY

This command is equivalent to an "EE 4 10" command. It has no arguments and is provided simply as a convenience. It sets the Terminal Ready bit in LA1 of the 201 and loads LA2 for an 8-bit character frame size.

Example: TY

SET n

This command sets the core bank switch used by the ALTER and DISPLAY commands The octal argument n must be a legal core bank--i.e., n may not be greater than the highest actual hardware core bank, nor less than zero.

ALTER n1 n2 [n3 n4 ...]

This command is used to alter the contents of core memory. Because there is no hardware core-protect feature and since no software checking is performed, it is possible to alter any memory location. The user should therefore be appropriately cautious when using this command. The four-digit octal number n1 specifies the low-order 12 bits of the 15-bit address of the memory location to be altered. The high-order three bits are specified by the SET command. The four-digit octal number n2 replaces the old contents of the altered location. The optional four-digit octal arguments n3, n4 ... replace successive memory locations. The argument n1 may optionally be given as an asterisk in which case the contents values n2, n3 ... will be placed in memory successively starting at the first location after the last location altered by the preceding ALTER command.

Examples:
```
AR  6000  1  2  3
AR  7000 -10  7777
AR   *  0
```

In the first example, location 6000 in the core bank specified by the most recent SET command will be altered to 1, location 6001 to 2, and location 6002 to 3. In the second example, location 7000 will be altered to -10 (7770) and 7001 will be altered to 7777. In the third example, location 7002 will be altered to zero.

DISPLAY n1 [n2 n3 ...]

This command causes a core memory dump to be output on the coupled sink device. The arguments are all four-digit octal numbers which specify the low-order 12 bits of memory addresses. The high-order three bits of each address is specified by the most recent SET command. The dump output may be one or more lines (8 locations are displayed on each dump line) and the first number on each line is the address of the location whose contents are given as the second number on the line. Successive numbers (if any) are the contents of successive locations. If the number of arguments given is even, then they are taken in pairs as lower and upper limits of what is to be displayed. If the number of arguments given is odd, then they are taken in pairs (as above) except that the last number is taken as a single address to be displayed.

Examples: DY 0

DY 3...4

DY 7000...7015, 10 71, 100

In the first example, the contents of location zero will be displayed. In the second example, the contents of locations 3 and 4 will be displayed. The contents of locations 7000 through 7015, 10 through 71, and 100 will be displayed in the third example. Note that any sort of punctuation or blanks may separate the arguments. The core bank being displayed is that one specified by the most recent SET command.

Command: IPL

Purpose: Load a program from the disk.

Prototype: IPL n

where n specifies the disk loader program number to be loaded. If omitted, n is taken as zero so that RAMP (disk program zero) is reloaded.

Effect: The display is stopped, interrupts are disabled, and the disk loader (presumably in core) is called with the program number as an argument.

Example: IPL 1

Load disk loader program number 1.

Command: HSR

Purpose: Select the High-Speed Tape Reader

Prototype: HSR

Effect: The PC-01 reader select I/O instruction is issued and the paper tape (if any) in the reader will be read into the reader buffer. Reading stops when either a carriage return (octal 215) or an end-of-file (ASCII EM, octal 231) is read. These are the only end-of-record characters recognized by the reader routines. The reader will automatically be restarted when RAMP has processed the entire record. If an X-OFF (octal 223) is encountered, the reader will permanently stop (i.e., not automatically restart) wherever it is in the record (an X-OFF is not an end-of-record character) until another HSR command is issued. The X-OFF serves as a control character and is not placed in the reader buffer. This character is useful for delimiting the end of a physical tape.

Command: TASK

Purpose: To add a task to the system task queue.

Prototype: TASK n1 [n2]

Effect: A task with the following characteristics is added to the CPU queue. The task entry point (octal starting location in core bank zero), TSKACT, is specified by the argument n1. The source and sink logical devices for the task, TSKDCB, are set to be the same as are currently being used by the device which generated this TASK command. The task return address, TSKLNK, is set to point to location TSK6 (the location also pointed to by DELETE) so that if the task tries to return to the caller, it simply deletes itself from the CPU queue. Finally, the task argument, TSKARG, is set to n2, if given, otherwise to zero.

Command: CAMERA

Purpose: To activate a camera shutter.

Prototype: CAMERA

Effect: An I/O instruction is issued which causes the camera shutter solenoid (if attached to the machine) to be activated.

Command: GRAFACON

Purpose: To activate the Grafacon test task.

Prototype: GRAFACON [n]

Effect: The Grafacon test task is inserted on the task queue. The task will continuously read pairs of x, y values from the Grafacon and print them on logical device n in octal format. If n is omitted or zero is given for n, then the Grafacon test task(s) will stop. The primary use of this command is for hardware checkout of the Grafacon tablet.

Command: DISPATCH

Purpose: To insert a dispatch task.

Prototype: DISPATCH n

Effect: A dispatch task (DISPAT) is inserted for logical device n. The dispatch task is described in the I/O Dispatcher section of the manual. This command should be used carefully since the arbitrary insertion of dispatch tasks usually interferes with the normal operation of the I/O routines. The primary use of this command is for dispatching a utility file device (logical device numbers 3 and 6).

Command: PFX

Purpose: To enable or disable prefixing.

Prototype: PFX n

Effect: If n is zero, then the I/O Dispatcher will no longer check for any of the special characters which may occur at the beginning of input records (i.e., SOH, STX, SOBF, and SOCR). This means that any input received from any logical device number will be processed using the default mode and sink for that device regardless of the first character of the input record. If n is non-zero, which is the default condition at load time, then special characters are checked for.

Command: NAMES

Purpose: Specify the sink LDN for the result of the PM, LP, and LT commands.

Prototype: NS I

where $0 \leq I \leq 7$ specifies the appropriate LDN.

If I is omitted, zero is assumed.

Effect: Specifies the contents of the SENDLDN register (internal to RAMP).

Example: NS 4

Send names to 201.

Note: An initial value of 4 is loaded, i.e., the default devices for PM results is the 201.

Command: LT (Lightpen Track)

Purpose: Track the light pen without rubber banding

Prototype: LT [X Y]

Result: Final X Y sent to device specified by NS command.

Effect: Places light pen tracking symbol at location X, Y on the screen, where X and Y are specified in decimal raster points. If X and Y are not specified, the tracking symbol will appear approximately 1 inch to the left of the position specified by the last LE, NL, PT, PN, or CH command. (In the case of lines, the line end-point is taken to be the position specified.)

Subsequent tracking can be performed. The user may finalize a point by pressing pushbutton 11. This action will cause the current X, Y position of the tracking symbol (in decimal raster points) to be sent to the NS device. The tracking symbol is then removed.

Example: LT 512 512

Start tracking at the center of the screen.

Command: LP (Lightpen Track)

Purpose: Track the light pen with rubber banding

Prototype: LP [X_S Y_S] [X_E Y_E]

Result: Final X Y sent to device specified by the NS command

Effect: Places lightpen tracking symbol at position X_EY_E (in decimal raster points) and places the start point of the rubber band at location X_SY_S. (The rubber band ends at the center of the tracking symbol.)

If X_SY_S and X_EY_E are not specified, positioning will occur at the point of the last LD, NL, PT, PN or CH command.

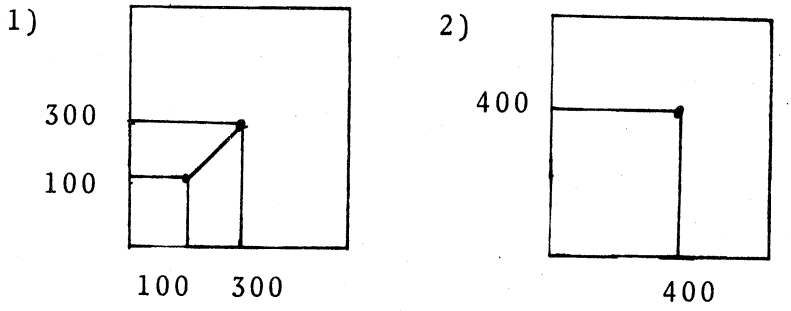
If X_SY_S is not specified, the rubber band is assumed to start at X_EY_E, i.e., it has zero length.

Pushbuttons 9, 10, and 11 are utilized in the subsequent tracking. Pushbutton 9 on constrains the rubber band to accept only vertical increments in the position of its end point at the time it was constrained. Pushbutton 10 on provides a similar horizontal constraint. As in the LT command, Pushbutton 11 is used to terminate tracking and send the final X,Y position of the end of the rubber band (which may or may not coincide with the center of the tracking symbol) to the NS device.

Examples: 1) LP 100 100 300 300

2) LP 400 400

Resulting screen



Command: PM (Pointing Mode)

Purpose: To place lightpen in pointing mode.

Prototype: PM

Result: NNNN D X Y

Where NNNN is the 4-digit octal "name" of the first entity seen by the light pen. (The name is the first location of the highest level push jump of the entity.)

D is the decimal displacement of the display order causing the light pen interrupt from the location specified by the second half of the highest level push jump (computed module 4096).

X is the X position of the light pen hit.

Y is the Y position of the light pen hit.

Effect: Sets lightpen service to identify a displayed entity (subfile) by examining the status of the pushdown stack. Waits for several hits on the same entity, thus providing a primitive check on the validity of the lightpen hit. The data above are sent to the sink LDN specified by the NS command (device 4 --the 201--is the default case). Further lightpen hits are ignored.

Command: SN (Start Again)

Purpose: A fresh screen and display file area.

Prototype: SN

Effect: Stop the display. Reset all internal pointers.

Command: LINE

Purpose: To draw an intensified line on the scope.

Prototype: LE X_S Y_S X_E Y_E [NAME]

Results: Name of line (four octal digits) returned to LDN specified by NS command if NAME not specified in the command.

Effect: Causes a line to appear between points $X_S Y_S$ and $X_E Y_E$ (in decimal raster points) on the screen. An eleven-word "line file" having the following structure is formed one level down from the high level pushjump list.

1-127	Vector mode enter data state
Y_S	Position beam to start point of line
X_S	
4001	Y=1, intensify
0	X=0
ΔY	Draw line
ΔX	
6001	ΔY=-1+intensify
4000	ΔX=0, escape
3000	Pop

The short vectors on either side of the actual line allow for easy identification when a line's end points cause the light-pen interrupt (see PM command).

The $X_E Y_E$ position is the position which is stored as a default value in subsequent LP and LT commands.

If NAME is specified, then the defined line will replace the previous line having name NAME. NAME should be a point or a line, as no error checking is done.

The name returned to the device specified by the NS command (default device is the 201) will be a four-digit octal number, which is the first location of the high level pushjump locations that cause the lower level file to be executed.

Examples: LE 1 1 1023 1023

LE 3 4 5 100 7762

Command: NON-INTENSIFIED LINE

Purpose: To draw a non-intensified line on the scope.

Prototype: NL X_S Y_S X_E Y_E [NAME]

Results: Same as LE command.

Effect: Same as LE command except nothing is intensified.

Command: POINT

Purpose: To display a point.

Prototype: PT X Y [NAME]

Results: Name of point (four octal digits) returned to LDN specified by NAMES command if NAME not specified in the command.

Effect: Causes a point to appear at raster position X, Y on the screen. The display file which is created one level "below" the pushjump is as follows:

1127	Vector Mode, enter data state
Y	Position
X	
4001	ΔY=1, intensify
0	
0	Zero length line
0	
6001	ΔY=-1, intensify
4000	ΔX=0 escape
3000	Pop

If NAME is specified, then the defined point will replace the previous point specified by NAME. NAME must be a point or a line.

Examples: PT 512 512

PT 324 243 7746

Command: CHARACTERS

Purpose: To display a character string.

Prototype: CH X Y $C_1C_2...C_n$

Results: Name of character string is returned to LDN specified by the NAMES command.

Effect: Causes positioning at point X, Y and display of the subsequent character string to the right of the position.

The structure of the display file below the pushjump is

1107	Point mode enter data state
Y	Position and escape
X+4000	
PAR	Parameter - doubles scale
PUSH	Push to system display files which
C_1	draw each character
PUSH	
C_2	
⋮	
PUSH	
C_n	
POP	

Note: The results of the PM command, in particular the displacement portion of it, will have little meaning when the light pen is pointed at a character string.

Examples: CH 512 512 MIDDLE OF SCREEN

Command: BLANK

Purpose: To temporarily blank a list of named entities.

Prototype: BL $NAME_1$ $NAME_2$ . . . .$NAME_n$

Results: None

Effect: Causes the pushjump for every named entity to be skipped. Thus subsequent removal of the skip instruction through an UNBLANK command will cause each entity to be reinserted, in its previous form, back into the display loop.

Examples: BL 7747

BL 7757 7763 7766

Command: UNBLANK

Purpose: To restore a previously blanked element

Prototype: UB $NAME_1$ . . .$NAME_n$

Results: None

Effect: Removes the skip instruction, which was inserted with the BLANK command, thus re-allowing display execution at the named entity. If an entity named in an UNBLANK command is not blanked, the command has no effect on that name.

Examples: UB 7746

UB 7757 7763 7766

Command: DELETE

Purpose: To delete a named entity from the display area and recover the storage for future use.

Prototype: DL $NAME_1$ . . . $NAME_n$

Results: None

Effect: Causes the high level push locations to be marked available for future use and the lower level storage to be compacted. Any relocation resulting from this compacting operation is handled by the system.

Examples: DL 5776

DL 6337 6341 6344

REFERENCES

1. Cocanower, A.B., The DF Routines User's Guide, Memorandum 23, Concomp Project, University of Michigan, Ann Arbor, May 1969, 5 pp. + appendices.

2. MTS Manual, 2nd ed. Computing Center, University of Michigan, Ann Arbor, 1969.

3. Mills, D.L., RAMP: A PDP-8 Multiprogramming System for Real-Time Device Control, Memorandum 5, Concomp Project, University of Michigan, Ann Arbor, May 1967, 24 pp.

4. Powers, V.M., Mills, D.L., and Laurance, N., An Assembly Language System for DEC Minicomputers, Memorandum 20, Concomp Project, University of Michigan, Ann Arbor, May 1969, 64 pp.

5. Mills, D.L., Multiprogramming in a Small-Systems Environment, Technical Report 19, Concomp Project, University of Michigan, Ann Arbor, May 1969, 41 pp.

6. Wood, D.E., A 201A Data Communication Adapter for the PDP-8, Memorandum 15, Concomp Project, University of Michigan, Ann Arbor, February 1968, 134 pp.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
THE UNIVERSITY OF MICHIGAN
CONCOMP PROJECT

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
GRAPHICS RAMP USER'S MANUAL

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Memorandum 34

5. AUTHOR(S) *(First name, middle initial, last name)*
W.S. Gerstenberger, R.W. Taylor

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1970	52	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DA-49-083 OSA-3050	Memorandum 34
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT
Qualified requesters may obtain copies of this report from DDC.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Advanced Research Projects Agency

13. ABSTRACT

The Graphics RAMP system provides basic display and communication support for a DEC 338 display system communicating over serial, synchronous, voice-grade telephone lines to a large computer. This memorandum details the various system commands and associated modules which are particular to the 338 graphics configuration. The general communication facilities of the RAMP system are documented elsewhere .

DD FORM NOV 65 1473

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
DEC 338 computer graphics terminal graphics software large-small computer configuration						