GRAPHICS MONITOR
Version 2

Steven Rubin
Donn Bihary

Computer Science Department
Carnegie-Mellon University
December 9, 1975

The Graphics Monitor is a PDP-11 program for use with the GDP2 super-display. The GDP2 consists of a PDP-11 computer, a Graphics keyboard, a cathode ray tube, a communications link to the PDP-10, and the Graphics Display Processor. The Graphics Monitor makes the GDP2 into an intelligent PDP-10 graphics terminal which can load and run PDP-11 programs passed from the PDP-10.

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense under contract number F44620-73-C-0074 and is monitored by the Air Force Office of Scientific Research.

TABLE OF CONTENTS

APPENDICES

I. Use as a Terminal

The GDP2 can be used to communicate with the PDP-10 much like any other display terminal. The user should keep in mind, however, several differences:

1) The keyboard is attached to the PDP-11, and only if the PDP-11 monitor is working can communication to the PDP-10 be established.

2) The GDP2 has a Graphics keyboard, which has more keys than a normal keyboard, and is thus used slightly differently.

3) The presence of programs on the PDP-11 makes it useful to designate certain keyboard characters (the Meta characters) to be read by the PDP-11 software and not passed on to the PDP-10.

4) The processing power of the PDP-11 makes possible certain features (e.g. the intra-line editor) not possible with "non-intelligent" terminals.

These differences are examined on the following pages.

1. System Start-Up

If the PDP-11 monitor is inoperative, it may be reloaded via the PDP-11 console switches. Starting the PDP-11 at address 173000 will load the monitor and a character set from the PDP-10. Note that this type of restart, called a bootstrap restart, will abort any running program on the PDP-10 and start another to reload the PDP-11. Minor system crashes can sometimes be fixed by a soft restart at address 1004, or a hard restart at address 1000. If DDT (the debugging package) has been loaded, it can be restarted at address 1002 (if DDT is not present, this address will do a hard restart). These do not affect the PDP-10. All of the above restarts can be done from the keyboard provided that the PDP-11 is running (see the subsection on Meta Characters).

2. The Graphics Keyboard

The Graphics keyboard has two types of keys: four special keys marked Shift, Top, Control, and Meta, and the fifty-eight other keys (called encoded keys). The special keys operate like the Shift key on an ordinary teletype; when depressed they cause no immediate action, but they change the interpretation of any encoded keys struck while they are depressed.

The special keys usually have the following interpretations: Shift is used only to get upper-case alphabetic characters. (The Shift Lock switch can be used to generate all upper case letters.) The Top key is used to get the upper symbol which appears on the encoded keys. (NOTE: The Shift key does this on most other terminals, but on this keyboard Shift applies only to alphabetic characters.) The Meta key indicates that the character is to be interpreted by the PDP-11 software and not passed to the PDP-10. The Control key is used like Control on teletypes, i.e. it indicates a special control function to be performed, not necessarily a printing character.

Throughout this manual, the term Meta Character will be used to denote the character generated by striking an encoded key while the Meta key is depressed. Individual characters in this group are denoted Meta-A, Meta-B, etc. Likewise we have Meta-Control characters when both Meta and Control are depressed. (In this case, Meta is dominant, i.e. the character is interpreted by the PDP-11.) And, of course, we can talk about Top characters, Control characters and Shift characters. All of these are keys that can be held down while striking an encoded key. Top-Shift characters are just Top characters - the shift is ignored. Top and Shift are usually ignored if Meta is depressed, although the PDP-11 programs can (and sometimes do) find out if they were depressed.

A few of the encoded keys are by their very nature Control characters, and it is not necessary to depress Control to use them as such. The keys, with their Control character equivalents, are given below.

Key	Equivalent	Function
BREAK	Control-S	Stop output
CLEAR	Control-Q	Restart output
FORM	Control-L	Form feed
VT	Control-K	Vertical tab
RETURN	Control-M	Carriage return
LINE	Control-J	Line feed
CALL	Control-C	Interrupt program
TAB	Control-I	Tabulation

3. Meta Characters

As mentioned, Meta characters and Meta-Control characters are not sent to the PDP-10. They are used to specify parameters or request action by PDP-11 software. Because it may be necessary to use Meta characters to communicate with several different parts of the monitor as well as various user programs, a convention has been adopted on the use of Meta and Meta-Control characters to avoid confusion.

Meta-Control characters are used to put the keyboard in various modes. Each mode is generally associated with one feature of the monitor or user program. (For instance, a mode to control the scroller, a mode to control the intra-line editor, etc.) Each mode has associated with it a group of **Meta characters** which causes certain actions. (For example, Meta-S can be used to set the size of the scroller text when the keyboard is in scroller mode.)

Changing the keyboard mode with Meta-Control characters associates specific actions to various Meta characters. The same Meta character may have different effects depending on what Meta-Control mode is currently in effect. User programs may establish their own Meta-Control characters (with associated Meta characters). Note that there are special Meta characters that never change in meaning. They are:

Meta-BREAK	Call DDT. If there is no DDT loaded, this has no effect.
Meta-\	Soft restart. This is useful for stopping runaway programs since it returns control to the monitor yet does not harm the user program or the display.
Meta-CALL	Hard restart. The monitor destroys everything except itself. Useful if your program is hopelessly tangled.
Meta-*	Bootstrap restart. The PDP-10 is called in to reload the monitor (this takes about 30 seconds). Necessary if your program wiped out both itself and the monitor.
Meta-Z	Clears the keyboard number (see next subsection)
Meta-DASH	Indicates negative keyboard number
Meta-0, ..., Meta-9	Entering keyboard numbers to the Meta-routines

At start-up time, all Meta characters are set to have no action (except for the special Meta characters) and the following system Meta-Control characters are available:

Meta-Control-S	Scroller control
Meta-Control-K	Keyboard control
Meta-Control-E	Intra-line editor
Meta-Control-G	Graphics control
Meta-Control-I	Input/output control

The Meta characters associated with these modes are presented later.

4. Keyboard Numbers

We mentioned before that Meta characters could be used to set parameters in the system. It is thus necessary to have some way to specify numerical values for the parameters. TECO and SOS users know that often numbers can be associated with one-letter commands by typing that number just before the command. A similiar general feature is available in the GDP system.

Typing a series of Meta digits prior to a Meta character will cause the number to be passed to the Meta routine that processes the character. Some Meta routines interpret the number as octal, others as decimal.

There are two other Meta characters used with the keyboard numbers: Meta-Z clears the keyboard numbers (useful when you make a mistake while entering the numbers), and Meta-DASH causes the following number to be negated.

For example, suppose you want to set the number of characters on a line to 50. You read in this manual that Meta-C in Scroller mode (set by Meta-Control-S) sets the number of characters on a line and that it uses a decimal value. Thus you type Meta-Control-S, Meta-5, Meta-0, and Meta-C. The Meta-Control-S puts you in scroller mode, Meta-5, Meta-0 sets the keyboard number to 50 (or 40 if read in octal but the octal value will be ignored) and Meta-C causes the number of characters to be changed.

5. Scroller Meta Characters

Normally, when using a GDP as a terminal, a user types commands to the PDP-10 and the PDP-10 executes them. This dialog is distinguished as a series of text lines on the screen. The text is scrolled by the graphics monitor in the same way as a normal terminal: the top line disappears forever when a new line is needed at the bottom and there is no room. The monitor does, however, support multiple scrollers which can be selectively displayed for multiple conversations and extended screen capacity.

This set of commands (entered with a Meta-Control-S) control the scroller. Commands that require a parameter use the decimal keyboard number unless otherwise indicated. If multiple scrollers are desired, the number of alternate scrollers should precede the Meta-Control-S. Thus, to create 19 additional scrollers (for a total of 20 scrollers) type Meta-Control-1-9-S. You may create up to 99 additional scrollers. It is not advisable to change the number of additional scrollers without doing a hard restart (Meta-CALL) but you may type Meta-Control-S with no parameter at any time and it will not change the number of scrollers.

The first three Meta commands below affect the selection and control of the scrollers. Note that at all times there is a "current" scroller for which the remainder of the Meta commands apply and on which all PDP-10 conversation appears.

Meta-P **Page set,** Set the current scroller to the keyboard number. Stop displaying the previous scroller and start displaying the current scroller. The keyboard number must be from 0 to the maximum number of scrollers defined by the Meta-Control-S.

Meta-G **Get page,** Set the current scroller to the keyboard number and display this scroller. The previous scroller is not turned off so many of them may be displayed with this command.

Meta-O **Off,** Stop displaying the scroller determined by the keyboard number. This does not affect the current scroller.

Meta-A **Scale,** Set scale to octal keyboard number. This may range from 0 (quarter scale) to 17 octal (3.5 times normal). The normal scale is 10 octal. See Appendix C for other scale values.

Meta-I **Intensity,** Set intensity to octal keyboard number. The intensity may range from 0 (which you can't see) to 17 octal (which is the normal intensity).

Meta-X **X position,** Set X-position of upper-left margin of the scrolled characters (initial value -475).

Meta-Y **Y position,** Set Y-position of upper-left margin of the scrolled characters (initial value 475).

Meta-L **Lines,** Set number of lines that can be displayed in the scroller (initially 56).

Meta-C **Characters**, Set number of characters per line (initially 98).

Meta-U **Units**, Set number of tab units per line (initially 98). Note that this parameter helps control the number of characters per scroller line. The characters parameter is simply the number of characters that may appear on a line. The units parameter determines the number of possible locations that a tab may start from. For the standard fixed width characters set, the number of units equals the number of characters. When using variable width character sets, characters may end anywhere on a line and considerably more tab characters are needed to line up the tab stops.

Meta-J **Jump**, Sets the number of extra lines to jump when the scroller fills and must be rolled up (initially 0).

Meta-S **Scale**, Set the scale parameter as in Meta-A, but also set the **Lines**, **Characters**, and **Units** parameters to values appropriate for the new scale. This command is the easy way to change the size of scrolled text.

Meta-N **Normal**, Reset all of the above parameters to their default values. For the standard fixed width character set, **Scale** is set to 10 (octal) **Intensity** is set to 17 (full on) **X position** is set to -475. **Y position** is set to 475. **Lines** is set to 56. **Characters** is set to 98. **Units** is set to 98. **Jump** is set to 0.

Meta-FORM Clear all scrolled lines except the current line.

Meta-CLEAR Like Meta-FORM, except that the current line is also cleared.

Meta-R **Retrieve**, Retrieve the nth line currently being displayed in the scroller, go to EDIT mode, and insert the text up to the carriage return in the current line. If the first character of the retrieved line is a period, an asterisk or an SOS line number, those characters are not copied. If n<0, retrieve the nth line before the current line. If n=0 then a Control-U is sent to the PDP-10 and the current line is retrieved in edit mode. (Useful if you discover the line you are typing is in error.)

6. Keyboard Meta Characters

These commands control the keyboard functions. They are initialized with a Meta-Control-K.

Meta-L **Local**, Place the terminal in local mode for non-Meta characters only. No characters are sent to the PDP-10.

Meta-N **Normal**, Reset action for non-Meta characters to normal mode. Useful when exiting Local mode. Meta-N causes all non-Meta characters to be sent to the PDP-10.

Meta-C **Clear**, Clear all the Meta characters of their actions. This is useful because each Meta-Control mode adds to the active Meta characters. Conflicting characters are replaced but old Meta characters with no equivalent in the new mode remain active.

Meta-W **Space War**, Turn on Space War mode. In this mode, the keyboard interrupts twice for each key stroke, once when the key is depressed, and once when it is released.

Meta-P **Peace**, Turn-off Space-War mode (default).

Meta-K **Keyboard lock**, The keyboard number will be OR'ed with all future keyboard input. Appendix A shows the bit pattern received for a key stroke. With the proper value to OR in, this routine can cause future characters to be treated as though some of the special keys were depressed when the character was typed.

Meta-O **Escape**, If the keyboard number is zero, turn off escape character sending. If non-zero (the default state) then send escape characters to the PDP-10 with each control character.

7. Intra-Line Editor

Normally, all non-Meta characters entered through the keyboard are sent immediately to the PDP-10. When the intra-line editor is enabled (with a Meta-Control-E), the line being typed is kept in a PDP-11 buffer so that editing may be done on that line before it is sent. At the conclusion of editing, the entire line is sent to the PDP-10 and another line may be entered.

The editor commands are modeled after the SOS intra-line edit commands, but provide easier editing in conjunction with the graphics system:

1) When editing is terminated (e.g. by typing a carriage return), the line is sent exactly as it then appears on the screen, so there can be no confusion as to the result of the editing.

2) The availability of Meta characters permits a clear distinction between text and editor commands.

The editor is turned on by typing Meta-Control-E. Text is then entered normally. The editor will maintain two cursors: the first cursor delimits text which has been sent to the PDP-10 from text that is being held in a PDP-11 buffer. The second cursor (the position-cursor) moves about in the PDP-11 buffer. Any text that is typed is entered in the buffer at the position-cursor location. Each text character typed causes the position-cursor to move one place to the right. At any time while entering text, the user may use the editing commands below to make corrections to what has been typed. When the line is complete, a break character (usually a carriage return) is typed, causing the line to be sent to the PDP-10 followed by the break character itself.

> NOTE: The "break" characters are a subset of the control characters. Break characters cause transmission of the edit line followed by the character itself. Non-break control characters are sent immediately to the PDP-10 without affecting the edit line.

All editor commands except backspace (BS key) are single Meta characters. Some commands make use of the keyboard number, denoted below as n. "Text," when used below, refers to non-Meta, non-Control characters, i.e. what is usually to be entered into the line.

Meta-Space Move right, Move cursor n characters to the right.

Meta-BS Move left, Move cursor n characters to the left.

Meta-TAB Move far right, Move cursor to the right end of the line.

Meta-RETURN Move far left, Move cursor to the left end of the line.

Meta-I Insert, Insert all following text characters at the present cursor position. Characters to the right of the cursor are pushed right to remain ahead

of the inserted text. This command is turned off by the next editor command. (NOTE: You are normally in over-write mode. Except after a Meta-I command, all entered text is entered at the cursor position, replacing any characters there previously.)

Meta-D **Delete right,** Delete n characters right of the cursor. Text on the right side of the deleted characters is moved left.

BS **Delete left,** Deletes n characters left of the position cursor. Text to the right of the cursor is moved to the left. Do not confuse this command with Meta-BS.

RETURN **Done,** This is not really an editor command, but a break character. It will cause the entire line being edited (regardless of the position cursor location) to be sent to the PDP-10, followed by a RETURN character. A new line may then be entered and edited. Note that transmission occurs from the position of the left cursor. This is usually the beginning of the line, but not always (e.g. see the Meta-B command below).

Meta-S **Skip,** This command causes the cursor to skip to the nth occurrence of the next character that you type. After the Meta-S, type one non-Meta character and the cursor will jump to the proper position.

Meta-K **Kill,** This is the same as Meta-S except that all characters from the current cursor position up to the new cursor position are deleted.

Meta-E **Exit,** Turn off the editor.

Meta-Q **Quit,** Delete the buffer and sends a line-feed.

Meta-B **Burst,** This command sends all characters up to the current cursor position back to the PDP-10. These characters are deleted from the buffer.

Meta-U **Delete everything,** Delete all characters. Position cursor at left end of buffer.

Meta-) **Delete all right,** Delete all characters to the right of the cursor.

Meta-(**Delete all left,** Delete all characters to the left of the cursor.

Meta-LINE **Retrieve,** This command retrieves the last edited line and puts it in the buffer for further editing.

needed:
meta-A
meta-F, ins
meta-M
meta-T
meta-V
meta-;

interested:
C - @ word mode
C (return) M
P - ! !
R - like C
X - TAB

8. Graphics Meta characters

These commands are for altering the Graphics processor display list. They are initialized with a Meta-Control-G. You can alter what is displayed, and what is not.

Meta-B **Blank**, Remove scroller from display list.

Meta-U **Unblank**, Put scroller back into display.

Meta-R **Remove**, Remove user graphics from display list.

Meta-P **Put**, Put the user graphics back into display list.

Meta-H **Halt**, Halt the entire display process.

Meta-G **Go**, Start the display running again.

Meta-D **Display**, Used when the display is halted. The display will be drawn once, then "Keyboard Number" of times after which it will halt again.

Meta-C **CSRset**, The graphics will have a new CSR determined by the octal keyboard number. Wrap around and clock speed doubling can be enabled or disabled. On some graphics, there is a feature called "shift" which will shift the display to the left or right. The following table shows the bits for each function and the equivalent octal values.

Bit	Octal	Function
2	4	Clock divide
3	10	Wrap around
4	20	Shift right
5	40	Shift left

9. Input/Output Meta characters

The following commands control input and output. They are initialized with a Meta-Control-I.

Meta-O Output, Reset all output conditions to the PDP-10.

Meta-T Input, Reset all input conditions from PDP-10.

Meta-P Parity, Uses keyboard number for parity control. If the keyboard number is zero, then all characters sent from the PDP-10 are treated as 7-bit ASCII and the high bit is ignored. If the keyboard number is non-zero (the default state) then the full 8-bit byte send from the PDP-10 is used.

Meta-A New ASLI, Switches you to the alternate ASLI. Thus, if the graphics has the capability of communicating with two computers, it will now communicate with the other. If there is no other computer, this will have no effect.

II. GDP Hardware

This section quickly describes the hardware and low level programming of the GDP which is necessary for users who wish to take full advantage of the GDP system. For more detail, see the GDP programmers guide.

1. Vectors and Characters

At the base level, the user will want to display a list of vectors. Vectors are represented by a pair of two's complement integers, each one representing the Delta-X and Delta-Y component of the vector. Vectors in the GDP are always relative, the next vector begins were the previous one left off. Absolute set-points (cursor positioning) can be performed with Control Words, to be discussed later. Vectors are drawn in a Cartesian plane whose coordinates are in the range of -512 to +512 (decimal) along both axis. The center of the screen is therefore at (0, 0). The screen is adjusted to have 100 points per inch.

Because core is somewhat limited, there are three formats available for storing vectors, allowing three packing densities. Long format uses two full PDP-11 words to describe each vector, the first word holds the Delta-Y component, the next word the Delta-X component. Long format is used for drawing long vectors (those longer than 1-1/4 inch: 127 units) or when you cannot tell beforehand how long the vectors will be. Next there are medium vectors. Medium format stores each component in one byte: the DY in the low byte, the DX in the high byte. Medium vectors are limited in length to plus or minus 127. Short vectors allow the greatest packing by putting each vector in one byte; the DY component resides in the lower 4 bits, the DX in the upper 4 bits. Short vectors are limited in length to plus or minus 7. They are most useful for curve approximations, wave forms, and character descriptions.

Characters may be drawn as well as vectors. Characters are represented by their ASCII values, packed one character per byte. The display buffer for drawing characters is first initialized by writing the control word to signal that characters are to be drawn (the control word is LCMD 1) and then writing the character 200 (octal) in the low byte of a full word. This character establishes the format and some other parameters of the current character set. It should be the first character in the character string (and never in a high byte as this would signal a control word).

2. Control Words

In order to indicate whether characters or vectors are to be drawn there are **Control Words**. Control Words are also used to specify several other characteristics as well (for example signaling the end of the display list). Control words are represented by a full word with a high byte of octal 200. This is a very large negative number and the user is warned never to write vectors of this magnitude as they will be mistaken for control words. There are also half-word control words, which occupy only one byte, but are recognized only when reading a short format vector list.

Control words are available for controlling the following:

- Vector intensity (16 levels)
- Gross intensity changes - (visible/invisible)
- Format (long, medium, short)
- Scale (16 scales from 1/4 scale to 3-1/2 scale)
- Indicate whether vectors or characters are to be drawn
- Set State - Do all the above at one shot
- End of list indicator
- Set cursor position (X only, Y only, or both X and Y)
- Interrupt - execute a software routine

For further detail see Appendix C.

3. Instructions

The GDP is a lot like a minicomputer in itself although the instruction set is rather small. It has only 4 instructions:

- Execute (draw) a display list (XQT)
- Jump unconditionally (JMP)
- Jump to a subroutine (JMS)
- Interrupt (INTR)

The format of these instructions is shown in Appendix D. They are composed of an address and a two bit "op-code" to indicate which of the four operations is desired. Note that the address must always point to a word boundary since the lowest bit is used in the op-code.

The XQT instruction is what causes vectors and characters to be displayed. Its operand is the address of a display list consisting of vectors, characters, and control words. The list ends with a terminator control word.

The JMP instruction merely jumps unconditionally to a new location determined by its operand.

The JMS instruction works just like the PDP-8 JMS. The operand of the JMS is an address of a buffer of instructions, but whose first word is vacant. When the JMS instruction is executed, the GDP places the return address in this first word, and execution begins on the second word in the buffer. The proper way to end a graphics subroutine is to put a JMP instruction in the last word with an operand address pointing back to the first word. When the GDP jumps to the first word a tricky thing happens. The address placed in the first word will always have 0 in the two end bits (op-codes) because addresses are always even and the highest address possible is far less than 100000 octal. This is therefore a JMP instruction! So, when we JMP back to the first word we JMP again back to the return address, and thus return from the subroutine. Note that subroutines are nestable but cannot be recursive without some additional software help (the return address would be over-written on the first recursion).

Lastly, there is INTR. In certain cases you may want to execute a PDP-11 software routine at a certain place in the display list. The INTR instruction is provided for this. The operand of the INTR is the address of the software routine (not an interrupt vector). Before the software routine ends it must set the GO bit in the Graphic's CSR (see Appendix F) so that the GDP will continue and process the next instruction. You should return with the RTI instruction (BLISS routines must be of type INTERRUPT).

Finally, the GDP will recognize the standard control words mentioned previously when it is executing these four instructions.

III. GDP Monitor Traps

The GDP monitor consists of a collection of routines to manage traffic among the PDP-10, the PDP-11, and the GDP2 processor, and to make life easier for the PDP-11 programmer. Most of the following discussion will be devoted to a presentation of the various routines, with notes on how they can be useful. As you read the following pages, you will notice that most of the Meta characters described in Section I are merely calls on these routines.

1. System Communication

User programs are loaded into PDP-11 memory along side the monitor, but no linkage of symbols between the two is possible. Thus the user must call on the monitor with the TRAP instruction. The addresses of the monitor routines are stored in the TRAP table and when a TRAP instruction is encountered, the monitor uses the operand of the instruction as an index to the table, looks up the proper address, and does a normal subroutine call. There is also a capability for user-defined routines that employ the EMT instruction. In this case, the monitor builds a table of user routines and EMT instructions associated with them. Execution of an EMT is handled by the monitor from that point on. Note that a user routine called via the EMT linkage (as with all user routines including the user's main program) should return with an RTS PC and not a RTI or RTT.

Here are four traps which can be used to access the TRAP and EMT tables, if you should have the urge.

TRPADD - TRAP 0

USE	Obtaining the address of the TRAP table.
RETURN	R0 - The address of the table (in particular, the address of the routine associated with TRAP 0).
NOTE	This trap is not guaranteed to work all of the time because your program may be running with memory-management in which case the returned value will be in terms of the Graphics Monitor's address space which your program cannot access.

EMTADD - EMT 0

USE	Obtaining the address of the EMT table.
RETURN	R0 - The address of the table (in particular, the address of the routine associated with EMT 0).
NOTE	The returned address may not always be valid for the same reason given in the last trap: the addressing space of your program and the monitor may not be the same.

EMTSET - TRAP 34

USE	Places a user routine in the EMT table
PARAMETERS	R0 - Holds address of routine R1 - Holds desired EMT number (don't use zero, the monitor is already using it).
RETURN	R0 - Set to one if successful. Cleared if R1 is too large for the EMT table (in which case no action is taken).

EMTSZ - TRAP 141

USE	Set size of EMT table (initially 16).
PARAMETERS	R0 - Holds new size (non-zero, please, or the monitor won't have room for its own EMT).
ACTION	Old table destroyed, new allocated.

2. The Activity List

When nothing else is running on the PDP-11, the routine ACTRTN is. This routine runs through the Activity List at low priority looking for things to do. A high priority interrupt performs the minimum amount to ensure that no data or vital system function is lost and then it queues the rest of the interrupt actions in the Activity List. Once a routine is placed in the Activity List, it will be executed when ACTRTN gets around to it. Although the Activity list is meant primarily for the monitor, the user may also place routines in it. After the routine is executed its entry is removed from the list. The following traps will find little use with most users but are offered anyway:

ACTVAT - TRAP 65

USE	Place a routine in the activity list. Up to 3 different routines allowed. The routine detects attempts to put the same routine address in twice and ignores request.
PARAMETERS	R0 - Holds address of routine. This routine should end with a RTS PC.
RETURN	R0 remains the same if successful, otherwise R0 set to zero.

ACTRTN - TRAP 145

USE	Returning the monitor to the idle state.
ACTION	Soft restart. This is equivalent to typing Meta-\. Note that the monitor will not return to you after you execute this trap.

3. Error Trapping

The monitor handles all errors that crop up from any source. When an error occurs, the scroller is displayed (in case it was turned off), the user display is removed (it is not needed), and a three letter error message is placed in the scroller. The monitor then goes into an infinite low-priority loop. This means that essential high-priority actions (like refreshing of graphics) will continue, but all non-essential actions (like the line clock routines and I/O with the PDP-10) will be prevented. Note that special Meta characters will be accepted so the user may do one of 4 things with the error: Meta-\ to soft restart, Meta-CALL to hard restart, Meta-BREAK to enter DDT, and Meta-* to bootstrap restart. The following table shows the possible error messages, their meanings, and possible causes.

Error	Meaning	Cause
ERR	Device error	Some piece of hardware attached to the PDP-11 did an interrupt with no address in mind. It trapped to word zero.
NXM	Non-existent memory	A reference was made to a location that does not exist in the PDP-11. This is also caused by a word access to an odd address.
ILG	Illegal instruction	A non-existent instruction was executed.
BPT	Breakpoint trap	The BPT instruction was illegally executed. This is also caused by setting the T-bit in the Processor Status.
IOT	I/O trap	The IOT instruction was illegally executed.
PWR	Power fail	The power failed (but returned since you are able to read the screen).
NST	No such trap	A non-existent monitor trap was executed.
INT	Graphics interrupt	An illegal graphics interrupt was executed.
NEC	Not enough core	A block of data from the PDP-10 could not be allocated due to insufficient core.
CKS	Check-sum error	A message from the PDP-10 was badly transmitted and its check-sum didn't add up.

4. Space Allocator

The space allocator manages all core in the PDP-11. Programs may request blocks from either of two types of space: program or graphics. Graphics space is core in the Double Port Memory, which must be used for display lists. Program space is all other memory (if the PDP-11 has all Double Port Memory and no Single Port Memory, both types of space are allocated from the same physical core). Note that whatever the core configuration, the Graphics Monitor uses the first 4K of memory and the Character sets require approximately 1.5K of graphics space. The space allocator routines use a "first-fit" algorithm for allocating space. There are also traps for de-allocating core which require the address of the block to be de-allocated.

GETSPC - TRAP 1
GTGRSP - TRAP 36

USE	Allocating memory. Use GETSPC for program space and GTGRSP for graphics space.
PARAMETERS	R0 - Number of words needed.
RETURN	R0 - Address of buffer. R0 will be zero if no space available.
NOTE	The allocator may occasionally give a few extra words to prevent fragmentation of memory.

BIGSPC - TRAP 2
BGGRSP - TRAP 37

USE	Allocates the largest buffer currently available. Use BIGSPC for program space and BGGRSP for graphics space.
RETURN	R0 - The address of the largest buffer. If no space is available then R0 will be zero. R1 - The size of the buffer in bytes.

GIVSPC - TRAP 3

USE	De-allocating memory.
PARAMETERS	R0 - Address of buffer to be de-allocated.
NOTE	De-allocated graphics buffers are held for one clock tick before they are available for re-allocation. This ensures that the GDP is finished displaying the contents of the buffer. When freeing graphics buffers, the first word of the buffer is immediately destroyed by the monitor. The user is therefore advised not to use the first word of a display buffer for the entry point of a JMS instruction.

SOMSPC - TRAP 4

USE	This routine allows the user to return the top part of a previously allocated buffer.
PARAMETERS	R0 - Holds address after which space no longer needed. R1 - Holds address of start of buffer.
ACTION	Core will be returned unless the amount of space is miniscule.

5. Utility Traps

These traps handle the miscellaneous functions of register saving, machine independent arithmetic, and machine environment information.

SAVE1 - TRAP 6

USE	Saving the registers.
PARAMETERS	R1 through R5.
ACTION	R1 through R5 are saved on the stack.

RETUR1 - TRAP 14

USE	Restoring the registers from the last SAVE1.
RETURNS	R1 through R5.
ACTION	R1 through R5 are restored from the stack.
NOTE	The programmer must be careful to insure that the stack is at the same "depth" when restoring as it was at save time.

MULT - TRAP 51

USE	To multiply two 16-bit numbers yielding a 32-bit result.
PARAMETERS	R0 - Multiplier. R1 - Multiplicand.
RETURNS	R0 - The high 16 bits of the result. R1 - The low 16 bits of the result.
ACTION	If the MUL instruction exists, it is used. If there is no MUL instruction, the EAE is tried. If no EAE exists, the multiplication is done with a standard looping program. In all cases, the results are the same.

DIVIDE - TRAP 52

USE	To divide a 32-bit number by a 16-bit number yielding a 16-bit result and a 16-bit remainder.
PARAMETERS	R0 - The high 16 bits of the numerator. R1 - The low 16 bits of the numerator. R2 - The denominator.
RETURNS	R0 - The quotient. R1 - The remainder.
ACTION	If the DIV instruction exists, it is used. If there is no DIV instruction, the EAE is tried. If no EAE exists, the division is done with a standard looping program. In all cases, the results are the same.

WHAT - TRAP 5

USE This is a capabilities trap. It tells you the environment of the PDP-11 you are on.

RETURNS R0 - A set of flags telling you the nature of the PDP-11:

Bits	Meaning
0-2	Amount of core in units of 4K
3	Amount of graphics core (0=8K, 1=16K) Note that machines with 16K of graphics core must have it placed on a 16K boundary. This means that some of it is unusable either because the monitor is occupying it or because the high part is un-addressable.
4-5	Location of graphics core in units of 8K
6-7	Type of PDP-11 (0=11/15, 1=11/40, 2=11/45)
8	If 1, indicates the presence of an EAE (on an 11/15), an EIS (on an 11/40) or an FPP (on an 11/45)
9	If 1, indicates the presence of a tablet
10	If 1, indicates the presence of an ASLI at location 310 (nominally to the AI/11)
11	If 1, indicates the presence of an ASLI at location 320 (nominally to C.mmp)
12	If 1, indicates the presence of Microstore
13	If 1, indicates the presence of Memory Management

Unfortunately necessary for our PDP 11s. ↑

6. Line Clock Traps

There is a line clock that interrupts the monitor 60 times a second. This interrupt allows synchronization of various system actions. When time becomes available a list of routines called the clock queue is executed. Both the system and users can put into and clear specific addresses in this list. The queue can hold 8 routines of which the system is never is using more than two. When a user routine is called, register R0 holds the current system 16-bit time. This 16-bit time is simply a one-word counter of the interrupts. At 60 interrupts per second, this word will overflow and go back to zero every 15 minutes (approximately). Therefore, the 16-bit time is useless as an absolute time-of-day but it does give a relative time. Note that any of the registers may be destroyed during routine execution. The user should return with a RTS PC. Note also that any routine in the clock queue will be called at each tick until it is specifically removed from the queue.

LINTIM - TRAP 46

USE	Obtaining the line time.
RETURN	R0 - The 16-bit time.

LINPUT - TRAP 47

USE	Putting a routine into clock service.
PARAMETERS	R0 - Address of routine to be entered.
RETURN	R0 - Set to one if successful, set to zero if no space available. If the routine is already in queue then LINPUT returns successful but doesn't enter the routine a second time.

LINCLR - TRAP 50

USE	Removing previously put routine.
PARAMETERS	R0 - Address of routine.
RETURN	R0 - Set to zero if the address is not found within the list of routines. Otherwise, R0 is left alone.

7. Output to the PDP-10

All output to the PDP-10 passes through a communications line currently 300 baud (30 characters/second). The PDP-11 monitor has a built-in multiplexing system for separating 7-bit character information and 8-bit image data. Eventually the PDP-10 will also have a corresponding multiplex system. Currently the PDP-10 monitor recognizes a special escape character. The character following the escape character will be used to affect changing mode, and forcing the monitor to interpret a control character even when in extended character set mode. Extended character set mode on the PDP-10 causes all 7-bit codes to pass as text characters except the escape character. Of course, escape followed by escape will always be interpreted as escape (that is the escape octal code will be passed to program). The multiplexing on the PDP-11 is transparent to the user; the user simply makes calls on the character and image mode output traps and the PDP-11 monitor performs all the multiplexing. Data to be sent to the PDP-10 is processed on a first come first served basis using one large buffer to hold data until it can be shipped. This buffer is initially 256 bytes long but it can be changed with the following trap.

OUTSET - TRAP 161 never used except by Exec

USE — For re-setting output state. All output is stopped. Output state is set to that at restart time.

PARAMETERS — R0 - Negative 1 to keep the current buffer size, otherwise the buffer size is set to R0.

ASLI - TRAP 43 This should come with the capability to buffer [illegible] input from each line [illegible]

USE — For changing the address of the ASLI. The ASLI is the communication device to the PDP-10. Each ASLI has an address in PDP-11 memory and the monitor maintains a "current" ASLI by its address. Many ASLIs may be attached to the PDP-11 and this trap will tell the monitor to use a different one.

PARAMETERS — R0 - The address of the ASLI interrupt vector in PDP-11 memory. The Unibus address of the ASLI will be obtained by adding 165000 (octal) to the interrupt vector address. Note that the first ASLI is usually at 300, the next at 310, then 320, etc.
R1 - The speed bits for the output onto the ASLI (normal 300 Baud speed uses the value 1000 octal).
R2 - The speed bits for input from the ASLI (normal 4800 Baud speed uses the value 3000 octal).

ACTION — The ASLI change will not take place until the next call to OUTSET (or the next restart).

8. Character Mode Output

Four routines are available for placing characters into the output buffer. Once the character is placed in the buffer, the system will get around to outputting it when it has time.

SLPONE - TRAP 31

USE	For outputting a 7-bit ASCII character.
PARAMETERS	R0 - Holds character in the low byte.

CONOUT - TRAP 35

USE	For outputting an escape control character. That is, the escape character (Control-P) is sent followed by the character in R0.
PARAMETERS	R0 - Holds character in the low byte.

SLPRTN - TRAP 32

USE	For outputting a 7-bit ASCII or control character. This routine checks to see if the eighth bit is set. If not then routine SLPONE is executed; otherwise, the eighth bit is cleared and the routine CONOUT is called. This routine is useful for outputting characters from the keyboard since the keyboard uses this format after transforming the keyboard character to ASCII.
PARAMETERS	R0 - The 8-bit character to output. Note that for control characters such as Carriage Return, Line Feed, Back Space, and others, you must turn on the high bit in the byte in order to send it correctly. Thus, a Carriage Return is a 215 (octal), and not 15 (octal).
NOTE	Character 220 is sent as 20 and 20 as 220. This is necessary for interfacing the escape character.

STRSLP - TRAP 45

USE	Sending a string of characters to the PDP-10.
PARAMETERS	R0 - The address of an ASCIZ string (an ASCII string terminated with a byte of zero).
ACTION	Each character in the string is passed to the PDP-10 with SLPRTN. Thus STRSLP can be used to send messages longer than one character and is more convenient than calling SLPRTN for each character.

EFCS - TRAP 163

USE	Control sending of escape character.
PARAMETERS	R0 - Zero to indicate that no escape character should be sent before control characters. Non-zero to send the escape character (Control-P) before control characters. Note that the escape character mode does not affect Control characters for image mode (next subsection).

9. Image Mode Output

Two words of data are necessary to specify an image message. The first word is a header that always proceeds the message to be sent, and the second word specifies where in core the message is found. The low byte of the header word (all words are sent low-byte high-byte by the output routine) is simply a message ID number. The PDP-11 monitor reserves the right to freely use message ID's between 1 and 77 octal. The high byte gives the number of words in the message, less the header word. This value should never get much larger than about 20 or 30. The entire message has the following format:

Header Word:	Bits 0 to 7: message ID number. Bits 8 to 15: n, the number of words in message.
Message Data:	n words of data (n may be zero).
Checksum:	One byte of data, which is minus the sum of all bytes in the header and message data. (The PDP-10 checks this to make sure that the message was received correctly.)

IMGRTN - TRAP 33

USE	For sending an image message to the PDP-10. The entire message is immediately copied into the output buffer.
PARAMETERS	R0 - Low byte holds the message ID. High byte holds the size of the message in words. If the size is zero, then only the message header is sent (i.e. the parameter in R1 is ignored). R1 - Holds address of message. Beginning at that location the number of words specified in the header will be sent. NOTE: following the header word and the data words of the message, a one byte checksum is sent (minus the sum of all the bytes that are sent in header and data words).

10. Input From the PDP-10

All input from the PDP-10 is via a 4800 baud (480 characters/second) line. The monitor services all characters from the PDP-10. Up to 100 8-bit characters can be buffered until the system has time for servicing them. The service routine can be used to send characters to the scroller (the default state) or interpret them as a message; and, in addition, the user can directly take control of the service routine to read data coming from the PDP-10.

The default state for the servicing routine is to interpret the 8-bit bytes from the PDP-10 as 7-bit ASCII characters to be sent to the scroller display. When the character 1 (downarrow) is followed by the character 0 (null), then an image mode message begins and the subsequent characters are no longer sent to the scroller. The user can at any time execute the routine TENSET, which clears the input buffer and resets the link to send characters to the scroller (default mode).

TENSET - TRAP 162

USE Full reset of the input state from the PDP-10.

PARITY - TRAP 42

USE Setting the parity mode on input from the PDP-10.

PARAMETERS R0 - If non-zero then full 8-bit characters are send to the scroller. If zero (the default state) then input characters are stripped of their high bit before being sent to the scroller.

LNKTAK - TRAP 41

USE Used to reset the routine which processes bytes from PDP-10.

PARAMETERS R0 - If zero then normal monitor handling is restored. Otherwise, the address in R0 is the new routine to be called for each byte to be processed.

11. Messages from the PDP-10

Messages from the PDP-10 consist of input data, some action to perform on it, and return results. Typically the action to be done is a call to a monitor trap or a user subroutine. The message can then be viewed as a subroutine call with input parameters and returned values. Of course, the parameters and return values are optional. The input parameters may be stored in one of four places: in program space, in graphics space, on the stack, or in the registers. The subroutine invocation consists of a one-word instruction that will be executed once the data is stored. The return results consist of shipping some of the PDP-11 registers back to the PDP-10 after the instruction is executed. All of these options are selected in the message header. The messages look like this:

Byte	Value
1	1
2	0
3, 4	Instruction word to execute on PDP-11
5, 6	Control word (see below)
7, 8	Message size, n
9 to 2n+8	Data words (parameters)
2n+9	Checksum for entire message

The **Instruction word** will be executed after the data is received. It should be an EMT or a TRAP instruction but currently any other PDP-11 instruction will work. Note that the TRAP instruction calls a monitor routine and the EMT instruction calls a user subroutine (see EMTSET, page 15).

The **Control word** determines where the input parameters will go and how many output registers will be returned. It has the following format.

Bits	Meaning
0 and 1	Method of parameter passing: Both bits off for Buffer passing (value=0) Bit 0 on, bit 1 off for Stack passing (value=1) Bit 0 off, bit 1 on for Register passing (value=2)
2	Use graphics space for buffer allocation if set
3	Return registers to PDP-10 if set
4 - 6	Number of registers to return (if bit 3 set)

In Buffer passing, a buffer is allocated from graphics or program space to hold the data words. When the Instruction word is executed, R0 will hold the buffer's core address. Register R1 will hold the number of data words in message. The buffer will be allocated from program or graphics space unless bit 2 of the Control word is set in which case the buffer will be taken only from graphics space. For Stack passing a buffer must also be allocated to hold the data words. Prior to Instruction execution the data words will be pushed down on the stack. Both Buffer and Stack passing require at least one data word. For Buffer and Stack passing, if no space is available for the buffer then an error will be issued on the screen (see subsection on Error Trapping). Register passing does not require a space allocated buffer, and the number

of data words can be from zero to six. After the Instruction is executed, the values of the hardware registers are saved so that one message may pass information to the next through any of six registers.

If bit 3 of the Control word is set, then, after the Instruction execution, a message with ID=22 and size specified by bits 4, 5, 6 of the Control word will be sent to the PDP-10 (see IMGRTN, page 26). If the Instruction execution does not change the registers, then the value of the registers after the Instruction execution of the previous message will be sent. This is because the values of the registers are saved from message to message.

As an example, suppose that the PDP-10 program wanted to know if there was a 4000 word buffer of program space available on the PDP-11. The program would send two messages to the PDP-11: the first would execute a GETSPC (TRAP 1) to allocate 4000 words. Input to this message would be one register: R0=4000. Output would be one register: the results of the TRAP telling if it was successful. The PDP-10 program now knows whether the buffer is available, but if it is, the buffer must be de-allocated before it can be used, so a second message must execute a GIVSPC (TRAP 3) to release the space. No input is needed for this trap since the address of the buffer is still in R0 from the last message (the GETSPC) and, conveniently, that is just what GIVSPC requires as input. The two messages would look like this:

Byte	Value	Meaning
1	1	This is the start of message 1
2	0	
3, 4	104401	This is octal for TRAP 1: GETSPC
5, 6	32	This stores input in registers, and requests 1 register
7, 8	1	The message size
9, 10	4000.	The parameter to GETSPC
11	253	Checksum

Now to deallocate the buffer:

12	1	This is the start of message 2
13	0	
14, 15	104403	This is octal for TRAP 3: GIVSPC
16, 17	2	Input is in registers, no return value
18, 19	0	No input
20	161	Checksum

Note that if you are using the GLSTER package then you don't have to worry about computing the checksum.

12. Loading Programs and Data

Blocks of data to be loaded into PDP-11 core use the following routines. The loader has two state variables which save the addresses of the program and graphics space sections that have been loaded. Note that LINK11 creates messages that use these traps to allow load time relocation of programs and data. Therefore no user program needs to use these traps. They are only presented for completeness.

LOADR - TRAP 7

USE	For loading relocatable data or programs, using buffered messages.
PARAMETERS	R0 - Set by message handler to point to start of buffer passed from the PDP-10. Note that the message handler determines whether or not to use graphics or program space for the buffer. This information is used in determining which state variable to set.
ACTION	The state variables are updated.

LOADA - TRAP 10

USE	For loading absolute locations of PDP-11 core using buffered message mode.
PARAMETERS	R0 - Set by message handler: the address of the data. R1 - Set by message handler: the number of words in the buffer. The first word of the buffer is the absolute address to load the data. The remaining words are the successive data words to be loaded.
ACTION	The data words are loaded last to first. This feature is useful when loading graphics data where the display list must always have a terminator.

RELOCA - TRAP 54

USE	RELOCA is used just after calls on the routine LOADR. RELOCA is called using a buffered message from the PDP-10. The buffer is automatically removed from core when the routine finishes.
PARAMETERS	R0 - Set by message handler: the address of the relocation buffer. R1 - Set by message handler: the size of the relocation buffer.
ACTION	The first word of the buffer is a control word indicating what kind of relocation to perform. If bit 7 is off, the last program space buffer will be relocated. If bit 7 is on, the last graphics space buffer will be relocated. If bit 15 is off, the relocation will be done relative to the last program space module. If bit 15 is on, the relocation will be done relative to the last graphics space module. The second word of the buffer is a relative address of the start of relocation. The remaining words of the buffer are a bit pattern to denote which successive words to relocate as in the following pseudo-program:

```
INTEGER Address, Bitword, Bit;
IF Bit7 of Word1 THEN Address ← LAST-GRAPHICS-SPACE-ADDRESS
    ELSE Address ← LAST-PROGRAM-SPACE-ADDRESS;
Address ← Address + Word2;
FOR Bitword ← (each successive word of buffer) DO
BEGIN
    FOR Bit ← (each bit in Bitword (right to left)) DO
    BEGIN
        IF Bit THEN
            IF Bit15 of Word1 THEN
                Memory[Address] ←
                  Memory[Address] + LAST-GRAPHICS-SPACE-ADDRESS
            ELSE
                Memory[Address] ←
                  Memory[Address] + LAST-PROGRAM-SPACE-ADDRESS;
        Address ← Address + 2;
    END;
END;
```

LOADTR - TRAP 11

USE	Serves as an "end block" for loads. Usually called using register passing message. This starts the user program that has just been loaded and relocated in the PDP-11.
PARAMETERS	R0 - Holds transfer address. R1 - Holds flags: bit 15 signals relocation of the transfer address by one of the state variables. Bits 0 and 1 determine which state variable to use (if they are equal to 1, the last program space address is used, and if they are equal to 2, the last graphics space address is used). If bit 15 is off, the address is taken as an absolute address and is not relocated. Bit 7 says that the state variables should be cleared, signifying a LOADTR has occurred. If the state variables are zero and Bit 7 of R1 is set then no transfer takes place.
ACTION	The user's main program is called with a JSR PC. It should return with an RTS PC.

13. Intra-Line Editor

Most of the actions of the Editor have been previously described in Section I. The Editor has two traps.

BRKSET - TRAP 146

USE	Setting up a control character break table. The 26 control characters (plus some others) may or may not cause breaks (i.e. bursts of the input buffer to the PDP-10). Currently, Control-C, G, J, K, L, M, U, Z, and Altmode will break. A single bit in a break table will cause a control character to break. This routine sets up such a break table.
PARAMETERS	R0 - Bits 0 to 15 are for ASCII break characters 0 to octal 17. The initial value is 36210 octal. R1 - Sets the ASCII break characters from octal 20 to octal 37. The initial value is 6040 octal. For example, if bit 5 of R0 is on, then Control-E will "break"; and if bit 0 of R1 is on then Control-P will "break". Note that bits are numbered right to left for PDP-11's. (This is the reverse of numbering for PDP-10's.)

ETASET - TRAP 40

USE	Allows the user program to retrieve the finished edit buffer.
PARAMETERS	R0 - If zero then restore normal sending of the edit buffer to the PDP-10 (routine SLPRTN). Otherwise R0 holds address of routine to service each character.
ACTION	When the user is finished editing the buffer, it will be passed, character by character, to this routine. The character passed will be in R0.

14. Characters

There are 256 characters in the graphics character set. The following table shows the conventions used:

Character	Conventional use
0	Null character. This character displays nothing and does not change any position or state.
1-177	Standard ASCII characters. Note that some of these characters are null, such as the tab. These characters are handled specially by the other characters below.
200	Initialization character. This character, which may not appear in the odd byte of a word because it will make graphics think that the word is a control word, initializes the status for the other characters in the set. For example, it must turn intensity on for character sets that assume intensity on at the start of each character. It may also set the scale, intensity, and vector format. This character will be the first one in the scroller.
201-247	Available for user defined characters.
250-367	Used for tabulation. Note that each character set has a different number of tab stops and accordingly, uses a different number of the characters from 250 to 367. The use expands downward from 367 so that a character set that only uses 8 of these (as does the standard fixed width set) will use 360 to 367. In this case, the characters 250 to 357 are also available for user defined characters.
370	Start-blink. All characters after this, up to a stop-blink or the end of the display, will blink on and off every quarter of a second.
371	Stop-blink.
372-374	Reserved for expansion.
375	Carriage-return with line-feed. This is effectively a combination of characters 15 and 12 and is used by the monitor in the scroller.
376	Cursor. This character will be placed at the last position in the scroller. Its vector description should start and end in the same place.
377	Carriage-return with line-feed and cursor. This is effectively a combination of characters 15, 12, and 376 and is used by the monitor in the scroller.

Carriage returns are performed using a SETX control word. Since the monitor must be able to dynamically change the value of the carriage return position, the SETX control word must always be at the same position in all characters with a carriage return (characters 15: standard carriage return, 375: carriage return with line feed, and 377: carriage return with line feed and cursor). The SETX control word is the first word of the character and the second word is the X position of the left margin.

15. Tabs

When a tab character is received by the scroller (ASCII code 11,) the PDP-11 monitor substitutes another character for the tab. This character varies depending on the distance to the next tab stop because the character draws an invisible line to that stop. To determine which character to substitute, the following procedure is used. Variable width characters must have the character's width in screen units specified for each character. This number must be stored in the memory word immediately preceding the characters "vector description" (tabs should not have this value). Besides containing character descriptions and a character dispatch table, a character set must also supply two data words located immediately before the dispatch table when the character set is loaded. The first data word denotes that the character set is variable width if zero. If non-zero, then the character set is fixed width. The other data word specifies the first tab character. Now, the last tab character is always character 367 (all character numbers are octal values). If the lowest tab character is 360, then we have the usual 8 tab characters. If the lowest tab character is 250 then we have 80 tab characters (80 distances that we can tab). For the default fixed width character set only 8 tabs are needed, since there are only 8 possible distances to the next tab character. Tabs are always performed by drawing invisible vectors. Let u be the number of character Units already in a scrolled line (or edit buffer), and let t be the number of different tab locations, then ((u MOD t)+Starting-Tab-Character) is the tab character chosen for tabbing to the next character position.

Consider a fixed width character set with each character 10 screen units wide. Since there are 8 possible distances to the next tab position, 8 tab characters are needed. Suppose the character is made with medium vectors (the standard character set has short vectors). The tab characters for such a character set would be:

```
CHR360:  .BYTE 10, 0
CHR361:  .BYTE 10, 0

   .       .       .

CHR367:  .BYTE 10, 0
         100000              ; Terminator
```

For a variable width character set that allows characters to end on any raster position, 80 tabs are needed (assuming that tab stops are 80 raster units apart on the screen) and they would be:

```
CHR250:  .BYTE 1, 0
CHR251:  .BYTE 1, 0

   .       .       .

CHR367:  .BYTE 1, 0
         100000              ; Terminator
```

16. Character Sets

When the system is bootstrap restarted the PDP-10 sends the monitor to the PDP-11 followed immediately by the character set. The name "GRAPH.GST" is the standard character set name. If a user is logged on, and he has a file by the name "GRAPH.GST", then this file will be sent to the PDP-11 as the character set. If the user does not have a "GRAPH.GST", then the system character set "SYS:GRAPH.GST" will be loaded. The standard character set, GRAPH.GST, is fixed size (10 screen units high by 10 screen units wide with 7 screen units between each line). There are currently two other character sets on [A630KS00]: BODONI.GST and NGR20.GST. These are both variable width character sets. Whenever the system is Hard restarted at address 1000, the current character set is preserved (the space allocator destroys all other buffers).

When processing a character, the graphics processor requires a character dispatch table. If a character has value n, then the value contained in address (Dispatch-Table + 2*n) is used to process the character. If this value is even (bit 0 equals 0), then the value is the address of a vector description of the character. If the value is odd, then bit zero is ignored and the resulting even address must point to an interrupt routine. The only distinction that the PDP-11 monitor makes beteen "vector-described" characters and "interrupt" characters is that interrupt characters always have zero width. The graphics monitor sets aside the top 256 words of graphics memory for the character dispatch table. The user need not know this address, however, because the following traps are available to change values in this table.

CHRSET - TRAP 153

USE	Allows user to make single changes to the character dispatch table and thus define new characters. You may also retrieve an entry in the dispatch table.
PARAMETERS	R0 - Holds the address of the new character. If R0 is zero, then the null character is used. If R0 is negative then do not replace the character; only return address of character.
	R1 - Holds the character to be changed, deleted, or examined.
RETURN	R0 - The value that was previously in the dispatch table.

CHRLOD - TRAP 156

USE	This routine facilitates loading of a new character set.
PARAMETERS	R0 - The address of the new character set.
	R1 - The address of the new dispatch table.
ACTION	The new character set (in the format previously described) is loaded and the old one is deleted.

The most likely method of passing a character set to the PDP-11 is by loading a completely new description from the PDP-10. The program SILOS[A630KS00] is a character set editor that produces a ready-to-load PDP-11 format as follows:

1) Character set vector descriptions and any "interrupt" characters.

2) Width indicator. This is non-zero for a fixed width character set and 0 for a variable width one.

3) Tab location value. This is the character number of the first tab characters. For the standard fixed width set, it is 360 (octal).

4) The dispatch table. This must be a 256 word block of character addresses. All addresses must be given a reasonable value (repeat the null character for undefined entries).

5) Code to load the character set:

```
START:  MOV   #Dispatch Table, R1    ; R0 already holds relocation value
        TRAP  156                    ; This is CHRLOD
        RTS   PC
        .END  START                  ; End of program: transfer to START
```

Note that for convenience, several other functions can be done when the new character set is initialized. For example, the number of lines in the scroller, units per line, and other related information could be set. SILOS generates code to update all of these parameters after the new character set is loaded.

17. The Scroller

Unlike a hard copy teletype in which a continuous record of output is kept, the graphics monitor maintains a display list of characters returned from the PDP-10. This display list is composed of lines of characters begining at the top of the screen. As new lines are entered at the bottom of the screen the lines on the top are removed. Characters are entered into the "current" line using the routine SCRRTN. As each character is entered into the current line, the number of units is kept in order to tab properly (see subsection on Tabs). A new line, that is a new buffer, is allocated for one of three reasons:

1) Line feed. Each occurrence of a line feed terminates the current line and a new buffer is fetched.

2) Character overflow. Whenever the number of characters in a line exceeds the maximum, a carriage return/line feed is inserted and a new buffer is fetched.

3) Unit overflow. Whenever the number of Units in a line exceeds the maximum a carriage return/line feed is inserted, and a new buffer is fetched. This is done to make each line finish at the same place for variable width character sets.

The following routines are available for affecting the scroller.

SCRRTN - TRAP 120

USE	Entering 8-bit characters into scroller.
PARAMETERS	R0 - Low byte of R0 holds the ASCII character to be entered.

STRSCR - TRAP 44

USE	Entering a string of characters into the scroller.
PARAMETERS	R0 - The address of an ASCIZ string (an ASCII string terminated with a byte of zero).
ACTION	Each character in the string is displayed with SCRRTN. Thus, STRSCR can be used to display messages longer than 1 character and is more convenient than calling SCRRTN for each character.

CLEAR - TRAP 134

USE	Removes all scrolled characters from the display. The screen is cleared.

FORM - TRAP 104

USE	Works like a CLEAR except that the "current" line is left in the display.

There are eight basic parameters to the scroller. The following table shows the default values for the standard fixed width character set.

X	Carriage return position. Default -475.
Y	Top of screen. Default 475.
SCALE	The scale for graphics. Default 10 octal.
INTENSITY	The intensity for graphics. Default 17 octal.
LINES	Number of lines allowed in scroller. Default 56.
CHARACTERS	Number of characters per line. Default 98.
UNITS	Number of tab units per line. Default 98.
JUMP	Number of lines to jump when scroller reaches maximum. Default 0.

SCRNEW - TRAP 111

USE Resets all of the above parameters to the default value.

Each of the parameters can be set individually. The following table describes eight separate routines. The new parameter value should be put in R0 prior to calling the trap.

Name	TRAP	Parameter Affected
XCHN	112	X
YCHN	113	Y
SCLCHN	115	SCALE
INTCHN	114	INTENSITY
LINCHN	116	LINES
CHRCHN	117	CHARACTERS
UNICHN	101	UNITS
JMPCHN	16	JUMP

SCRDEF - TRAP 30

USE For setting the defaults for all the above 8 parameters.

PARAMETERS R0 - Holds the address of 8 word buffer of values to be given to the above 8 parameters, respectively. I.e. first word changes value X, and second alters Y, and so on.

ACTION All subsequent SCRNEW calls will reset the 8 parameters to these values.

18. Multiple Scrollers

The monitor has the facility to store up to 100 different scrollers and selectively display any of them. At any time, there is a current scroller for which the traps in the previous section apply. The following traps allow the user to control the creation and selection of the scrollers.

PAGSET - TRAP 20

USE	For setting the number of additional scrollers that the monitor is to handle.
PARAMETERS	R0 - The number of additional scrollers (from 1 to 99).
ACTION	Space is allocated for a current scroller and the requested number of additional scrollers. The current scroller becomes scroller 0.
NOTE	It is unwise to execute this trap more than once without hard restarting because storage cannot be released from the previous PAGSET.

PAGGET - TRAP 17

USE	Selecting and displaying a scroller.
PARAMETERS	R0 - The number of a scroller (from 0 to the number of additional scrollers available).
ACTION	The requested scroller is displayed and selected as the current scroller. No action is taken if the scroller number is invalid.

PAGOFF - TRAP 77

USE	Turning off the display of a scroller.
PARAMETERS	R0 - The number of a scroller to be turned off.
ACTION	The requested scroller is no longer displayed. If it is the current scroller, it continues to be so. If the scroller number is invalid, nothing is done.

19. Graphics Display Components

The monitor maintains the basic display list for the GDP. The display loop is as follows:

JMS Scroller	Display the scroller
JMS Editor	Display the editor
JMS User	Display for the user (may be XQT)
SETXY, 0, 0	To keep an idle display mostly in the center of the screen
INTR Refresh	Wake up the PDP-11, wait for refresh tick

For an explanation of these and other GDP instructions, see Section II on GDP hardware. The following routines are used to affect what is placed into the three display locations. The first set controls the scroller position. The user may turn on the standard scroller, turn it off, or display his own.

SCRCLR - TRAP 71

USE	Turns off the Scroller.
ACTION	Removes the instruction in the scroller position. (I.e. a jump to the next location is put in.)

SCRDIS - TRAP 73

USE	Allows a user-defined scroller to be displayed.
PARAMETERS	R0 - Holds instruction to put into the scroller position.

SCRUNB - TRAP 107

USE	Turns on the standard scroller.
ACTION	The standard scroller is replaced.

The next set of traps controls the user display. The user display may be filled, cleared, or restored from the last clear.

USECLR - TRAP 72

USE	Turns off the user display and saves the instruction that was there.
ACTION	The user instruction is removed. If it contained an active display, it is saved.

USERET - TRAP 144

USE	Undo a USECLR.
ACTION	Returns the active user display word that was cleared by the last USECLR.
NOTE	The USECLR/USERET traps save only a single address so "nesting" of saved user display lists does not work.

USEDIS - TRAP 74

USE	Filling the user display.
PARAMETERS	R0 - The value to be put into the display. Thus, R0 should be an address with the opcode bits for a JMS or XQT.

The next set of traps affect the editor display. The average user should find little use for these traps.

EDCLR - TRAP 75

USE	Turning off the editor display.
ACTION	The editor position is cleared.

EDDIS - TRAP 76

USE	For putting a value into editor position.
PARAMETERS	R0 - Holds the value to be inserted into the editor position.

This last trap is used to obtain the address of the entire display which starts with the scroller word. Once again, the average user will find rare use for this trap.

GRAADD - TRAP 27

USE	Obtaining the address of the Scroller, Editor, and User displays.
RETURN	R0 - The address of a three word buffer containing the address of the scroller display, the editor display, and the user display.

20. Graphics Display Control

The monitor normally attempts to run the GDP at a 60 Hertz refresh rate. To do this, it waits until the line clock tells it that a sixtieth of a second has elapsed, and then starts up the GDP for another cycle. If the display list is very long, the GDP cannot complete it in this time. In this case, the monitor restarts the GDP immediately when the display finishes instead of waiting for the line clock to "tick". If the monitor discovers that the display list is longer than 1/3 of a second (20 ticks) then it assumes that the display list is garbaged and forces a refresh from the beginning of the display list. The user can control refreshing with the following traps.

HALDIS - TRAP 130

USE	Halt graphics display after the current refresh.
ACTION	The GDP will not be restarted after the current refresh finishes.

GODISP - TRAP 131

USE	Start graphics display normally.
ACTION	The normal 60 Hertz refreshing will be started after the next line clock tick.

DISPIT - TRAP 132

USE	Controlled start up graphics.
PARAMETERS	R0 - If negative, start 60 Hertz refresh cycles after the next tick. If positive, display R0 refreshes at 60 Hertz and then stop refreshing.

GRASTP - TRAP 100

USE	Halt graphics display instantly.
ACTION	The processor is immediately halted and is not refreshed any more.

21. Graphics Interrupts

A graphics control word interrupt traps to address 104. This interrupt is serviced by the monitor which extracts the 4 bit operand and selects a routine from a 16 word table of routines. The graphics display "pauses" during an interrupt. When the routine returns via an RTS PC, the graphics will be continued from the paused state if R0 is non-zero. The interrupt routine should be very short, and should not make calls on any monitor routines that could cause "race" conditions i.e. starting and stopping graphics. The following table shows the pre-defined meanings of interrupts 0 through 7 (the user may fill interrupts 10 thru 17 octal).

Interrupt	Meaning
0	Blink On
1	Blink Off
2	Turn on Clock Divide
3	Turn on Wrap Around
4	Turn on Shift Right
5	Turn on Shift Left
6	Clear Clock Divide, Wrap Around, Shift Right, Shift Left
7	Reserved for monitor

Note that Interrupts 0 and 1 (Blink On and Blink Off) must come in matched pairs with the items to be blinked laying between them. Blinking is accomplished by setting intensity zero every half a second and restoring it to its last intensity a quarter of a second later. The blinking characters 370 and 371 (octal) merely call on these interrupts.

INTRST - TRAP 15

USE Put a routine into the interrupt dispatch table.

PARAMETERS R0 - Holds the number of the routine. This value should be between 10 and 17 octal (codes 0 to 7 are reserved for monitor).
R1 - Holds the address of interrupt routine. Note that if the graphics processes an undefined interrupt then an "INT" error message will appear on the screen (see subsection on Error Trapping).

22. Graphics Registers

The following two traps allow you to access any of the Graphics Registers. Note that you cannot access these registers while the GDP is running so these traps must be called during an interrupt routine or while the GDP is halted.

GETREG - TRAP 67

USE — Obtain the value of any of the graphics registers.

PARAMETERS — R0 - A bit pattern of the registers to obtain. The low 10 bits of R0 will indicate which of the 10 registers to read. If a bit is on, the register will be read. The bits correspond as follows:

Bit	Register	Bit	Register
0	Control Status Register	6	Character Buffer
1	Program Counter	7	Character Pointer
2	Vector Pointer	8	Vector Buffer
3	Interrupt Status	9	X Position
4	State	10	Y Position
5	Character Dispatch Table		

R1 - The address of a buffer in which to place the register values. Note that this buffer need only be as large as the number of 1 bits in R0 since it will be filled sequentially with the requested registers starting at bit 0 of R0.

LODREG - TRAP 66

USE — Change the value of any of the graphics registers.

PARAMETERS — R0 - A bit pattern of the registers to load (see GETREG).
R1 - The address of a buffer of the new register values.

NOTE — Do not load the Program Counter, Vector Pointer, or Character Pointer.

The following two traps are provided to work with the Graphics Control Status Register. You can use these traps to access the CSR while the GDP is running because they do not take effect immediately: they wait for the GDP to pause.

LODCSR - TRAP 12

USE — Change the CSR register of graphics.

PARAMETERS — R0 - The new CSR value (see Appendix F).

NOTE — You may only set the clock divide, wrap around, shift left, and shift right bits.

GETCSR - TRAP 53

USE — Obtaining the value of the Graphics CSR.

RETURNS — R0 - The Graphics CSR (see Appendix F).

23. Graphics Keyboard

The Graphics Keyboard can generate 1024 characters because of the many shift keys. The bit pattern, therefore, is a 10-bit field spread across a 16-bit word as shown in Appendix A. Most user routines will not have to worry about this format since the monitor converts it to ASCII as soon as possible. There are times, however, when the user will receive the character in Graphics format and will want to convert it to ASCII. The following trap is provided:

KEYASC - TRAP 64

USE	Transform keyboard value to ASCII.
PARAMETERS	R0 - Holds Graphics keyboard character to be decoded.
RETURN	R0 - Holds the ASCII equivalent.

When the monitor receives a character from the keyboard, it OR's in a user-settable mask of bits before interpreting the character. With this mask, the user can, for example, fake the existence of a "Meta-lock" which is why the trap is sometimes called METALK.

KEYLOK - TRAP 26

USE	Locking and unlocking software bits.
PARAMETERS	R0 - This value will be OR'ed with all future input from keyboard. If R0 is passed with the value 100000 octal, for example, then all input will be as if the Meta key were held down.

Note that the user is not expected to handle interrupts in order to read characters from the keyboard. Rather the mechanism of Meta characters should allow a PDP-11 program to adequately interface the keyboard. The first action to occur when the monitor receives a character (even before OR'ing in of the KEYLOK) is a check for the special Meta characters. These were previously described in Section I, but are repeated below.

Meta-\	Soft restart at 1004
Meta-BREAK	DDT restart at 1002
Meta-CALL	Hard restart at 1000
Meta-*	Bootstrap restart 173000

Meta-BREAK is useful only if the following routine has been executed (DDT executes it for you).

DDTCAL - TRAP 133

USE	Pass the address of a debugging routine.
PARAMETERS	R0 - Holds address of debug routine.
ACTION	When a Meta-BREAK is typed, this routine will be called with a JSR PC.

After the special Meta-characters are checked, the character is passed to the keyboard handler. This routine is normally SKIRTN (see below) but is controlled by the following traps:

SKISET - TRAP 70

USE	Obtaining complete control of the Graphics Keyboard.
PARAMETERS	R0 - The address of a routine to be called when any character is typed. Upon entry to the routine, the character (in Graphics format) will be in R0. The routine should return with an RTS PC.

SKINWR - TRAP 63
SKIWAR - TRAP 23

USE	Initialize the keyboard.
ACTION	This routine will undo the effect of a SKISET, causing SKIRTN to process all characters thereafter. The only difference between SKIWAR and SKINWR is that SKIWAR will set a bit in the status word that causes an interrupt to occur each time the bit pattern from the keyboard changes. Thus in "war" mode, depressing the character "A" will cause an interrupt, and releasing the key "A" will cause another interrupt.
NOTE	The "war" bit is occasionally called the "space war" bit; do not confuse the bit with the game of space war. Also as a side affect to both of the routines: any currently pending character is lost.

The following routines handle all keyboard input. You may use them to fake characters that you don't want the user to have to type (i.e. kicking off Meta-Control routines).

SKIRTN - TRAP 55

USE	SKIRTN is routine which processes all keyboard characters.
PARAMETERS	R0 - Holds the character to be interpreted.
ACTION	If the ASCII conversion bit is off (bit 9), then the character is assumed to be in Graphics format and the low byte of R0 is converted to ASCII using KEYASC. To stop conversion to ASCII simply set bit 9.

SKISOM - TRAP 126

USE	Allows a buffer of 16 bit characters (in Graphics keyboard format), to be processed.
PARAMETERS	R0 - Contains the address of the buffer. R1 - Contains number of words to process.
ACTION	Each word in the buffer is processed using SKIRTN. The buffer is released after the processing.

The keyboard has associated with it a state which includes the keyboard numbers, the non-Meta-character handler, the Meta-Control table and the Meta table. The keyboard numbers were previously discussed in Section I. The non-Meta-character handler can be one of three routines: the default routine, SLPRTN, which sends characters to the PDP-10; the local routine which merely echos characters; or a user defined routine that may do as it pleases.

SKINOR - TRAP 60

USE	Returns to normal character processing.
ACTION	Set the value of the non-Meta-character handler to the default value: SLPRTN (which sends ASCII characters to the PDP-10).

SKILOC - TRAP 22

USE	Put keyboard into local mode.
ACTION	The non-Meta-character handler becomes a local routine which does not send characters to the PDP-10 but merely displays them on the screen.

SKIGET - TRAP 56

USE	For obtaining the address of the character handler.
RETURN	R0 - The address of the non-Meta-character handler.

SKIPUT - TRAP 57

USE	Changing the value of the non-Meta-character handler.
PARAMETERS	R0 - The address of the new non-Meta-character handler.
NOTE	Upon entry to the character handler, R0 will contain the ASCII value of the character just read. To obtain Graphics keyboard format, use the KEYLOK procedure to turn on bit 9. This will cause the monitor not to convert input characters with KEYASC and you will receive the Graphics format. Also, note that control characters will come in with bit 7 set to indicate that they are control characters. Thus a backspace will be a 377 (octal) and a carriage return will be a 215 (to mention a few).

To summarize, most users need only Meta characters to control programs. See the next section for how to set them up. If you need to take over total control of the keyboard (Meta and non-Meta-characters) use SKISET. If you want to intercept all non-Meta-characters, use SKIPUT.

24. Meta Tables

Meta tables are the principal means of communication between a PDP-11 program and the keyboard. They contain addresses of routines that will be called when their associated Meta characters are struck. At system startup the tables are cleared to zeros, then the standard system Meta-Control characters are copyed into the Meta-Control table. Each Meta-Control character has a routine and a set of Meta characters associated with it. When a Meta-Control character is struck the monitor copies its table of Meta characters into the monitor's Meta table. The following is the format of a Meta-Control character table (where all characters are in Graphics format and not ASCII).

```
TABLESTART:
            C1          ; example of table
            C1ADDR      ; first character
            C2          ; address of Meta character routine
            C2ADDR

            . . .

            Cn'th
            CnADDR
            0           ; this signals the end of the table
            MCROUT      ; Meta-Control routine
```

Note that at the end of the table is the address of the Meta-Control routine. If the routine address is zero, no routine will be executed. The method of setting up Meta-Control characters is probably still confusing to the reader. Try reading the descriptions of the following traps. The mechanism is actually very simple....

MCINIT - TRAP 62

USE	Setting up a Meta-Control character.
PARAMETERS	R0 - The character (in Graphics format) that will cause the Meta-Control action. E.g. if R0 is 1, then this will set up Meta-Control-A. Note that all of the upper case ASCII letters correspond to the Graphics format letters (by coincidence).
	R1 - The address of the Meta table (whose format was shown above). Note that at the end of the Meta table is a zero followed by the address of the Meta-Control routine (zero if none). This routine will be called every time the Meta-Control character is struck. Note also that if the end-of-table indicator is -1 instead of 0, the monitor's Meta table will be cleared before this one is read in.
ACTION	The table address in R1 is put into the the appropriate Meta-Control table location in the monitor (as determined by R0).
NOTE	If R1 contains zero, then no action will take place when the particular Meta-Control character is struck and no Meta characters will be set up.

MCLR - TRAP 24

USE	Turns off all Meta characters except, of course, for the special Meta characters.
ACTION	This routine clears the monitor's Meta Table of all values. Thus, Meta characters will cause no actions.

Meta character processing is simple. If the monitor's Meta table location corresponding to the character is zero, the character is ignored. Otherwise the value is taken to be an address of a routine to call. When processing Meta and Meta-Control characters, all registers will be available for use by the called routine. In particular the registers will contain the following information:

R0 The original input character in Graphics keyboard format.
R1 The decimal keyboard number.
R2 The octal keyboard number.

Meta-Control character processing is similar, except that before the routine is JSR-ed to, the user's Meta table is copied into the monitor's Meta table. This copying is the only way in which addresses can be placed into the Meta table.

As a final clarification, the following is an example of setting up a Meta-Control character and some Meta characters. Suppose we wish to set up 3 Meta characters: A, B, and C. Pretend that these routines are part of a help system so that they are set up with a Meta-Control-H. Suppose also that we don't want the user to have to worry about the Meta-Control-H or even the Meta part of the characters. We therefore want to "type" the Meta-Control-H ourselves and we want to treat all input characters as if the Meta key had been held down. The user will merely type A, B, or C to call the routines AHELP, BHELP, and CHELP. The following code shows it all:

```
                    ! In BLISS-11:

REQUIRE SX.B11[A630GS00];

ROUTINE HELP = KEYLOK(#100000);     ! The Meta-Control routine

BIND METATABLE = PLIT("A", AHELP, "B", BHELP, "C", CHELP, 0, HELP);

ROUTINE START =
BEGIN
   ! Set up the Meta table
   MCINIT("H", METATABLE);

   ! Now fake the Meta-Control-H to set up the table
   SKIRTN(#100000 + #400 + "H");
END;
```

```
                    ; In Macro-11:

.REQUIRE SX.P11[A630GS00]

HELP:                                   ; This is the Meta-Control routine
        MOV     #100000, R0     ; The Meta bit
        KEYLOK
        RTS     PC

METABL: 'A                              ; This is the Meta table
        AHELP
        'B
        BHELP
        'C
        CHELP
        0
        HELP

START:  MOV     #METABL, R1     ; The Meta table address
        MOV     #'H, R0         ; The Meta-Control character
        MCINIT

        ; Now fake the Meta-Control-H

        MOV     #100410, R0     ; Meta-Control-H
        SKIRTN
        RTS     PC
```

When routine START is executed it does two things. First, it sets up the Meta-Control-H. This involves telling the monitor that when a Meta-Control-H is struck, the Meta table should be read in (thus enabling the Meta-A, Meta-B, and Meta-C) and the routine HELP should be executed. The second thing that START does is to actually type a Meta-Control-H. At this point, the Meta characters are set up and the routine HELP turns on the keyboard lock so that all future characters are interpreted as if the Meta key had been held down. Now, each time an A, B, or C is typed, it is read as a Meta-A, Meta-B, or Meta-C and the appropriate routine is executed.

IV. Compiling, Linking, Running and Debugging

Assuming that you have created a source file, you must compile it with the COMPILE command. BLISS-11 files (with the extension "B11") yield a Macro-11 source file (with the extension "P11") and this must be compiled again to get an object file (with the extension "OBJ"). SAIL programs compile into object files (with the extension "REL").

Linking OBJ files is done with LINK11. You must type "R LINK11" and specify the OBJ files to be linked and the LNK file to be produced as follows:
<LNK file> ← <OBJ file>, <OBJ file>, ..., <OBJ file> /M

The /M switch specifies that the program is for a GDP. Other switches include /T to have a map printed on your terminal, /D to include DDT (see below), and /G to load the entire program into graphics space instead of program space. See SYS:LINK11.DOC for more detail. Note that if you use the LOAD command instead of COMPILE, LINK11 will be run for you.

Linking REL files is done with the LOAD command. The resulting core image can be stored in a SAV file with the SAVE command.

Running LNK files is done by various means, the easiest being the INITIA command. Type "INITIA @<LNK file name>". This will ship the program to the PDP-11 and start it running. Note that the GLSTER system has facilities that allow SAIL programs load PDP-11 programs implicitly.

Running SAIL programs can be done with the EXECUTE command for SAI and REL files. Use the RUN command for SAV files.

Debugging PDP-11 programs is best done with DDT. To get it, append the /D switch to your LINK11 command string. For best results, append the clause "(EN:ISD)" to your COMPILE command. This will cause the internal symbol dictionary to be included so that DDT can refer to your program with these symbols. For more detail on DDT, see SYS:DDT11.DOC and the PDP-10 reference manual.

Debugging SAIL programs is done with the DEBUG command. Use DEBUG instead of COMPILE, LOAD, or EXECUTE.

APPENDIX A

Graphics Keyboard Character Format

The character coming off the keyboard buffer (at 165202) has the following format. Note that bits 9 and 10 (A and I) are not set by the hardware but by various routines to add meaning to the character.

Bit	Octal	Meaning
M	100000	On if Meta key is depressed.
I	2000	Service this character immediately at interrupt time (this is not set by hardware)
A	1000	This bit is not set by the hardware but when the routine SKIRTN interprets a keyboard character this bit will (if set) stop keyboard translation to ASCII.
C	400	On if Control key is depressed.
T	200	On if Top key is depressed.
S	100	On if Shift key is depressed.
Key		The particular key struck (see Appendix B for character values).

APPENDIX B

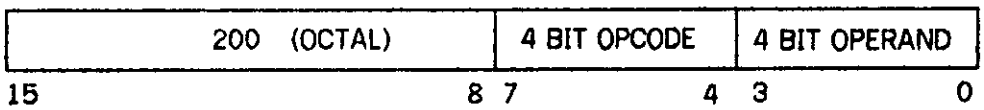
Graphics Keyboard Character Values

Octal	Key	Octal	Key	Octal	Key
1	A	26	V	53	+
2	B	27	W	54	,
3	C	30	X	55	-
4	D	31	Y	56	.
5	E	32	Z	57	/
6	F	33	RETURN	60	0
7	G	34	\	61	1
10	H	35	LINE	62	2
11	I	36		63	3
12	J	37		64	4
13	K	40	Space	65	5
14	L	41	BREAK	66	6
15	M	42	ESC	67	7
16	N	43	CALL	70	8
17	O	44	CLEAR	71	9
20	P	45	TAB	72	:
21	Q	46	FORM	73	;
22	R	47	VT	74	BS
23	S	50	(	75	ALT
24	T	51	)	76	
25	U	52	*	77	

Bit 6:	On (100) if Shift	Off if no Shift
Bit 7:	On (200) if Top	Off if no Top
Bit 8:	On (400) if Control	Off if no Control
Bit 15:	On (100000) if Meta	Off if no Meta

APPENDIX C

Control Words



OPCODE	NAME	FUNCTION
0	TERM	This will signal the end of a display list and the GDP will process the next instruction. A TERM in an instruction or character list will halt the graphics.
1	INTR	Interrupt the PDP-11 at vector 104. See subsection on Graphics Interrupts.
2	LCMD	Load character mode. Operand 0 means draw vectors, 1 means draw characters.
3	LFMT	Load vector format. Operand 0 means SHORT, 1 means MEDIUM, and 2 means LONG vectors.
4	LILA	Load intensity level absolute. Operand may be 0 through 17 (octal) with 17 the highest intensity. Normal intensity is 17.
5	LILR	Load intensity level relative. Operand is taken to be a two's complement integer and is added to the current intensity.
6	LSCA	Load scale absolute. The scale will multiply the vectors following it by these values:

Operand	Scale	Operand	Scale
17	3-1/2	7	7/8
16	3	6	3/4
15	2-1/2	5	5/8
14	2	4	1/2
13	1-3/4	3	7/16
12	1-1/2	2	3/8
11	1-1/4	1	5/16
10	1 (normal)	0	1/4

7	LSCR	Load scale relative. The operand is taken to be a two's complement integer and is added to the current scale.

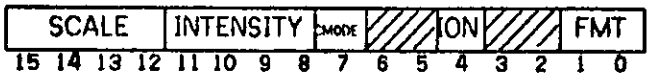
OPCODE NAME FUNCTION

10 SPL1 Special Codes 1. Operand interpretation is shown below. The last 4 operands below will be cancelled upon the occurrence of any control word, and thus are assured of never crossing boundaries between display lists.

Operand	Name	Function
0	TERM1	Same as TERM
1	ION	Turn intensity on
2	IOFF	Turn intensity off
3	ICOM	Complement the current intensity state
4	IOF1	Intensity off for next vector
5	IOF2	Intensity off for next 2 vectors
6	IOF3	Intensity off for next 3 vectors
7	IALT	Alternate intensity (starts as off)

11 SPL2 Special codes 2. The operand is interpreted as follows:

Operand	Name	Function
0	SETX	The following word is taken as the new X value
1	SETY	The following word is taken as the new Y value.
2	SETXY	The following two words are taken as the new X and new Y values respectively (first X then Y).
3	SETS	Set state in one fell swoop. The following word is interpreted as follows:

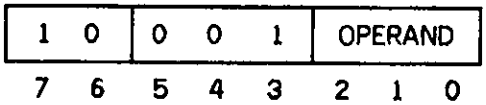


BITS	FIELD
0-1	Format
4	1=ION 0=IOFF
7	1=Characters, 0=Vectors
11-8	Intensity level
15-12	Scale

12 - 16 Unimplemented

17 GNOP No operation.

Halfword control words are interpreted only in SHORT format vector lists. The operand has the same effect as the SPL1 operands. The format is:



APPENDIX D

GDP Instructions

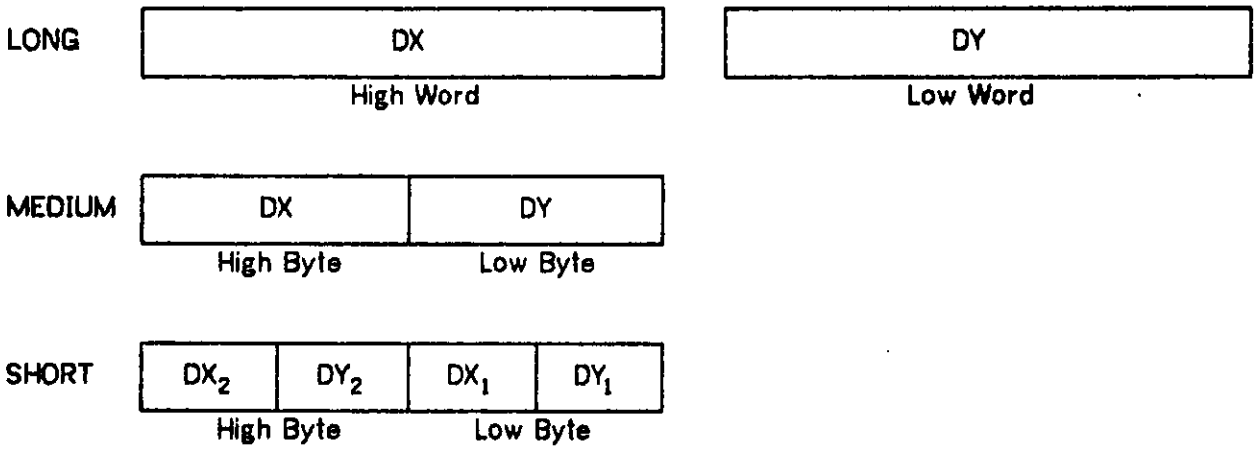
The GDP instructions are interpreted as shown below. Note that the operand address always points to a word boundary since the low bit is used as part of the op-code. Note also that the high bit is also used as part of the op-code; this limits the GDP to a 16K address space.

OPC1	OPC2	NAME	FUNCTION
0	0	JMP	Jump unconditionaly
0	1	INTR	Interrupt
1	0	JMS	Jump to subroutine
1	1	XQT	Execute

APPENDIX E

Vector Formats

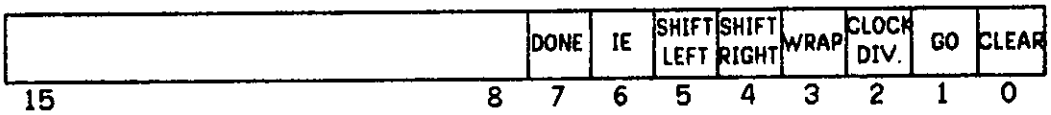
The three vector formats are shown below. Note that the Y component can always be thought of as preceding the X component both in word addresses and bit addresses.



APPENDIX F

Control Status Register

The layout of the Graphics Control Status Register is shown below. Note that the monitor takes care of the Clear, Go, Interrupt Enable, and Done bits. The User's only concern is the Clock Divide, Wrap Around, Shift Right, and Shift Left bits.



Bit	Function
0	Clear
1	Go
2	Clock Divide
3	Wrap Around
4	Shift right
5	Shift left
6	Interrupt enable
7	Done

APPENDIX G

ASCII Character Values

Code	Character	Code	Character	Code	Character
0	none	53	+	126	V
1	↓	54	,	127	W
2	∝	55	-	130	X
3	β	56	.	131	Y
4	∧	57	/	132	Z
5	∘	60	0	133	[
6	∈	61	1	134	\
7	∩	62	2	135	]
10	λ	63	3	136	↑
11	tab	64	4	137	←
12	line feed (1)	65	5	140	'
13	vert tab (1)	66	6	141	a
14	form feed (1)	67	7	142	b
15	car. ret (1)	70	8	143	c
16	∞	71	9	144	d
17	∂	72	:	145	e
20	⊂	73	;	146	f
21	⊃	74	<	147	g
22	∩	75	=	150	h
23	∪	76	>	151	i
24	∀	77	?	152	j
25	∃	100	@	153	k
26	⊗	101	A	154	l
27	↔	102	B	155	m
30	_	103	C	156	n
31	→	104	D	157	o
32	~	105	E	160	p
33	≠	106	F	161	q
34	≤	107	G	162	r
35	≥	110	H	163	s
36	▪	111	I	164	t
37	∨	112	J	165	u
40	space	113	K	166	v
41	!	114	L	167	w
42	"	115	M	170	x
43	#	116	N	171	y
44	$	117	O	172	z
45	%	120	P	173	{
46	&	121	Q	174	|
47	'	122	R	175	}
50	(	123	S	176	≠ (2)
51	)	124	T	177	back space (1)
52	*	125	U		

1. These characters are automatically treated as control characters. KEYASC will set the high bit for these characters.
2. This character is a duplicate of character 33 for SOS compatibility.

APPENDIX H

Monitor Traps

The following table shows the monitor traps in the same order that they were presented in the manual. Each trap is quickly described. Note that the mnemonics for the traps can be found in the file SX.P11[A630GS00], SX.SAI[A630GS00] or SX.B11[A630GS00]. so that with this file in your compilation, the mnemonic name may be used instead of the word TRAP and the trap number.

No.	Name	Page	Description
TRAP AND EMT TRAPS			
0	TRPADD	15	Returns the address of the trap table
34	EMTSET	15	Routine R0 becomes EMT number R1
141	EMTSZ	16	EMT table becomes R0 words long
ACTIVITY TRAPS			
65	ACTVAT	17	Routine R0 entered into activity list
145	ACTRTN	17	Soft restart
SPACE ALLOCATORS			
1	GETSPC	19	R0 becomes address of program buffer R0 words long
36	GTGRSP	19	R0 becomes address of graphics buffer R0 words long
2	BIGSPC	19	R0 becomes address of largest program buffer, R1 is size in bytes
37	BGGRSP	19	R0 becomes address of largest graphics buffer, R1 is size in bytes
3	GIVSPC	19	Buffer at R0 freed
4	SOMSPC	19	Buffer at R1 freed after R0
UTILITIES			
6	SAVE1	20	R1 through R5 saved on stack
14	RETUR1	20	R1 through R5 restored from stack
51	MULT	20	R0 multiplied by R1 yields R0 (high) and R1 (low)
52	DIVIDE	20	R0 (high), R1 (low) divided by R2 yields R0 remainder R1
5	WHAT	20	Capability information placed in R0
CLOCK TRAPS			
46	LINTIM	22	Line time returned in R0
47	LINPUT	22	Routine R0 put in clock queue
50	LINCLR	22	Routine R0 removed from clock queue

No.	Name	Page	Description
OUTPUT TRAPS			
161	OUTSET	23	Reset output. Set buffer to R0 if R0 not -1
43	ASLI	23	ASLI at R0 set to input speed R1, output speed R2
31	SLPONE	24	ASCII character in R0 sent to PDP-10
35	CONOUT	24	Control character in R0 sent to PDP-10
32	SLPRTN	24	Character in R0 sent to PDP-10 using CONOUT or SLPONE
45	STRSLP	24	String at R0 sent to PDP-10
163	EFCS	24	If R0=0, don't send escape characters to PDP-10
33	IMGRTN	26	Message at R1 (size in high byte of R0) sent to PDP-10
INPUT TRAPS			
162	TENSET	27	Input from PDP-10 reset
42	PARITY	27	If R0 is non-zero, full 8-bits from PDP-10 accepted
41	LNKTAK	27	Use routine R0 for input (normal routine if R0=0)
7	LOADR	30	Accepts program load, remembers address
10	LOADA	30	Load absolute data at R0 (for R1)
54	RELOCA	30	Program relocated by bits at R0 (for R1)
11	LOADTR	31	Transfer to R0, flags in R1 determine relocation, etc.
EDITOR TRAPS			
146	BRKSET	32	Break table set from 32 bits in R0 and R1
40	ETASET	32	Routine for edit buffer processing in R0 (normal routine if R0=0)
CHARACTER SET TRAPS			
153	CHRSET	35	Character R1 can be found at routine R0
156	CHRLOD	35	Complete re-do of character set (new set at R0, table at R1)
SCROLLER TRAPS			
120	SCRRTN	37	Character in R0 put in scroller
44	STRSCR	37	String at R0 put in scroller
134	CLEAR	37	Clear all characters from scroller
104	FORM	37	Form feed scroller
111	SCRNEW	38	Set scroller parameters to default state
112	XCHN	38	Set X position of scroller to R0
113	YCHN	38	Set Y position of scroller to R0
115	SCLCHN	38	Set scale of scroller to R0
114	INTCHN	38	Set intensity of scroller to R0
116	LINCHN	38	Set number of scroller lines to R0
117	CHRCHN	38	Set number of characters per scroller line to R0
101	UNICHN	38	Set number of tab units per scroller line to R0
16	JMPCHN	38	Set scroller jump parameter to R0
30	SCRDEF	38	8 word buffer at R0 sets current parameters
20	PAGSET	39	R0 alternate scrollers are allocated
17	PAGGET	39	Scroller R0 is selected and displayed
77	PAGOFF	39	Scroller R0 is no longer displayed

No.	Name	Page	Description
DISPLAY TRAPS			
71	SCRCLR	40	Turn off scroller position
73	SCRDIS	40	Put R0 in scroller position
107	SCRUNB	40	Return scroller to normal state (displayed)
72	USECLR	40	Turn off user position
144	USERET	40	Turn user position back on from last USECLR
74	USEDIS	41	Put R0 in user position
75	EDCLR	41	Turn off editor position
76	EDDIS	41	Put R0 in editor position
27	GRAADD	41	Return graphics display address in R0
100	GRASTP	42	Hard stop of graphics
132	DISPIT	42	Refresh graphics R0 times
130	HALDIS	42	Soft stop of graphics
131	GODISP	42	Restart of graphics
GRAPHICS INTERRUPTS			
15	INTRST	43	Graphics interrupt R0 will be serviced by R1
GRAPHICS REGISTERS			
67	GETREG	44	Get graphics registers determined by R0 into buffer at R1
66	LODREG	44	Set graphics registers determined by R0 from buffer at R1
12	LODCSR	44	Load graphics CSR from R0
53	GETCSR	44	Get graphics CSR in R0
KEYBOARD TRAPS			
64	KEYASC	45	Decode Graphics character R0 to ASCII character R0
26	KEYLOK	45	Logically OR all future characters with R0
133	DDTCAL	45	Routine R0 is now DDT
70	SKISET	46	Set handler for all characters to routine in R0
63	SKINWR	46	Initialize keyboard without space-war mode
23	SKIWAR	46	Initialize keyboard with space-war mode
55	SKIRTN	46	Interpret the character in R0 (from keyboard)
126	SKISOM	46	Interpret R1 characters at R0 with SKIRTN
60	SKINOR	47	Set keyboard handler to normal non-local routine
22	SKILOC	47	Put keyboard in local mode
56	SKIGET	47	Return address of keyboard handler in R0
57	SKIPUT	47	Set keyboard handler to routine R0
62	MCINIT	48	Character R0 has Meta table at R1
24	MCLR	49	Clear the Meta table

APPENDIX I

Sample Programs

The following programs are given as examples. They are written in Macro-11, BLISS-11, SAIL, and BLISS-10 respectively. Each does very little: it draws a triangle on the screen just to the right of center.

```
        .TITLE  TRIANG          ; Macro-11
        .CSECT  TRIANG

 .REQUIRE SX.P11[A630CS00]      ; Monitor trap defines

 .REQUIRE GRADEF.P11[A630CS00]  ; Graphics control word defines


PIXSTR: 0                       ; This will hold the display list address

START:  MOV     #20.,R0         ; I want a 20 word buffer
        GTGRSP                  ; Get from Graphics Space.  On return, R0 will
                                ; hold the buffer's address.
        MOV     R0,PIXSTR       ; Store the address
        MOV     #TRIVEC,R1      ; Address of vectors to be loaded into display list
LOAD:   MOV     (R1),(R0)+      ; Move a word at a time into display list
        CMP     (R1)+,#TERM     ; Was it the last word? (a TERMINATE)
        BNE     LOAD            ; Loop back if not and load next word
        MOV     PIXSTR,R0       ; Retrieve the display list address
        BIS     #GXQT,R0        ; Make it an XQT instruction
        USEDIS                  ; Put into user location
        RTS     PC              ; Return.  Let monitor do it's thing

; Here are the vectors and control words describing the triangle:
TRIVEC: LCMD    0               ; Load C-MODE with 0.  Vector mode
        LONG                    ; Long vector format
        ION                     ; We want visible vectors
        LILA    17              ; Maximum intensity
        LSCA    10              ; Normal scale
        SETXY   0,0             ; Set point to (0,0)
                                ; Now the triangle's vectors:
        LVEC    100.,0
        LVEC    -50.,60.
        LVEC    -50.,-60.
        TERM                    ; TERMINATE

        .END    START
```

```
MODULE TRIANGLE (START=DRAW) =          ! BLISS-11
BEGIN

    REQUIRE SX.B11[A630GS00];           ! Monitor instructions
    REQUIRE GRADEF.B11[A630GS00];       ! Control words

    BIND TRIVEC=PLIT(LCMD(0),           ! The vector list describing the triangle
                LONG,
                ION,
                LILA(15),
                LSCA(8),
                SETXY(0,0),
                LVEC(100,0),
                LVEC(-50,60),
                LVEC(-50,-60),
                TERM);

    ROUTINE DRAW =
    BEGIN
        LOCAL PIXSTR;
        PIXSTR=GTGRSP(20);      ! Get 20 words of Graphics Space and save it's address
        INCR I TO .TRIVEC[-1]-1 DO (.PIXSTR)[.I]=.TRIVEC[.I];   ! Load pixel
        USEDIS(GXQT(.PIXSTR));  ! Set opcode for XQT and put into user display location
    END;

END ELUDOM


BEGIN "TRIANGLE"                        Comment SAIL;

    REQUIRE "BAYSAI.SAI[A710SA00]" SOURCE!FILE;
    SOURCE!(ATYTLK.DCL[A630GS00]);
    SOURCE!(TERP10.DCL[A630GS00]);
    SOURCE!(TABL10.DCL[A630GS00]);
    SOURCE!(GRADEF.SAI[A630GS00]);      ! Convenient defines;

    TERINI("ATY",NULL);                 ! Initialize the "ATY" line;
    PIXCRE(20);                         ! Create a pixel of size 20 words;

    FOR DUM=LCMD(0),
        LONG,
        ION,
        LILA(15),
        LSCA(8),
        SETXY,0,0,
        LVEC(100, 0),
        LVEC(-50, 60),
        LVEC(-53, -60),
        TERM    DO PIXONE(DUM);         ! Load pixel;

    PIXDIS;                             ! Display pixel;
    GRAFOR;                             ! Force out buffer onto PDP-11;
    TERREL;                             ! Release ATY line, we're all done;

END "TRIANGLE";
```

```
MODULE TRIANGLE(STACK)=                          ! BLISS-10
BEGIN
    REQUIRE CLSTER.BLI[A630CS00];
    REQUIRE GRAPEF.BLI[A630CS00];

    BIND TRIVEC = PLIT(LCMD,
                  LONG,
                  ION,
                  LILA(15),
                  LSCA(8),
                  SETXY(0,0),
                  LVEC(100, 0),
                  LVEC(-50, 60),
                  LVEC(-50, -60),
                  TERM);

    TERINI(SIXBIT 'ATY', 0);
    PIXCRE(40);
    INCR I FROM 0 DO
    BEGIN
        IF .TRIVEC[.I] EQL TERM THEN EXITLOOP;
        PIXONE(.TRIVEC[.I])
    END;
    PIXDIS;
    GRAFOR();
    TERREL()

END ELUDOM
```

APPENDIX J

References

Questions, suggestions, and problems should be brought to:

Steven Rubin [A640SR12]	For software
Brian Rosen [A210BR11]	For hardware

Manuals about related systems:

GDP2.XGO[A630GS00]/B	Hardware documentation *
GLSTER.XGO[A630GS00]/B	Display package
GPI.XGO[A640SH01]/A	Convenient display package
SYS:LINK11.DOC	To link programs
SYS:DDT11.DOC	To debug programs
MONLOD.XGO[A630GS00]/B	Systems support
TABLET.XGO[A630GS00]	Tablet support
SILOS.XGO[A630KS00]/B	Character set editor
GDP.DEM[A630GS00]	Demonstration pointers
BSG.DOC[A630GS00]	SAIL-like runtime routines

Software aids:

SX.P11[A630GS00]	Macro-11 Trap defines
SX.B11[A630GS00]	BLISS-11 Trap defines
SX.SAI[A630GS00]	SAIL Trap defines
SX.BLI[A630GS00]	BLISS-10 Trap defines
GRADEF.P11[A630GS00]	Macro-11 Graphics defines
GRADEF.B11[A630GS00]	BLISS-11 Graphics defines
GRADEF.SAI[A630GS00]	SAIL Graphics defines
GRADEF.BLI[A630GS00]	BLISS-10 Graphics defines

* Available in document room as "GDP Programmers Guide"

INDEX