GS--GRAPHICS SYSTEM

Lee Copeland, et al

Utah University

Prepared for:

Advanced Research Projects Agency

15 November 1967

Technical Report 4-1

Lee Copeland
C. Stephen Carr

GS - GRAPHICS SYSTEM

November 15, 1967

COMPUTER SCIENCE

Information Processing Systems

University of Utah

Salt Lake City, Utah

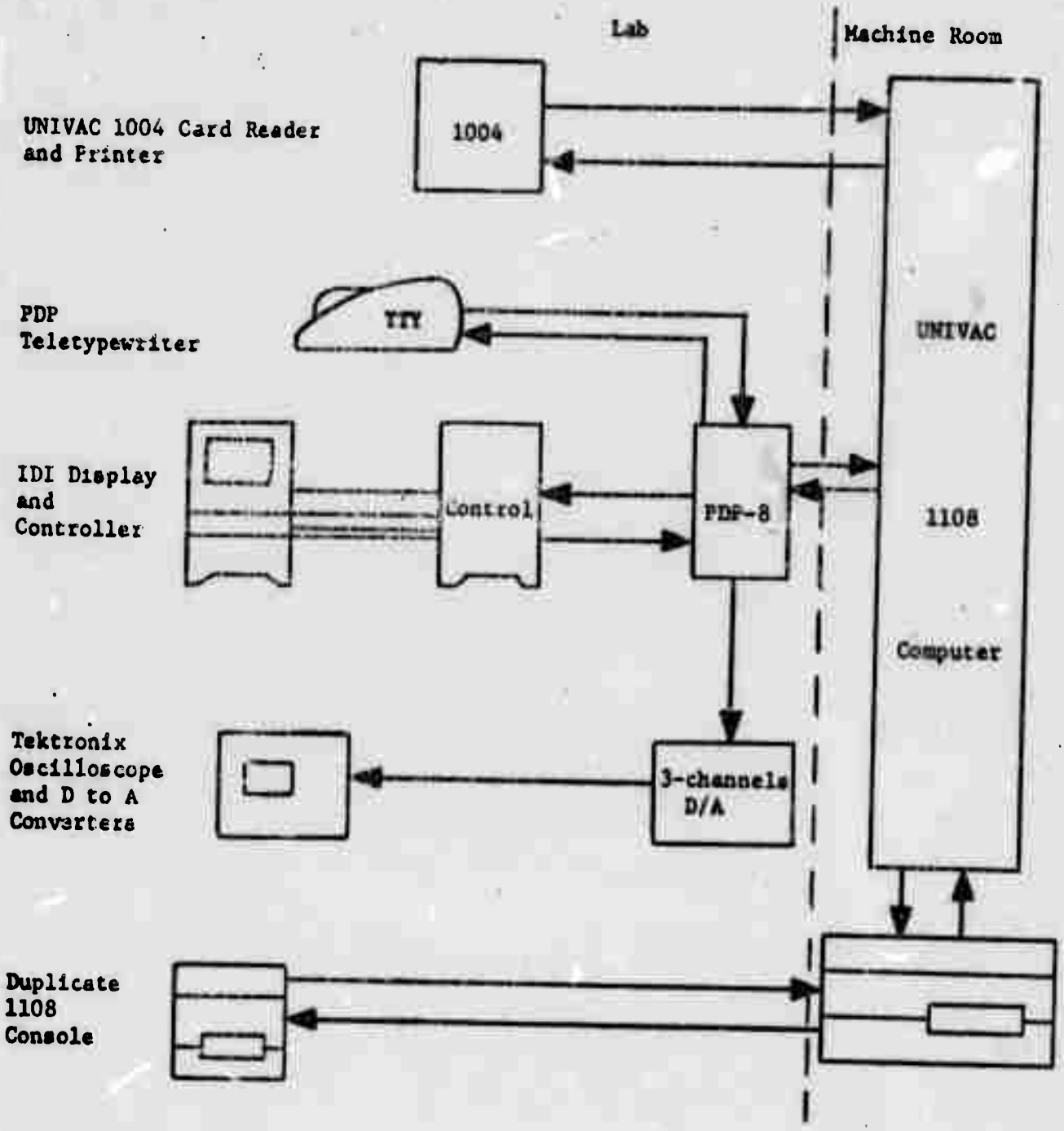
Advanced Research Projects Agency · Department of Defense · ARPA order 829

Program code number 6D30

This reference manual is intended for programmers working on the ARPA Project at the University who are familiar with the 1108, Exec II, and Fortran or Algol. This is not a training manual for the uninitiated. However, suggestions on ways to improve the system and the manual are welcome.

TABLE OF CONTENTS

GRAPHICS LABORATORY



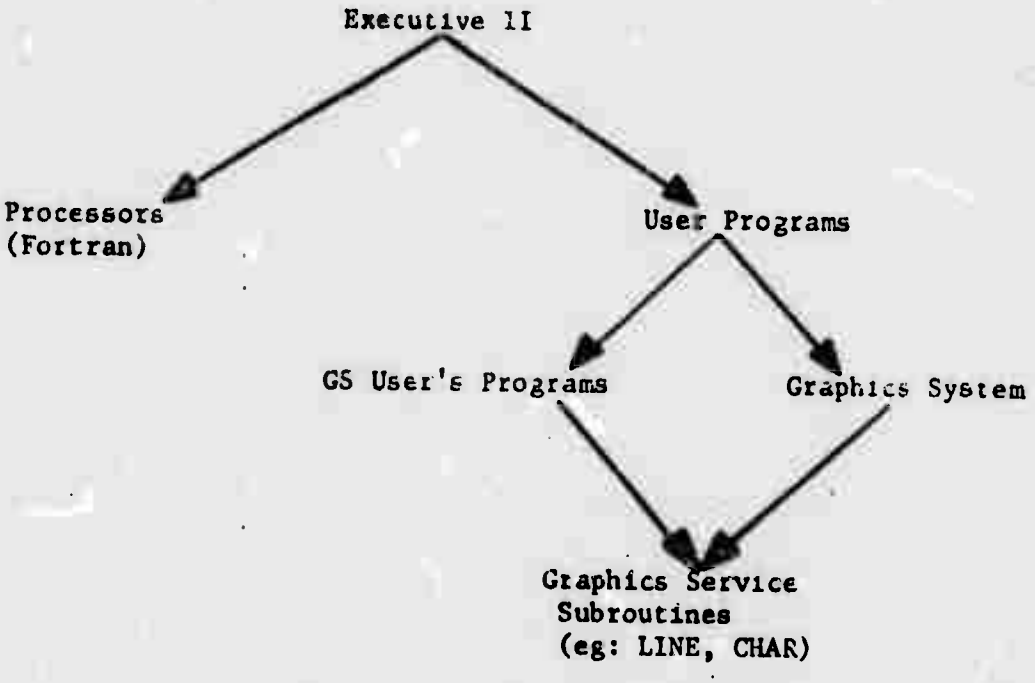
Additional devices, projected for the future, are a Sylvania or Rand tablet, graphical plotter, and half-tone display.

GS - GRAPHICS SYSTEM

The hardware shown on the preceding page has considerable potential when coupled with appropriate software. Of course, each user could write his own programs from scratch and he would certainly learn from the experience. However, he would be sidetracked from attacking his primary goals. Furthermore, he would continually need to modify existing programs to keep pace with changes (even minor ones) made to the equipment in the laboratory.

The Graphics System makes the display hardware easier to use as the programmer works exclusively within the 1108 computer and uses the standard systems program in the 1108 and the PDP-8. A relatively stable programming interface is maintained as equipment is added or modified.

GS is a submonitor under the 1108 Executive.



Programs and ideas within the Graphics System are due to a number of people associated with the University Computer Center and the ARPA Project.

Richard Blackburn, Bill Boam, John Warnock	Computer Link Programming Swapping System
Charles Brauer	GPM, General Purpose Macro Generator
Steve Carr	DEBUG, Conversational Debugger SPEED, Speedy Editor UNIDEC Assembler
Lee Copeland	PDP and Graphics System Programming
Ed Dallin	CalComp Simulator Routines
Gus Eastman, Bob Whited	Debugging Techniques
Alan Kay, Dave Walton	Simula/Algol
Devon Mecham	TRAC, Text Reckoning and Compiling
Dave Walton	1108 Assembler

I. On-Line Operation

The current 1108 Graphics System resides on the public, read-only Fastrand File, $$GS$$. The Graphics System is loaded into core with

```
∇ ASG    A=$$GS$$
∇ XQT    CUR
  IN A
```

The PDP-8-GS tape should be loaded into the PDP-8 and started at location 6000 octal before the 1108 program begins to execute within the Graphics System.

GS service routines are available to user programs; however, GS can act as the main program. To begin on-line operation, use the "GS" entry

point.

V XQT GS

Depress the reset button on the IDI display and an initial set of options should appear. The currently available options are:

* UNIDEC ASSEMBLER
* 1108 DEBUG
* SPEED TEXT EDITOR
* TRAC
* GPM
* CONTINUE
* LOGOUT
* SWAP

Other options in the planning stages are:

* 1108 ASSEMBLER
* PDP DDI
* FILE PROCESSOR
* ALGOL/SIMULA
* LOAD
* EXECUTE
* DISPLAY DEMO

Point the light pen at the star corresponding to the desired function and squeeze the light pen switch. Control is transferred to the specified routine and a new display appears.

Within a processor (text editor, for example), a single option is displayed at the bottom of the picture.

* RUBOUT

The rubout function allows the user to return to the initial options of the Graphics System at any time. Rubout is not equivalent to the batch

processing concept of abort as the LOGOUT option serves this need. Rather, rubout causes the current activity to be suspended (frozen) and control returns to the Graphics System. Of course, the suspended process may be resumed at will with

* CONTINUE.

In addition to the light button, RUBOUT, the rubout key on the teletype-writer performs an identical function.

II. Graphics System Services

1. General Information Services:

User programs may use GS subroutines without starting at "GS" as described in Section I. The services provided include:

PDP teletype processing

Display file processing

Light button declaration and monitoring

Pen tracking and pointing

Swapping

File manipulation

Programming Languages:

Throughout this manual, calling sequences are shown for Fortran programs. It is not intended that user programs be restricted to a single language, however. Indeed, languages more powerful and general than Fortran are being designed by several potential users of the Graphics System.

GS subroutine calls from Fortran programs have the form:

```
CALL  SUB(ARG1,..., ARGN)
```

The equivalent assembly language code is:

```
LMJ   B11,SUB
+     ARG1    . address of 1st argument
.      .
.      .
.      .
+     ARGN    . address of Nth argument
NOP
```

The Fortran programmer locates an absolute machine address with:

```
CALL LOC(A,B)
```

which stores the 1108 octal address of the variable or statement label A into location B.

The Algol programmer declares each GS entry point (except that for IDLE) to be

EXTERNAL FORTRAN PROCEDURE SNDFLE, LINE, . . . etc. . . ;

IDLE is a recursive procedure and should be declared as

EXTERNAL PROCEDURE IDLE.

All variables are of INTEGER type.

In this manual, the user is the man interacting at a display while the programmer is the man whose program calls on the Graphics System.

Character Codes:

Nine bit, ASCII characters are used throughout as six bit characters are simply inadequate for several projected uses of the system. Furthermore, the display character generator and teletypewriter use ASCII or a close variant of it. Only the seven right hand bits are interpreted by the Graphics System leaving the two right hand bits free for flags, etc. Since the requirements of some users are adequately served by 6-bit, Fieldata code, automatic conversion is available.

CALL CVION

invokes conversion between Fieldata and ASCII on all subsequent character transmissions between GS and the programmer.

CALL CVIOFF

restores transmissions to pure ASCII. Conversion is on initially.

Character string conventions are:

Code	Characters/word	Bits/character	
Fieldata	6	6	left justified
ASCII	4	9	" "

Fieldata

G	R	A	P	H	I
C	S				

ASCII

G	r	a	p
h	i	c	s

Single characters are simply one character strings and are consequently left justified.

H	
h	

The ASCII end-of-text character, "D^C", serves as an end-of-text character throughout the system. The equivalent Fieldata code is "Δ".

Pen:

Computer input via pointing is possible with either the light pen or Sylvania Tablet stylus. The particular pointing device in use is declared with:

```
CALL PEN(N)
```

where N=0 light pen (normal)

N=1 Sylvania Tablet stylus

The status of the pen switch is returned by GS routines relating to the stylus. In addition, an interrupt (discussed in Section II below) exists which occurs whenever the switch status is changed.

CALL PENSW(INTERRUPT SUB., SW)

where: INTERRUPT SUB = User's subroutine for processing the pen switch.

SW = Variable where pen status is to be stored by GS.

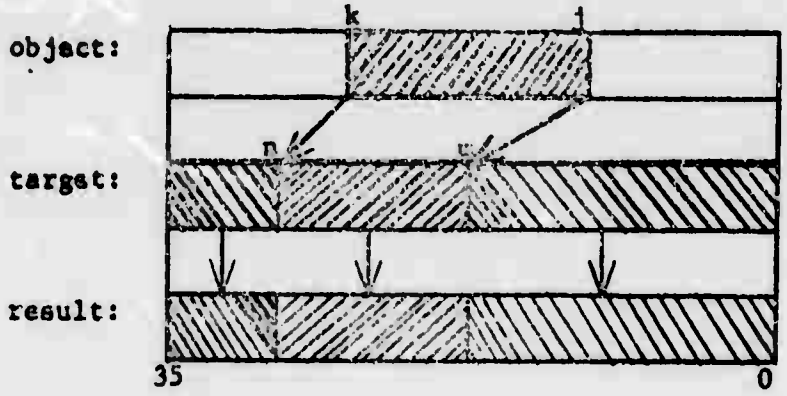
SW = 0 = Switch up.

= 1 = Switch down.

Miscellaneous:

Insert:

Bit manipulation is carried out via the INSERT function. Three machine words are involved: an object, a target, and a result.



result=INSERT(Target, n, m, Object, k, j)

Bits j through k inclusive are extracted from the object word. The resulting field is then trimmed on the left to fit into the target between bits m through n inclusive. The object and target words themselves are not disturbed as the result of the operation is the value of the function.

The sequence to unpack a word of 6-bit characters is:

```
      DO 100 I=1, 6
      J = 30 - (I-1)*6
100   K(I) = INSERI(6H    , 35, 30, WORD, J+5, J)
```

The current contents of the A, B, and R registers is printed on the 1004 for debugging purposes with CALL DUMP.

The Graphics System programs in the PDP-8 and 1108 computers are initialized with CALL RELOAD. A sign on message including the date and time is typed on the PDP teletypewriter.

CALL RUBOUT (INTERRUPT SUB)

An interrupt processing subroutine (discussed below) is declared which is executed when the user types a rubout character or his program attempts an illegal operation which the 1108 hardware detects. If a user rubout processing subroutine has not been declared, the following happens: illegal operations cause the run to be aborted and the rubout character is treated as a normal character.

CALL KNOB (INTERRUPT SUB, VALUE)

An interrupt processing subroutine (discussed below) is declared which is executed when the user rotates the knob at the display. The new value of the knob setting is stored in VALUE.

2. Software Interrupts:

The programmer writes interrupt subroutines to process asynchronous events of interest to him. Subsequent user actions at the graphics terminal (eg: pen matching) cause the PDP-8 to signal the 1108 Graphics System program over the communications link. The Graphics System will

branch off to the subroutine declared by the programmer to process the condition

These are "software" interrupts in the sense that they are carried out by the GS program, not the 1108 interrupt hardware. The programmer should execute

```
CALL   IDLE
```

periodically to allow the necessary bookkeeping to occur. Typically, a conversational program takes the form

```
(declare interrupt
   subroutines)

      .
      .
      .

C     WAIT FOR USER ACTION
100   CALL IDLE
      GO TO 100
```

When IDLE is called, the interrupts are processed according to the following priority:

(high) rubout

pen tracking

light buttons

pen match

knob

TTY input character count exceeded

(low) TTY output character count not exceeded

Interrupt declarations are discussed below in the context of specific GS functions. In each case, a subroutine is named which the programmer writes to process the interrupt. For example:

```
      CALL RUBOUT(ABORT)
                .
                .
                .
      SUBROUTINE ABORT
C     PROCESS USER RUBOUT BUTTON ACTION
                .
                .
                .
      RETURN
```

3. PDP Teletypewriter Handling:

Six routines apply to the PDP teletypewriter:

TYPCHR -- type a character

TYPLIN -- type a line

GETCHR -- get a character

GETLIN -- get a line

CHRINI -- character input interrupt

CHROUI -- character output interrupt

CALL TYPCHR (CHARACTER)

types one character on the PDP typewriter. The character should be left justified in the word.

CALL TYPLIN (LINE STARTING ADDRESS)

types one line of up to 127 characters on the PDP typewriter. The string should be packed, left justified and terminate with an end of text character.

CALL GETCHR (CHARACTER LOCATION)

gets one character from the PDP typewriter, storing it, left justified, in the specified location.

CALL GETLIN (LINE STARTING ADDRESS, MAXIMUM NUMBER OF CHAR., TERMINATING CHAR., ACTUAL NUMBER OF CHARACTERS)

gets one line from the teletypewriter. The operation terminates when either the requested number of characters or the terminating character is typed. This routine yields a left justified string. The end of text character and the terminating character are returned as part of the line. All characters that are actually typed are included in the character count.

CALL CHRINT (CHARACTER COUNT, INTERRUPT SUB)

an interrupt to the specified subroutine is generated when the input buffer contains at least the specified number of characters.

CALL CHROUT (CHARACTER COUNT, INTERRUPT SUB)

An interrupt to the specified subroutine is generated when the output buffer contains fewer than the specified number of characters.

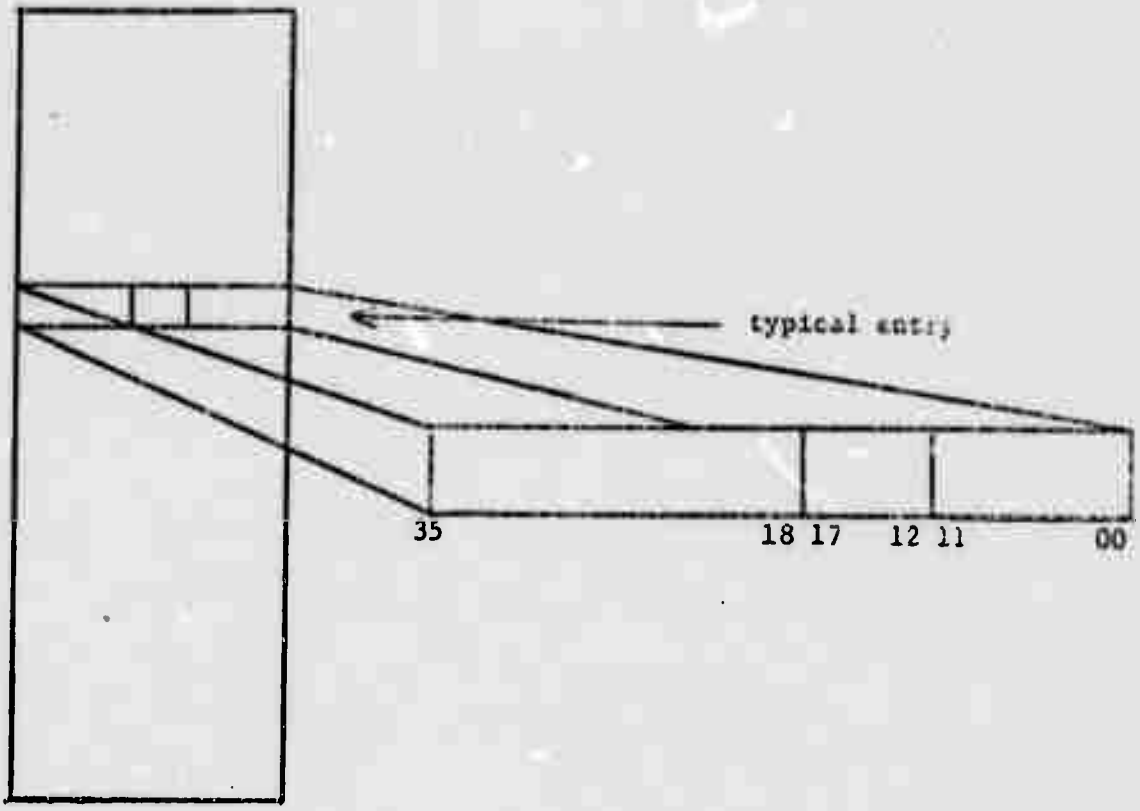
CHRINT and CHROUT allow input and output to occur concurrently with computation and swapping.

4. Display File Processing:

The display file is a vector of commands which direct the display controller to produce the desired picture. Since this is very hardware dependent, the Graphics System provides a number of routines which partially shield the user from the grubby particulars of each display device.

The display file is in the user's 1108 program to facilitate its manipulation and linkage to the data structure from which it was generated. Upon demand the Graphics System sends an image of the file to the appropriate display hardware.

The display file format is:



bits	
0-11	display byte
12-17	communication bits
18-35	user defined pointers and flags

Only the twelve, right-hand bits of each entry are actually transmitted to the display processors. These twelve bit bytes are the actual display controlling words defined in the Appendices. The eighteen, left-hand bits are exclusively for the programmer's use. This area is convenient for flags and back pointers into the data structure. The middle six bits are reserved for communications, as explained below.

The programmer can modify any bit in his display file but the Graphics System will only modify the right hand half of the file.

The six communication bits allow GS and the programmer to communicate information relating to individual display file entries.

The communications bits are interpreted as follows:

Bit	Meaning
12	1 = This is a display instruction word 0 = This is a display data word
13	
14	
15	
16	
17	

The Tektronic Oscilloscope is also available as explained in Appendix B. One program can switch between both scopes maintaining separate display files for each.

The display and associated display file to be used are declared before any parts of a picture are created.

```
INTEGER DF(1000)
CALL IDI (DF)
INDEX = 1
```

This indicates the location of the display file within the user's program and sets on index variable to one.

Usually the programmer does not wish to compose the absolute display file entries (bits 0-11) himself Therefore, a number of routines are provided to streamline this work Each routine requires an index number indicating where the entry is to be made within the file. Upon exit from the routines, the index is updated to point to the first cell after the entry. Therefore, entries may be arbitrarily positioned but if the index is untouched between calls, sequential entries result.

CALL LINE(FN, X,Y,INDEX)

where FN is the display file name, X,Y are the ending x,y coordinates of the line to be drawn, and INDEX is is the index of the last "line" command in FN.

CALL CHAR(FN, CHARACTER STRING, INDEX)

where FN is the file name. The character string is left justified with proper conversion in effect, and the last character in the string is the appropriate end-of-file mark. INDEX points to the next available display file word.

Character strings are created in different ways (eg.: CALL GETLIN) or by the subroutines ENCODE and DECODE. ENCODE takes user supplied variables and creates a character string according to the Format supplied by the user. DECODE performs the converse operation. Input/Output unit number 23 is associated with the encoding and decoding of strings.

```
      INTEGER IBUF(22)

      CALL ENCODE(IBUF)
      WRITE(23, 1), A, I, J, Q
    1 FORMAT(F10.3, 10X, 2I10, 5X, F7.6, 1H1 )
      CALL CHAR(DISPLAY FILE, IBUF, INDEX)
```

or

```
      CALL GETLIN(IBUF, etc.    )
      CALL DECODE(IBUF)
      READ(23, 2), I
    2 FORMAT(I10)
```

INDEX points to the next display file word.

```
      CALL  INCVEC(FN, ΔX,ΔY, REPEAT, INDEX)
```

where FN is the file name and INDEX is the index of the next available word in the display file.

In addition to these routines, other subroutines are provided to set the various modes of lines, vectors, and characters.

CALL MATCH(N, INTERRUPT SUB, INDEX, SW) N=0 Pen Match Off (normal)

N=1 Pen Match On

INTERRUPT SUB. = User's subroutine which processes pen switches.

INDEX = Address of matched word (set when match occurs).

SW = 1 if the pen switch was down at the time of the switch.

= 0 if the pen switch was up.

CALL BLINK(N)	N=0 Blink Off (normal)
	N=1 Blink On
CALL BLANK(N)	N=0 Blank Off (normal)
	N=1 Blank On
CALL INTEN(N)	N=0 Normal (normal)
	N=1 Dim
	N=2 Bright
	N=3 Off
CALL OFFSET(N)	N=0 Normal (normal)
	N=1 Small, superscript
	N=2 Small, subscript
	N=3 Small, no offset
CALL INTYPE(N)	N=0 Position
	N=1 Position, write dot
	N=2 Dash
	N=3 Solid (normal)
CALL MARGN(N)	N=0 Nop (normal)
	N=1 Set Margin
	N=2 Return to Margin
	N=3 Return to Margin and line feed

These modes remain in effect until declared differently. Alternately, one could imagine indicating the mode as an additional argument to calls on LINE, CHAR, etc. Which approach is more natural and efficient depends on each user's applications and both methods are equivalent; the former was chosen here.

A previously declared display file is sent from the 1108 to the PDP for display with

CALL SNDFLE(DISPLAY FILE, INDEX).

Many display files may be in existence within the 1108 but only one can be stored within the PDP-8 at one time. Words 1 to INDEX of the designated DISPLAY FILE are sent by SNDFLE.

5. Light Button Declarations and Monitoring:

Light buttons are simply sensitive spots on the display tube which the light pen or other pointing device can detect. Their inclusions in the Graphics System allows the programmer to easily provide for user input signals. This facility is used to implement the light buttons which appear when GS itself is entered (see Section I).

The programmer declares a light button with

CALL LITBTN(DISPLAY FILE, TEXT, INTERRUPT SUB, X, Y, INDEX).

A light button appears at X,Y on the screen when the DISPLAY FILE is sent to the PDP computer (ie: when SNDFLE is called). This light button is labeled with the TEXT string of characters.

When the stylus is placed over the light button, a transfer is made to the INTERRUPT SUBroutine. Of course, the transfer cannot actually occur until GS is given control by calling the IDLE subroutine.

6. Pen Tracking:

GS tracks the light pen or other stylus if requested to do so. Tracking should not be confused with the match operation described above under Display File Processing.

CALL TRACK(DISPLAY FILE, RADIUS, INTERRUPT SUB)

where: DISPLAY FILE is a file for which tracking should operate.

INTERRUPT SUBroutine is the routine to interrupt to when an X,Y coordinate is available. An interrupt routine need not be declared unless desired.

The next X,Y coordinate pair is read in with:

CALL GETXY(X, Y, SW)

where: SW=1 if the pen switch was down at the time

0 if the pen switch was up.

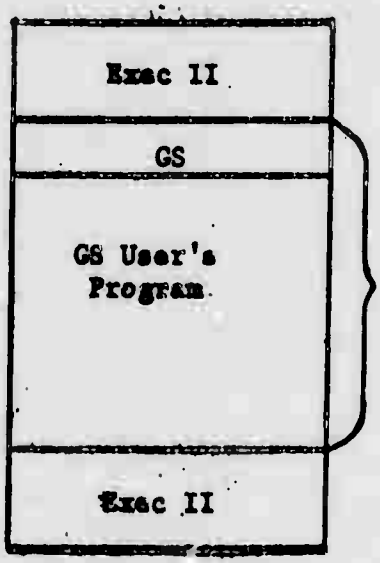
-1 if no further coordinates are available at the time.

Obviously, some data filtering is usually desirable. RADIUS is the distance from one returned point to the next. The quantity of data returned to the user depends on the radius used.

7. Swapping:

Eventually, Exec 8 will allow the 1108 computer to be time-shared among many users. During the interim before Exec 8 is ready for use, Exec II has been modified to allow the GS user to swap in and out of core with the normal batch processing stream of jobs. This temporary system is not a time-sharing system in the normal sense, and it is justified in economics only.

Many features are only available before swapping begins (eg: all compilers provided by Univac, for example). However, some limited functions can be performed in the swapping mode at a considerable reduction in operating costs.



Assume that GS and the user's program are in core. A call to SWAP writes this core image onto a reserved region of the fast drums (FH-432) and effectively ends the run. EXEC II goes on to the next job in the input stream. Its tables are initialized and it is no longer concerned with the graphics user.

However, user actions at the display cause the PDP-8 to interrupt and swap out the batch mode user and to restore the graphics user. Restricted service after swapping results, since Exec II is set up for a batch user. The graphics program can execute and access Fastrand files, however.

When a program is waiting for the on-line user to respond, the system routine IDLE should be called allowing GS to take appropriate action. At all times, a call to IDLE allows GS to service queued interrupt requests which result from earlier declarations and do other bookkeeping.

The graphics programmer swaps himself out with:

CALL SWAP(RETURN CONDITION)

where the return condition tells GS how to reactivate him.

Return condition: 0 = manual mode

1 = automatic mode

In automatic mode, the graphics program is returned to core whenever any GS interrupt is waiting for processing. In manual mode, the graphics program is not reactivated until a specific swap character is typed. This swap character is declared before swapping out with:

CALL SWPCHR(CHARACTER)

The up-arrow character is assumed initially.

8. File Manipulation:

Because of the impending conversion from Exec II to Exec 8, it is unclear whether it is worth the effort to provide file manipulation capabilities within the Exec II framework. It is clear, however, that file manipulation (declaration, access, deletion) by running user programs is highly desirable. Unfortunately, there is no provision for this in Exec II.

In addition, SPEED, the text editor, is available to create and modify symbolic files.

The preceding material describes the Graphics System as it stands on 15 October, 1967. However, evolutionary changes are envisioned. As the project acquires additional graphics hardware, convenient access to it will be provided by the expanded Graphics System program.

III. Appendices

1. Information Displays Inc. Display Codes
2. Tektronix Oscilloscope Display Codes
3. Alphanumeric Character Codes
4. PDP-8 Graphical Display System
5. 1108 - PDP-8 Interface
6. Examples

Appendix I

Information Displays Inc. (IDI) Display Codes

This unit displays points, strung characters in two sizes plus subscripting and superscripting and vectors in two-line structures (dash, or solid). All may be displayed in any of three brightnesses, or blinked as selectively controlled by the digital input. Programmable line margin control is included.

Upon receipt of a light pen signal, the display will halt, will leave the contents of the Program Address Counter on-line to the computer, and will send an interrupt signal to the computer. When the computer accepts the address, it will send a restart signal to the display, which will then continue with its normal display routine.

In order to provide a more uniform intensity with different size display messages, the display controller contains a Repetition Rate Control to prevent refresh rates greater than a preselected amount (typical 40 frames/second).

DISPLAY FILE

The display operates from instructions in the form of consecutive 12-bit bytes. Basic operations are to: plot points, write characters, and draw straight lines between specified locations. Symbols, lines and dots can be positioned with a resolution of 1024 locations (10 bits) in both X and Y.

Characters may be written in two sizes: centered, or as small subscript or superscript; in three intensities. Characters are normally written in a typewriter mode with program controlled margin set, carriage return, and line feed - carriage return.

Dots may be randomly positioned, or incremented, at increments of 0, $\pm 2^1$, or $\pm 2^2$ in either or both X and Y. Under program control, any set of increment values can be repeated for up to fifteen dots, without using additional bytes.

A vector may be drawn from a current location to the X and Y location specified in succeeding vector bytes with either of two line structures and three intensities.

Figure 1 shows the organization of the computer bytes. Three types of bytes are used: Program Control, Display Control, and Display Data.

Program Control bytes provide an unconditional jump instruction within the display file.

Control bytes generally establish the operation mode (such as dot increment, symbol, vector, and packed symbol), while the data bytes contain the specific graphic element information, such as position, or symbol.

A description of the byte organization follows.

Bit Ø is logical "1" for all control bytes and is logical "0" for all data bytes (except the packed symbol data byte).

Bit 1 is logical "1" for all Display Control bytes, bits 3-4 determine the mode.

Each Control byte is 12 bits. All Data bytes, except vector are 12 bits. Vector Data requires two 12 bit bytes.

In each Control byte, bit 5 is used to enable the light pen for succeeding Display Data bytes; bits 9-10 set intensity for succeeding data bytes; and bit 11 sets the Blink for succeeding Display bytes.

PROGRAM CONTROL

This byte is used with the Jump (JMP) instruction to provide essentially the same program control operations as in the PDP-5. Program Control bytes may be interspersed with Display Control and Display Data bytes without disturbing the display operation.

With bit 3 as logical "0", the address in bits 5-11 is the direct address (JMP) with bit 3 as logical "1", the location specified in bits 5-11 contains the 12 bit address to be placed in the Program Address Counter (JMP*).

With bit 4 as logical "0", the address (bits 5-11) is address specified of page zero. With bit 4 as logical "1", the address (bits 5-11) is address specified of current page.

The memory addresses for the display bytes are supplied to the computer Memory Address Input from a Program Address Counter in the display.

The Program Address Counter will always start at address 0007_8. The contents of this memory cell are interpreted as an Indirect Address. Upon receipt of an EOF message, the Program Address Counter will go to the next address and wait for the start of the next frame. Recycling of the display is accomplished by the use of a 5407 (JMP* 7) instruction.

Typical Operation Time:	JMP,	1.0 USEC
	JMP,*	3.0 USEC

(All typical Operation Times exclude computer access time total Operating Time is less than the sum of Display and Computer

[illegible] computer access occurs during the Display operations)

VECTOR MODE (10)

This mode is used to either randomly position dots or to position the beam for succeeding dot, vector or symbol modes, or to string vectors (lines). Frame sync is also accomplished in this Mode.

CONTROL BYTE

This byte is used to set the conditions for interpreting the following pairs of Data bytes as either beam position, random dot position, or end of vector, depending on bits 7-8 (until a new Control byte is encountered)

With bit 7-8 = 00, the beam will be positioned only; with bit 7-8 = 01, the beam will write a dot after positioning; with bits 7-8 = 10, the beam will be controlled by the vector generator and the line will be dashed; with bits 7-8=11, the beam will be controlled by the vector generator and the line will be solid Bit 6 is for Frame sync. Blink bit 11 is provided for program compatibility but does not control the display.

Typical Operation Time: 4.0 USEC

DATA BYTE

Bits 1-10 of the first Data byte determine the X Position and Bits 1-10 of the second Data byte determine the Y Position. Bit 11 of the first Data byte is used to blank the beam

irrespective of any control bit settings in previous Control bytes. Bit 11 of the second Data byte controls blink.

Typical Operation Time:	Beam position	17.0 USEC
	Vector	50.0 USEC

SYMBOL MODE (10)

This mode is used to string a series of characters, (one character per Data byte) with each character individually controlled for size, offset, intensity and blink.

CONTROL BYTE

This byte is used to set the conditions for interpreting the following Data bytes as strung characters until a new Control byte is encountered.

Bits 7-8 set the margin characteristic, according to the following table:

00	NOP
01	Set Margin
10	Return to Margin
11	Return to Margin and Line Feed

The number of characters per line depends on the character size as set in the Data byte. Sixty-four normal size characters or 128 small size characters fill a complete horizontal line.

Line feed is based on 64 lines of characters per page. Intensity bits 9-10 and Blink bit 11 are for program compatibility,

only, and do not control the display.

Typical Operation Time: 4.0 USEC

DATA BYTE

The character determined by bits 1-6 (see table 2) will be written with the intensity determined by bits 9-10. Character size and offset will be determined by bits 7-8 according to the following table:

00 - normal, no offset

01 - small, superscript

10 - small, subscript

11 - small, no offset

Typical Operation Time: 13.0 USEC

PACKED SYMBOL MODE (11)

This mode is used to string a series of characters (two characters per data word), with the character string as a group controlled for size, offset, intensity and blink.

CONTROL BYTE

This byte is used to set the conditions for interpreting the following data bytes as strung characters until an escape character is encountered.

Characters determined by the following Data byte will be written in the size and offset determined by bits 7-8, with

the intensity determined by bits 9-10, and will blink as determined by bit 11. Size and intensity codes are as given in the Symbol Mode 10.

Typical Operation Time: 4.0 USEC

DATA BYTE

Two characters are packed into each Data byte, with bits 0-5 determining character 1, and bits 0-5 determining character 2 (see table 2). Because no control bit is available, one character code (77_8) is reserved as an escape code.

Typical Operation Time: 26.0 USEC
(for two characters)

DOT INCREMENT MODE (00)

This mode is used to program a series of closely spaced dots.

CONTROL BYTE

This byte is used to set the conditions for interpreting the following Data bytes as strung dots until a new Control byte is encountered. Bits 9-10 determine dot intensity; and bit 11 determines dot blink.

Typical Operation Time: 4.0 USEC

DATA WORD

One dot increment (ΔX, bits 1-3: ΔY, bits 4-6) is specified by each Data byte according to the following table:

001 $+2^0$

010 $+2^1$

111 $-2^0 = -1$

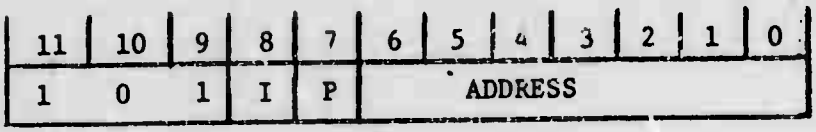
110 $-2^1 = -2$

others - no increment

Bit 11 blanks the dot regardless of any control bit setting in the previous Control byte. The program dot increment will be automatically repeated the number of times determined by bits 7-10 (up to 15).

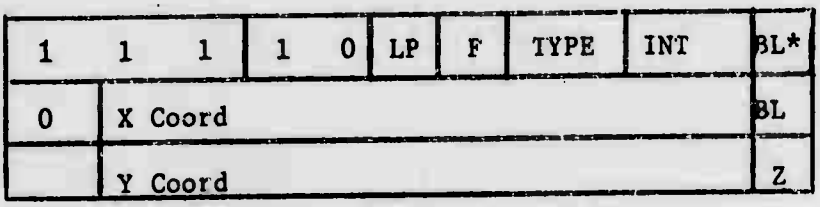
Typical Operation Time: 5.0 USEC/DOT

DISPLAY FILE BYTE ORGANIZATION



11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	I	P	ADDRESS						

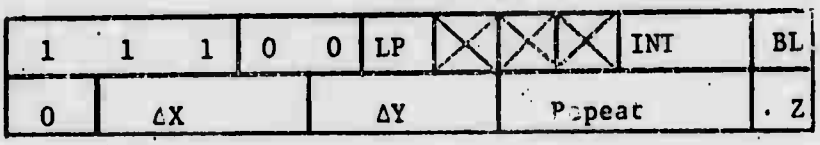
Program Control Byte



1	1	1	1	0	LP	F	TYPE	INT	BL*
0	X Coord								BL
	Y Coord								Z

Vector Mode Control Byte

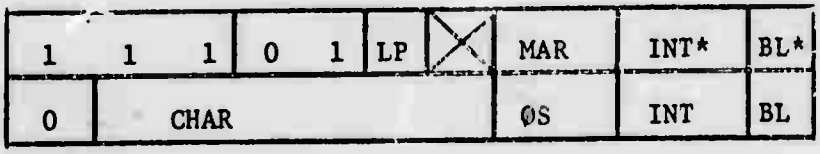
Vector Data Double Byte



1	1	1	0	0	LP	X	X	X	INT	BL
0	ΔX		ΔY		Repeat					Z

Dot Increment Mode Control Byte

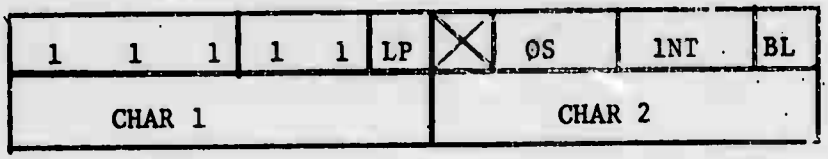
Dot Increment Data Byte



1	1	1	0	1	LP	X	MAR	INT*	BL*
0	CHAR						ØS	INT	BL

Symbol Mode Control Byte

Symbol Data Byte



1	1	1	1	1	LP	X	ØS	INT	BL
CHAR 1						CHAR 2			

Packed Symbol Data Byte

Packed Symbol Data Byte
(77_8 = escape)

LP = enable light pen
BL = enable blink
Z = blank
F = frame sync

INT = 00 - normal
= 01 - dim
= 10 - bright
= 11 - off

TYPE = 00 - position
= 01 - position, write dot
= 10 - dash
= 11 - solid

ΔX,ΔY = 001 +1
010 +2
111 -1
110 -2
Others - no increment

ØS = 00 - normal, no offset
01 - small, superscript
10 - small, subscript
11 - small, no offset

MAR = 00 - NØP
01 - set margin
10 - return to margin
11 - return to margin and line feed

Appendix 2

Tektronic Oscilloscope 453

GS provides special facilities for using the oscilloscope in connection with the half-tone display research.

The programming sequence is:

```
      INTEGER DF(171)
      CALL TEX(DF)
      DO 100 I=1,512

      (GENERATE A LINE OF THE RASTER)

100   CALL SNDFLE(DF)
```

Appendix B

Alphanumeric Character Codes

Symbol	ASCII	Fieldata	Trimmed-ASCII	Card
Ø (zero)	60	60	20	0
1	61	61	21	1
2	62	62	22	2
3	63	63	23	3
4	64	64	24	4
5	65	65	25	5
6	66	66	26	6
7	67	67	27	7
8	70	70	30	8
9	71	71	31	9
A	101	06	41	12-1
B	102	07	42	12-2
C	103	10	43	12-3
D	104	11	44	12-4
E	105	12	45	12-5
F	106	13	46	12-6
G	107	14	47	12-7
H	110	15	50	12-8
I	111	16	51	12-9
J	112	17	52	11-1
K	113	20	53	11-2
L	114	21	54	11-3
M	115	22	55	11-4
N	116	23	56	11-5
O	117	24	57	11-6
P	120	25	60	11-7
Q	121	26	61	11-8
R	122	27	62	11-9
S	123	30	63	0-2
T	124	31	64	0-3
U	125	32	65	0-4
V	126	33	66	0-5
W	127	34	67	0-6
X	130	35	70	0-7
Y	131	36	71	0-8
Z	132	37	72	0-9

Symbol	ASCII	Fieldata	Trimmed-ASCII	Card
a	141	-	-	-
b	142	-	-	-
c	143	-	-	-
d	144	-	-	-
e	145	-	-	-
f	146	-	-	-
g	147	-	-	-
h	150	-	-	-
i	151	-	-	-
j	152	-	-	-
k	153	-	-	-
l	154	-	-	-
m	155	-	-	-
n	156	-	-	-
o	157	-	-	-
p	160	-	-	-
q	161	-	-	-
r	162	-	-	-
s	163	-	-	-
t	164	-	-	-
u	165	-	-	-
v	166	-	-	-
w	167	-	-	-
x	170	-	-	-
y	171	-	-	-
z	172	-	-	-
Space	240	05	00	Blank
Control Characters	(see Note 1)	05	00	Blank
Rubout	177	05	00	Blank
!	41	55	01	11-0
"	42	-	02	-
#	43	03	03	12-7-8
$	44	47	04	11-3-8
%	45	52	05	0-5-8
&	46	46	06	2-8
'(apostrophe)	47	72	07	4-8
(	50	51	10	0-4-8
)	51	40	11	12-4-8
*	52	50	12	11-4-8

Note 1 - Control codes are formed by taking the regular code and subtracting 100_8.

Symbol	ASCII	Fieldata	Trimmed-ASCII	Card
+	53	42	13	12
,(comma)	54	56	14	0-3-8
-(hyphen)	55	41	15	11
.(period)	56	75	16	12-3-8
/	57	74	17	0-1
:	72	53	32	5-8
;	73	73	33	11-6-8
<	74	43	34	12-6-8
=	75	44	35	3-8
>	76	45	36	6-8
?	77	54	37	12-0
@	100	00	40	7-8
[	133	01	73	12-5-8
\	134	57	74	0-6-8
]	135	02	75	11-5-8
↑	136	-	76	-
←	137	-	77	-

Appendix 4

PDP-8 Graphical Display System

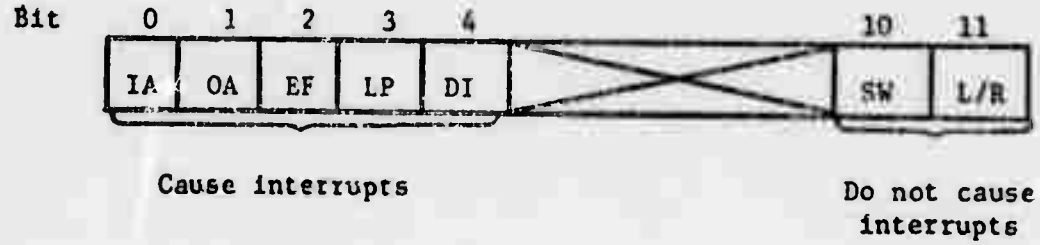
In the Graphical Display System it will be the function of the PDP-8 to act as a communications handler and storage device linking the displays with the UNIVAC 1108.

To accomplish this objective, additions have been made to the PDP in both circuit design and instruction codes.

The hardware design changes include addition of:

1. Interrupt processor flags.
2. An 1108 - PDP interface buffer and control circuitry.
3. A Digital-to-Analog converter and control equipment for a Tektronic Oscilloscope display.

The interrupt processor flags take on the following configuration:



IA - 1108 Input Acknowledged Flag

OA - 1108 Output Acknowledged Flag

EF - 1108 External Function Flag

LP - Light Pen Flag

1 = Interrupt

0 = No Interrupt

DI - Display Interrupt Flag

SW - Light Pen Switch

1 = Up

0 = Down

The following mnemonics have come into general use for creating display files. A complete description of the IDI instruction bit structure can be found in the IDI user's manual.

FRAM	= 7440	Frame Sync
JUMP	= 7400	Position Beam
LINE	= 7430	Line Mode Command
VEC	= 7000	Strung Dot Command
SYMB	= 7200	Symbol Mode Command
SCH	= 7600	Strung Symbol Mode Command
PEN	= 0100	Enable Light Pen Match
FIN	= 5407	JMP *7
DOT	= 7410	Position Beam, Write Dot
DASH	= 7420	Dash Lines
DIM	= 0002	Dim Intensity
BRI	= 0004	Bright Intensity
OFF	= 0006	No Intensity
XPØ	= 0000	ΔX +0
XP1	= 0400	+1
XP2	= 1000	+2
XM1	= 3400	-1
XM2	= 3000	-2
YPØ	= 0000	ΔY +0
YP1	= 0040	+1
YP2	= 0100	+2
YM1	= 0340	-1
YM2	= 0300	-2
DI	= 6XXX	Display Interrupt

Each display user should acquaint himself with the operation of the PDP-8 and IDI consoles. The "PDP-8 User's Handbook" contains the necessary information for operating the computer.

The IDI display light pen is a device used by the operator at the display. A "Micro-Switch" is provided on the pen and its position can be program-sampled.

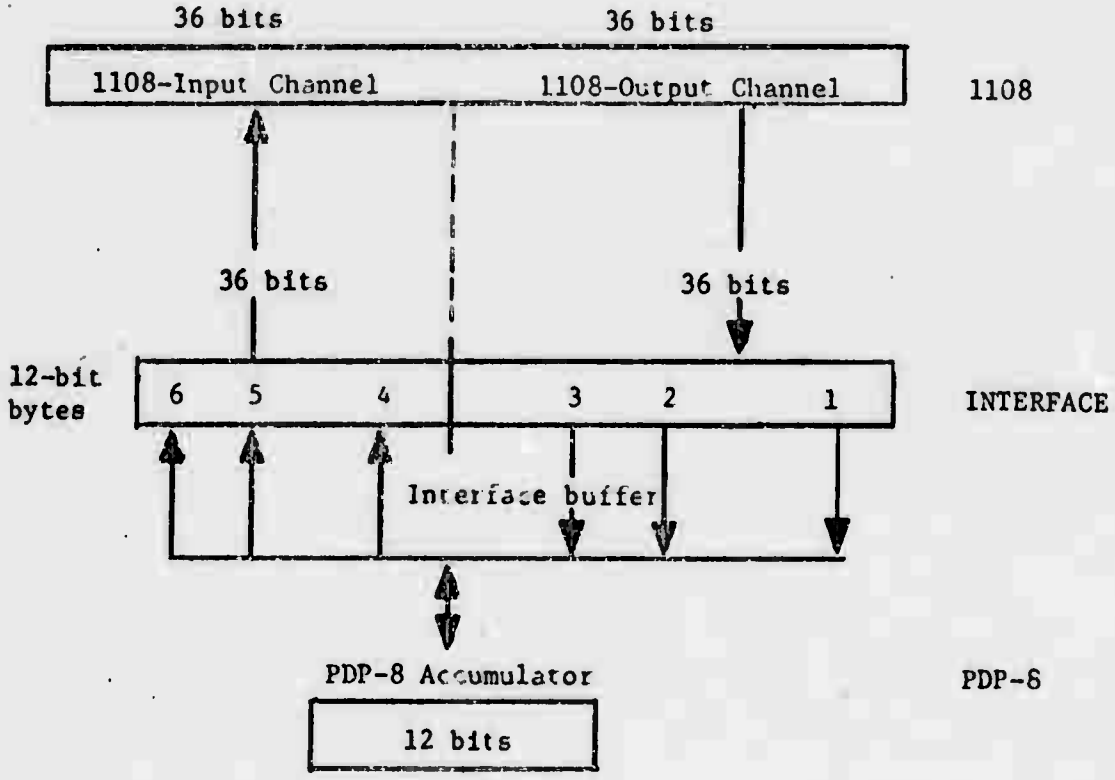
Appendix 5

The 1108 - PDP-8 Interface

The PDP-1108 Interface is first described by its logical function and second by the programs needed to drive it.

Interface Hardware

The interface joins Channel 1 of the 1108 to the PDP-8 accumulator and provides control of input-output with two 36-bit buffers and suitable control circuitry. A block diagram is given below.



A data transfer between computers involves clearing and loading the buffer, then signaling that this process has been completed. This signaling is done by various pulses and logic levels as described below.

A more complete description (voltage, timing, etc.) can be found in the UNIVAC 1108 I/O Channel Specification.

1. Output Acknowledge (OA)

   This line is activated by the 1108 when its output data is available on the channel.

2. Input Acknowledge (IA)

   This line is activated by the 1108 when its input data lines (in this case, the interface buffer) have been sampled by the 1108.

3. External Function (EF)

   The external function line is activated by the 1108 when its output data lines contain an external function word.

4. Input Data Request (IDR)

   This line is activated by the PDP when its output data is available in the interface buffer.

5. Output Data Request (ODR)

   This line is activated when the PDP is ready to receive data from the 1108.

6. External Interrupt (EI)

   This signal is activated by the PDP when a status word is present on the 1108 input lines.

At the present time, the interface hardware is operating at an acceptable standard.

NOTE: In operating any type of program which uses the interface, the off-on switch below the interface must be in the on position. When turning the computer power on and/or off, be sure the interface switch is in the off (down) position.

Interface Programming

The two basic transmit/receive routines needed for the 1108-PDP communications have been written and successfully used. At present a higher level interface handler with teletype I/O is being debugged.

The following operation codes have been chosen for communications between 1108 and PDP-8.

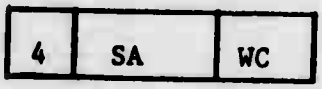
1108 to PDP

Function Word = 1 (1108 read)

Do you have any?

1. LP match
2. Tracking cross match
3. TTY input

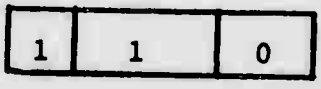
- Is printer busy?



- Send this block

PDP to 1108

Swap.



- Swap Graphics user IN

Input Medias and their capabilities

Input to the 1108 program will consist of the following 3 forms:

1. Teletype Input Fuction

   Input from the keyboard will be stored in a buffer in ASCII form and sent to the 1108 upon command. If the buffer becomes full before the 1108 asks for more input, none will be accepted. Each accepted character will be echoed by the PDP immediately. A "line-feed" will be produced after typing "carrier-return."

2. Light Pen Pointing Function

   This function provides the graphics user with the ability to specify existing objects on the screen by pointing to them

with the light pen. The light-pen switch should be depressed to indicate the line/character chosen.

3. Light Pen Tracking Function

This function will provide the graphics user with the absolute X,Y coordinates on the scope face where he is pointing. The tracking cross will follow his movements of the light pen to do this. These X,Y coordinates will be saved in a buffer until call from the 1108. New X,Y data should be stored in the buffer only when the pen has been moved more than a specified tolerance.

L/R - Left/right half word character indicator for strung symbol mode.

The added instruction codes are as follows:

1. Interrupt Processor Instructions:

6111	Clear IA Flag
6112	Clear OA Flag
6121	Clear EI Flag
6122	Clear DI Flag
6101	Clear Light Pen Flag
6102	IDI Address Register → PDP Accumulator
6104	Interrupt Flags → PDP Accumulator
6124	Restart the display scan from the beginning of the display file
6114	Stop Display

2. 1108 - PDP Interface Instructions:

6231	Clear and Reset Interface
6221	(AC) → Link Buffer
6224	Give External Interrupt
6222	Give Input Data Request
6211	(Link Buffer) → AC
6214	Give Output Data Request

3. Digital-to-Analog Converter:

CLA;6147	Clear X,Y,Z registers
6142	$(AC)_{2-11}$ → X register
6144	$(AC)_{2-11}$ → Y register
6141	$(AC)_{2-11}$ → Z register

1108 to PDP

Function Word = 2 (1108 write)

Write the block of data of length WC PDP words into PDP core starting at location SA.

Type out the contents of PDP-8 core from location SA until an End-of-Text (204_8) is encountered.

Start executing instructions found at location SA in PDP core.

CALL ARPAIO (Actual Words, Status, Op Code, Buffer Address, Requested No. of Words, Infor Word)*

Actual Words

The number of words actually written or read from a device like 200 out of 350 requested since you've reached end of allotted space.

Space

0 = Normal

1 = Partial Read/Write

2 = Abnormal i.e. EOF

3 = Error Condition with Fastrand Hardware

4 = Label not on Fastrand/Read or Write less than three word

5 = No Room on Fastrand File/Label not on Fastrand

6 = Fastrand File Not Opened Yet - or Trying to Open with ≠ 0 Op Code

7 = No Unit Available - 26 Units have been Already Assigned to Fastrand

8 = Trying to Read/Write $\frac{\text{out of}}{\text{into}}$ Protected Core

9 = Illegal Functions

Op Codes

Fastrand

0 = Open with File Name in Buffer Address

i.e. 0,($GMΔΔ$)

1 = Read

2 = Write

3 = Rewind

4 = Write EOF

5 = Backspace

* For multiple files see Boam, Blackburn or Palkovic

PDP-8

10 = Read

11 = Write

1004

20 = Write (Not Implemented)

Buffer Address

Array where Data Is/Is to be Stored

Requested No. of Words

No. of Word to be Read/Written

Starting Location PDP-8 (Not Implemented)

If used, first word of Buffer will be filled by Driver Package.