MPS/CO
BC/je

14 November, 1972

PDS1 DISPLAYS

Please find attached the new version of the programmer's guide, which describes the 8 K software - operational (hopefully) by the end of the long shut-down. There are many small additions, but note especially two changes:

1) it will no longer be necessary to load CHKLS into the PDS1 - it's there all the time

2) "Return changes" mode will exist no more. You have been warned.

Because of debugging, no promise is made about the availability of the PDS1's during the shut-down.

B. Carpenter

Distribution

IBM Programmers
J.H.B. Madsen

MPS/CO Computer 72-4
14 November, 1972

PDS1 DISPLAYS - PROGRAMMER'S GUIDE

B. Carpenter

CONTENTS

/je

INTRODUCTION

This document is intended as a manual for programmers wishing to use the PS or PSB IMLAC PDS1 displays. It supersedes previous documents on the subject.

Part 1 consists of a general description of the system and of how to use the displays from remote programs.

Part 2 is for the use of those who wish to write PDS1 programs, thus extending the standard facilities to suit their personal requirements. Other documents relevant to Part 2 are the manual "PDS1 Assembly Language PDSA" and two IMLAC manuals, "User's Reference Manual" and "Programmer's Supplement".

Distribution

IBM Programmers
J.H.B. Madsen

1. PART 1

1.1 General description of system

Two IMLAC PDS1 display mini-computers are installed in the MCR. Each has two display consoles (with keyboards) attached to the processor unit. Both have 8 K of 16 bit core store. The first, which has a light pen, is dedicated to

a) PS operation (first console)

b) injection (second console).

The second PDS1 is dedicated to the PSB. The PS and injection consoles will back each other up and the two PDS1's will back each other up.

The two processors are connected to the computer system by parallel interfaces and STAR so that information may be transferred at a rate of 40 K words/second in blocks of approximately 250 words.

The software available to make use of the PDS1's consists of four main parts:

1) communications routines

2) EXEC - a program resident in the PDS1

3) Assembler program for PDS1 Assembly language

4) programs and routines to use the disk file on which PDS1 coreloads are stored.

(Items 3) and 4) are discussed in Part 2 of this document).

The normal procedure is that a program in a remote computer uses the communications routines to transmit data to, and to receive data from, a particular PDS1. The EXEC in the PDS1 handles certain types of transmission automatically, but if the standard facilities are

insufficient the user programmer may add to them by writing a PDS1 program which will, in effect, extend the EXEC. This subject is covered in Part 2.

1.2 Display and keyboard possibilities

The screen of each display is 20 cm (horizontal) by 30 cm (vertical). The analogue XY signals are produced by hardware vector generators operating under software control. This software is in the form of a"display list" in the PDS1 core store, which is executed by the display processor and modified by the mini-computer processor.

The EXEC program modifies the display list according to information received from the user program or from the keyboard.

For character display, the screen provides a matrix of 54 lines of 80 characters (character size approximately 2.5 mm, similar to line printer).

The character set is:

letters : A - Z (upper case only)
digits : 0 - 9
symbols : + - . , : [ ] * / = > < %
format characters: space, new line, tab.

Lines 1 - 50 are available for users. Lines 51 - 54 are reserved for the system. A tab character causes an immediate skip to character position 43 on the current line. (See also page 1.6.1.)

For graphical display, a resolution of $\frac{1}{3}$ mm on the screen can be achieved. EXEC provides either point or histogram plots of one or two graphs on up to five sets of axes simultaneously.

The maximum amount of information that can be displayed at one time is limited by the core available for the display list and by the time avaible to display each frame. Some data are given in section 1.5

to help in estimating the limits in a particular case, but experimentation will always be necessary.

The keyboard provides all the displayable characters plus some others which are not used. In addition there are a number of control keys, some of which have predefined meanings, and a key "CTRL" which may be used to modify a character for control purposes in a way analagous to the SHIFT key. Upper and lower case letters are treated alike by EXEC. There is a list of control key allocations in the Appendix.

1.3 PDS1 Resident EXEC program

1.3.1 General

The EXEC consists of three main parts: a monitor, an interrupt servicing routine and a set of subroutines.

The monitor is a short looping program which checks certain software flags and carries out actions requested by the user program or the keyboard, where these are not carried out at interrupt level. It includes a check-list handler.

There is only one interrupt level and therefore only one interrupt servicing routine, which handles the following:

a) IBM transmissions to PDS1

b) 33 1/3 Hz display frame synchronization

c) keyboards and cursor movement.

There are various subroutines which are capable of modifying the display and of transmitting data to the IBM.

1.3.2 Extending EXEC

The normal mainline program in the PDS1 is the looping monitor. Where this does not provide a particular facility, the user may put his own mainline program in the PDS1. This program may use the EXEC subroutines

and other features of EXEC to give a high degree of flexibility. The user will not need to learn the details of display generation, and the main benefit will be more effective use of the keyboard.

1.4 Routines in the IBM

1.4.1 General

Data is transmitted between the user and a PDS1 in the form of a file. The file is preceded by a 16 bit word of which the top 8 bits are a control function (CF) and the bottom 8 bits are a word count (WC). The useful length of a file is limited to slightly less than 255 words and its format is defined by the CF bits.

Interrupts are also exchanged between the computers. The only one of these which affects the user is called PRQ, i.e. "PDS1 Request". A typical procedure is as follows. An IBM coreload transmits an EBCDIC file to the PDS1, followed by a data file which is to be changed by the operator. The PDS1 displays the text, which is in the form of a checklist, and the operator types changes. When he has finished, he presses the "TRANSMIT" key. EXEC then sends a PRQ interrupt to the IBM. The interrupt servicing subroutine in the IBM reads in the PRQ and then the CF/WC word. The latter is placed in a skeleton flag in the IBM. The IBM user coreload picks up the CF/WC and calls a routine to read in the data file as changed by the operator. The user must reset the skeleton flag to zero.

The PDS1 interrupts are on Level 5. Difficulties may occur if a program at Level 5 or higher priority attempts to transmit to or receive from a PDS1.

The skeleton flag mentioned above serves both as an interrupt indicator and as a means of passing the CF/WC word to the user. There is an extra skeleton flag associated with the PDS1's, called the queue indicator. When the PDS1 send a PRQ interrupt, the coreload associated

with each non-zero bit in the queue indicator is queued, and the bit is reset. The interrupt indicator is set in any case. It is a user responsibility to set the bit required in the queue indicator. Because of restrictions in MPX it is only possible to have a total of 16 such queueable coreloads.

1.4.2 Transmission of data to PDS1*

(Check latest version of calling sequence in subroutine library manual.)

```
            CALL        PDSTX
            DC          LIST
            DC          COMPL
             |
             |
             |
             |
             |
             |
             |
             |
LIST        DC          /XYYZ       TRANSMISSION CONTROL WORD
            BSS         5
            DC          /QQ00+N     CF/WC
            BSS         6
BUFR        BSS         N           MAXIMUM 244
             |
             |
             |
             |
             |
COMPL       DC          *-*         COMPLETION CODE
```

Transmission control word

1st digit	:	PRU number
2nd + 3rd digits	:	button number
4th digit	:	0 = PS PDS1 1 = PSB PDS1

The first three digits are included primarily as a user identification. A dummy code (with PRU number 0) may be used when no program request unit is involved.

*Also see description of PDSAL in the subroutine library manual

The routines in the IBM keep a record of the busy status of each PDS1. If a display is created by a call with a particular transmission control word, subsequent calls will be refused unless they provide the same word. The busy status must be cleared by PDLIB when a user no longer needs the display.

PDS1 control function

These 8 bits, supplied by the user, are transmitted to the PDS1 at the head of the file. See Appendix for control function definitions. Only those marked with an asterisk are recognised and processed automatically by EXEC in the PDS1. The others can be exploited by writing a PDS1 program.

Completion code

0 after successful transmission

>0 indicates an error - see subroutine library manual for details

Format of buffer when CF indicates EBCDIC

```
BUFR    DC      first line          } present only if delete
        DC      number of lines     } specified in CF

        DC      line                } position of first charac-
        DC      character pos.      } ter to be inserted

        DC      word count

        EBC     .TEXT.
```

In the text, newlines and tabs may be represented by the normal EBCDIC codes, or by $ and ' (apostrophe) respectively. Character position 81 does not correspond to position 1 on the next line - you must include newline or $ characters.

Care must be taken to use the correct card code for brackets. Model 26 punches give the wrong code. The & sign may be used for +.

(If only the delete is required, give a dummy line number of zero only. Deletion of all graphics may also be specified in the CF.)

Double size characters

To obtain double size characters, insert the control character "ENTER SCALE 2" in the EBC statement. Following characters will be double size. At the end of the double size message, you must insert the control character "EXIT SCALE 2". You must insert a space after every character, and two new lines instead of one.
(M E S S A G E $ $).

There are therefore 25 lines of 40 double-size characters. You may combine normal and double characters in one message if required, as long as at the end of the message the PDS1 is in normal size.

```
ENTER  SCALE  2 - - - - -EBCDIC / 5E ,  punch 11, 8, 6
EXIT   SCALE  2 - - - - - EBCDIC / 5F,,  punch 11, 8, 7.
```

Multiple tabs

In addition to the normal tab to position 43, a tab to every 8th position is available by including in the EBCDIC string !n where n is between 1 and 9. This gives tabs in positions 8, 16, ... 72.

Format of buffer when CF indicates check-list data file

The text already on the screen should contain a number (maximum 64) of the following special characters:

- : defining a 2-character EBCDIC field
- > defining a 6-character EBCDIC field
- = defining a 6-character signed decimal field
- / defining a 4-character hexadecimal field.

The special character must be followed by the appropriate number of characters or blanks.

The data-file contains the same number of items as there are fields in the text. The initial contents of each item are defined by the programmer. Decimal and hexadecimal fields are in pure binary, and EBCDIC fields are in packed EBCDIC.

When the data file reaches the PDS1, the check-list section of the EXEC becomes active. The operator may select a field using CTRL/→ and CTRL/←; the special character defining the selected field is intensified. He may then type in a modified version of the field; this appears on the screen and the corresponding item in the data file is modified.

When the operator presses "XMIT", the sequence described in 1.4.1 occurs, the data being read by PDSRX. If the CF indicated "cursor request" the cursor line (top byte) and position (bottom byte) are written into the first word of the data file; thus the first data item must be a dummy if the cursor position is requested.

A decimal field may contain a decimal point which is completely ignored in the conversion to binary (e.g. 24.8 becomes 248). Remember that the displayed text does not automatically correspond to the initial contents of the data file, so the programmer must provide the "before" versions of all fields in the text.

Format of buffer when CF indicates graphics

If "delete" is specified in the CF, all graphics previously on the screen will be deleted. Drawing operations automatically delete previous graphs with the same logical number.

BUFR	DC	n	LOGICAL NUMBER OF AXES (0-4)
	DC	1	DRAW AXES
		0	AXES ALREADY DRAWN
		-1	DELETE AXES AND GRAPHS
	DC	1n	LINE AND CHARACTER POSITION
	DC	pn	OF ORIGIN
	DC	-x	LIMITS OF X
	DC	+x	
	DC	npx	LENGTH OF X-AXIS ON SCREEN (CHARACTER POSITIONS)
	DC	ndx	NUMBER OF SUBDIVISIONS OF X-AXIS
	DC	-y	SAME FOR Y
	DC	+y	
	DC	npy	
	DC	ndy	
	DC	ng	NUMBER OF GRAPHS (0, 1 or 2)
	DC	m	* LOGICAL NUMBER OF GRAPH (0-1)
	DC	gc	* POINT PLOT
		-gc	* HISTOGRAM PLOT
	DC	pc	NUMBER OF POINTS
	BSS	2*pc	LIST OF PAIRS OF X AND Y VALUES

Only present if axes are to be drawn

For each graph

Example 1

Draw a set of axes, logical number 3, for displaying Y values from -300 to +300 and X values from 0 to 100.

---

* m = o -- points plotted as "X"
m = 1 -- points plotted as "."
gc = 1 --- delete previous points
gc = 2 --- no delete of previous points (stacking mode)
gc = 4 --- overwrite previous points (refresh mode)
gc+/4000--- escape mode (for vector graphics, etc.)

```
BUFR    DC      3           LOGICAL NUMBER
        DC      1
        DC      10          } NEAR TOP OF SCREEN,
        DC      1           } EXTREME LEFT
        DC      0           } X  LIMITS
        DC      100         }
        DC      80          FULL WIDTH OF SCREEN
        DC      10          10 SUBDIVISIONS
        DC      -300        } Y  LIMITS
        DC      300         }
        DC      20          20 LINES HEIGHT
        DC      12          MARK EVERY 50 UNITS
        DC      0           DRAW NO GRAPHS
```

Example 2

Draw or refresh a graph of 14 points, in histogram form, on these axes.

```
BUFR    DC      3
        DC      0
        DC      1           DRAW ONE GRAPH
        DC      0           LOGICAL NUMBER OF GRAPH
        DC      -1
        DC      14
        BSS     28          DATA
```

Note

The PDS1 uses single-length multiply and divide routines, so in order to get good results in graph plotting it is important to use data suitably scaled.

Each character or line space on the screen corresponds to 32 least significant bits in the PDS1. The scaling factor stored in the PDS1 for the x and y directions of each set of axes is:

$$s = \frac{npx * 32}{\Delta x} \quad \text{or} \quad \frac{npy * 32}{\Delta y}$$

Here Δx is the range of x values as defined when the axes are drawn, and npx is the total length of the x axis in character positions.

Δx and Δy must be less than 2560. The values of s must be less than 64. They are stored in fixed-point form in 14 bits, i.e. to an accuracy of $\pm$ 0.004.

Values of x and y outside the range of the axes will be plotted, unless they would be off-screen or result in arithmetic overflow.

1.4.3 Receiving data from PDS1 in IBM 1800

(Check latest version of calling sequence in subroutine library manual.)

```
            CALL    PDSRX
            DC      LIST
            DC      COMPL
            '
            '
            '
LIST        DC      /XYYZ       TRANSMISSION CONTROL WORD
            DC      N+1         WORD COUNT +1
            DC      *-*
BUFR        BSS     N           MAXIMUM 254
            '
            '
            '
COMPL       DC      *-*         COMPLETION CODE
```

Transmission control word and completion code

As for PDS TX.

Word count

This is the bottom 8 bits of the CF/WC word provided in the skeleton flag (section 1.4.1). For CF definitions, see Appendix. Only "check-list data" is provided automatically by EXEC. The others can be exploited by writing a PDS1 program.

Format of buffer for check-list data file

As described in 1.4.2.

A routine is provided to simplify waiting for the reply when using the check list technique.

```
Example:        LOOP    ....
                        ....
                        (computing while waiting)
                        ....
                        CALL    PDSRW       PDS1    READ/WAIT
                        DC      LIST
                        DC      ERROR       BRANCH HERE IF ERROR
                        DC      LOOP        BRANCH HERE IF NO REPLY
                        ....
                        (process reply)
                        ....
```

See the subroutine library manual for details.

1.4.4 Clearing busy status of PDS1

```
                CALL    PDLIB
                DC      TRCON
                |
                |
                |
TRCON           DC      /XYYZ       TRANSMISSION CONTROL WORD
                                    (AS FOR PDSTX)
```

This clears the busy status for any PDS1 in use by the calling program. It must be called when the user has finished with a display; otherwise future users will be blocked.

1.4.5 Checking the abort status of PDS1

```
                CALL    PDSAB
                DC      TRCON
                DC      LABEL
                |
                |
                |
TRCON           DC      /XYYZ       TRANSMISSION CONTROL WORD
                                    AS FOR PDSTX)
```

If the PDS1 specified has sent an abort request, the calling program is continued at address LABEL.

An abort request is sent by EXEC when the operator keys in CONTROL/A. It could also be sent by a user's PDS1 program.

1.5 Length and duration of display list

The maximum core available in the PDS1 for the display list is 3000 words, and the maximum time per frame is 40 000 µsec. If these limits are exceeded, the display will not be correct. The following data may be used to estimate the core usage and time for a particular case. (A shared PDS1 would have half this capability per user. Sufficient space is always reserved for alarm messages.)

Fixed and system area	80 words	1300 µsec
Each character (visible, space)	1 word	20 µsec (average)
Each set of axes (no labels)	* 100 words (average)	** 500 µsec (average)
Single point	5 words	t+14 µsec
Histogram point	5 words	t µsec

(t = 8 to 40 µsec depending on data)

Example

Fixed area	80 words	1300 µsec
500 characters	500	10000
2 axis sets	200	1000
2 x 50 points	500	3000
2 x 20 histogram pts.	200	800
TOTAL	1480 words	16100 µsec

* 10+2.(ndx + ndy) words
** (ndx + ndy + 2).(4 + t/2) µsec

1.6 Errors

Bad calls, etc., return a suitable completion code to the user. However, the files transmitted to the PDS1 are not checked, so if they are faulty the calling program will not be informed. The following error messages may appear on the PDS1 screen system area:

```
BAD  CALL
CORE FULL
 ?   FILE   (Unknown CF on file)
 ?   INPUT  (Unknown interrupt or undefined control key)
IBM NO RESPONSE
```

1.7 Timing note

Information is not displayed instantaneously after transmission to the PDS1. If the programmer does not allow time for the PDS1 software to edit the display list before the next transmission, his level will be suspended by PDSTX or PDSRX until the PDS1 sets a completion bit in its status word.

The time taken to display data depends on the position on the screen, on the amount of information involved, and on the existing display. An average figure might be 100 msec but the range of variation is great.

Deletion and insertion of text or graphics involves physically moving up to 3 K of data in the PDS1 memory. Therefore it is more efficient to refresh (i.e. to overwrite) than to delete and insert; it is more efficient to write complete lines than to write partial lines; and it is more efficient to write from left to right and from top to bottom (remembering that graphics are logically below text).

1.8 Back-up arrangements and faults

The transmission routines automatically attempt to use the other PDS1 if the unit requested is broken. The skeleton flags are treated as if back-up had not occurred, so the calling program need take no account of back-up arrangements.

On the PS PDS1, the same display is seen on both screens, but only one keyboard is in use at a time. When the operator keys in CONTROL/B on either keyboard, EXEC changes to the one not currently selected. EXEC initially selects keyboard A (on the central console). A user's PDS1 program can also change the keyboard selection.

In the event of a fault destroying EXEC, the operator may force a reload by pressing once or twice the FCS button on the PDS1 interface. An interrupt coreload in the IBM will attempt to reload the PDS1 from disc. If this operation succeeds, the screen should be cleared and the message *RESTART should appear on the system area. If it fails, the PDS1 interface may be switched to "off-line" and FCS pressed again. The back-up PDS1 will then be used.

PART 2

2.1 General

As indicated in section 1.3, the EXEC does not fill the entire PDS1 core memory and it does not handle all possible types of file. In particular, its use of the keyboard is rather restricted. By writing a program to fill the remaining core of the PDS1, the user can extend the facilities of the system. There are three main advantages to this:

1) direct use of keyboard and display without using computing time elsewhere
2) user definition of file format, if required
3) user definition of method of interaction between operator and console (question-and-answer, interpretive language, check-list, light-pen, etc.)

The user need only learn a part of the PDS1 assembly language and will not need to program the display directly. He will also need to know a little about the organization of EXEC in the PDS1.

2.2 PDS1 Processor and Assembly Language

The PDS1 processor unit has two important registers:

1) the program counter
2) the accumulator.

The program counter contains the core address of the instruction being executed. The accumulator is used in the normal way for arithmetic calculations. The carry from Bit 0 of the accumulator complements a special bit, the link. In rotate operations, the link is transferred to Bit 0 and Bit 15 is transferred to the link.

Instructions fall into four main classes:

1) addressed instructions (arithmetic and jump)

2) micro-instructions (tests and manipulations of the accumulator)

3) IOT instructions (input/output)

4) display processor instructions.

Full descriptions of the instructions are given in the PDSA Assembly Language Manual and in the manufacturer's manuals. For simple applications, IOT instructions and display processor instructions may be ignored.

Note that all PDS1 instructions are single-length. The address field of an instruction contains an 11-bit absolute address; therefore the core store is divided into 2 K pages for addressing purposes. Locations outside the current 2 K page must be addressed indirectly.

The Assembly language which is available at CERN allows the programmer to use symbolic names as addresses, to define buffers with BSS statements, etc., but does not have more complex features such as macros. It uses the same card format as the IBM 1800 Assembler, but in general the mnemonics are different.

2.3 Getting a program into a PDS1

The PDSA Assembly Language Manual describes how to build a PDS1 core load (PCL) and store it in the disc area reserved for PCL's.

Assuming that EXEC is running correctly in the required PDS1, a PCL may be loaded from the IBM by use of the library subroutine PDSCL. (Check latest version of calling sequence in subroutine library manual.)

```
            CALL    PDSCL
            DC      NAME
            DC      LIST
            DC      COMP
            :
            :
            :
            BSS  E  0
NAME        DN      CLNAM       NAME IN PACKED FORM
            :
COMP        DC      *-*         COMPLETION CODE
            :
LIST        DC      /XYYZ       TRANSMISSION CONTROL WORD
            BSS     260
```

NAME

This is the PCL name as defined when it was built.

COMPLETION CODE

As for PDSTX, plus 16 { Name does not exist / Disc fault.

PDTEM can be used to check the completion code.

TRANSMISSION CONTROL WORD

As for PDSTX.

2.4 Facilities offered to the user by EXEC in the PDS1

2.4.1 General

As discussed in section 1.3, EXEC carries out input/output operations and display modifications. This section describes how a user program employs these EXEC facilities.

2.4.2 Skeleton flags

Symbolically, skeleton flags are addressed as SKF+n. (See section 2.5 for values of EXEC symbols). These are the uses of the skeleton flags:

SKF+1	Normally zero. When non-zero, a binary file has arrived. SKF+1 contains the CF/WC word, exactly as supplied to PDSTX in the IBM. The user should reset SKF+1 to zero. (For binary files, see section 2.7.)
SKF+2	Normally zero. When non-zero, a character has arrived from the keyboard currently selected. SKF+2 contains the EBCDIC value of the character (-1 for delete, $ for newline, apostrophe for tab.). User should reset SKF+2.
SKF+3	Keyboard selection. 0 = Keyboard A, -1 = Keyboard B.
SKF+4	Communication between interrupt level and monitor (system use only).
SKF+5	Alternates between 0 and -1 every 40 msec.
SKF+6	Load address for the file reported by SKF+1.
SKF+7	Normally zero. When non-zero, transmit key has been pressed. The user should reset SKF+7 to zero.
SKF+8	Communication between parts of ISSR (system use only).
SKF+9	Communication between EXEC routines (system use only).
SKF+10	Check-list busy indicator (read only).
SKF+11	Current line number of cursor (read only).
SKF+12	Current character position of cursor (read only).
SKF+13	Check-list data file has arrived (system use only)
SKF+14	Control-key flag (system use only).
SKF+15	User timer (incremented every 40 msec.

2.4.3 Escapes

A user program normally monitors incoming files and the keyboard by use of SKF+1 and SKF+2. However, an "escape" facility is provided so that the user can write part of his program to run on interrupt level.

When an undefined CF is detected on an incoming file, the data is read into core starting at address TXBUF. Then, still at interrupt level, the interrupt servicing subroutine executes the instruction JMP I UFESC, Normally the location UFESC contains the address of coding to display an error message and return to the mainline. During program initialisation the user may execute these instructions to modify UFESC:

```
LAC      KUSAD
DAC  I   L$UFE
```

KUSAD contains the address of coding to handle the unknown file as the user wishes. Then, when such a file arrives, the ISSR will execute an effective JMP I KUSAD. After processing the file, the user must set a completion bit by executing ISZ I L$TCP.

The user program must leave the interrupt level by executing JMP I L$BAK. It is absolutely forbidden to call any EXEC subroutine while on interrupt level.

There are two other escapes which work analogously. An unknown interrupt (e.g. light-pen) causes the ISSR to execute JMP I UIESC, and an unknown control key causes JMP I UCESC. In this case the accumulator contains the binary value supplied by the keyboard. There is also a special escape for the vector graphics (see B. Carpenter).

2.4.4 Leaving a program

Normally, the last logical statement in a program should be CALL EXIT. This causes EXEC to restart its monitor loop, after clearing SKF+1 to SKF+15. There is no transmission to the IBM. The escape addresses (section 2.4.3) are reset to their normal values. The display is not cleared.

If the operator presses CONTROL/A the program will be aborted. EXEC will send an abort request to the IBM and will then execute CALL EXIT. (Also see descriptions of PDFCS and PDFRS.)

2.4.5 Suspension

User programs will spend most of their time waiting (principally for characters from the keyboard). Waiting loops must always contain a CALL SUSPN; this allows routine display and check-list handling to continue in parallel. SUSPN changes the accumulator and gives no completion code.

2.4.6 EBCDIC subroutines

```
        CALL    INSST
        DC      N       LINE
        DC      M       POSITION
        DC      LIST
        |
        |
        |
LIST    DC      WC      WORD COUNT (GREATER THAN ZERO)
        EBC     .TEXT.
```

This subroutine displays the text, starting at the line and position given. See section 1.4.2 for punching conventions, etc. To display one character, WC = -1; the character is in the low byte of the next word. It returns with zero in the accumulator if OK and -1 if there is an error. (An error message will be displayed the next time SUSPN is called.)

```
        CALL    DELLN
        DC      N       FIRST LINE
        DC      NUM     NUMBER OF LINES
```

This deletes NUM lines starting at line N. Error code as for INSST.

2.4.7 Graphics subroutine

```
          CALL      DRGRA
          DC        LIST
          |
          |
          |
LIST      BSS       N
```

This carries out a graphics drawing operation according to the contents of LIST. The format of LIST is identical to that given in section 1.4.2 Error code as for INSST.

```
          CALL      DELGR
```

This deletes all graphics. No error code. Contents of accumulator changed.

2.4.8 Error Monitor subroutine

```
          CALL      ERMON
          DC        MESS
          |
          |
          |
MESS      DC        WC
          EBC       .ALARM.
```

If the accumulator contains zero, the message supplied is displayed in the next position on the system area at the bottom of the screen. The message may be up to 80 characters, not including a new line.

If the accumulator has certain non-zero values, then no parameters are supplied and fixed messages are displayed:

```
-1      BAD CALL
-2      FULL
-3      ? FILE
-4      ? INPUT
-5      RESTART
-6      IBM NO RESPONSE
```

There is no completion code and the accumulator is changed.

2.4.9 Transmission to IBM

```
        CALL    TRFIL
        DC      LIST
        DC      WC          WORD COUNT
        :
        :
        :
LIST    BSS     WC
```

The CF bits for the transmission are supplied in the top half of the accumulator. Control functions are defined in the Appendix. The format of the list is indicated by the CF. Only "check-list data" format is defined for system use.

TRFIL sends a PRQ interrupt to the IBM. This routine normally remains busy until the user's program in the IBM has called PDSRX in response to the interrupt.

(However, if the CF indicates one-word transmission, by having the two most significant bits equal to 1Ø, there are no parameters and no response by a user's program in the IBM is required.)

TRFIL gives no completion code and the accumulator is changed. Before calling it you must wait until the location TFBSY contains zero.

2.5 Core available and addresses

1 K is available to user programs, starting at the pre-defined address FSTAD. Three methods are used to communicate with the EXEC via predefined symbols:

1. CALL routine (translated by the assembler into an indirect JMS)
2. To address certain key locations in EXEC, address indirectly via a pre-defined symbol (e.g. to address UFESC, use DAC I L$UFE)
3. Other address, e.g. SKF, are pre-defined only as absolute addresses. For example, to access a skeleton flag:

```
        XAM   I   ADSKF
        ....
        ....
ADSKF   DC        SKF+10     USER PROVIDES INDIRECT ADDRESS
        ....
        ....
```

The following table gives the pre-defined symbols. Others could be added on request.

CALL addresses (location in user page which contain entry point addresses of EXEC subroutines)

SUSPN	FNCHN
INSST	
DELLN	
DRGRA	
ERMON	
TRFIL	
EXIT	

Key addresses

Symbol	Contents	Use
L$TCP	TCP	Completion indicator (see section 2.4.3)
L$TFB	TFBSY	TRFIL busy indicator
L$UFE	UFESC	Escapes (see section 2.4.3)
L$UIE	UIESC	
L$UFE	UFESC	
L$ESC	ESCPT	
L$BAK	BACK	Return from interrupt
L$N	BL+3	

Absolute definitions

FSTAD	User program first address
SKF	Base address for skeleton flags
TXBUF	Buffer for files from IBM

The first statement in a user's program must be ORG FSTAD, which sets the origin address of his program to the vaue FSTAD. The address of the last location used must not exceed 4095 (7777 octal) and the system does not check this.

2.6 Light-pen

The EXEC provides no facilities for the light-pen except the escape UIESC (see section 2.4.3).

To initialize the light-pen execute:

```
IOT     )132    CLEAR     FLAG
LAW     3
IOT     )141    ENABLE    INTERRUPT
```

At interrupt level (each time the light-pen interrupts) execute:

```
IOT     )132    CLEAR     FLAG
IOT     )171    READ      X     POSITION
 :
 :
IOT     )172    READ      Y     POSITION
 :
 :
JMP  I  L$BAK
```

To exit, execute:

```
IOF             MASK      INTERRUPT
CALL    EXIT
```

The X and Y positions are encoded as described on page V-6-3 of the Imlac manual. After decoding, they are related to screen positions as follows:

Line N : Y = 3584 - 64N
Position M : X = 736 + 32M

2.7 Miscellaneous details

Some of the auto-index registers are used by EXEC and are therefore not available for other programs. Users may employ the four with octal addresses)4014 to)4017.

Note that octal locations)10010 to)10017 in TXBUF are also auto-index registers.

If an IBM program calls PDSTX with a control function specifying "binary file", the format of the LIST for PDSTX is slightly modified:

```
LIST    DC      /XYYZ           TRANSMISSION CONTROL WORD
        BSS     5
CFWC    DC      /QQOO+N         CF + WC
CA      DC      ADR1            CORE ADDRESS
SA      DC      ADR2            STARTING ADDRESS
        BSS     4
BUFR    BSS     N               BINARY FILE
```

In this case, the binary information is loaded into the PDS1 core at address ADR1. If the CF specifies "load into core and enter" the binary file is treated as a program and entered at address ADR2. If the CF specifies "load and chain" the PDS1 stays at interrupt level and awaits another file, sent by another call to PDSTX. (This call to PDSTX must not send an interrupt to the PDS1. Bit 13 of the transmission control word is set to 1 to suppress the interrupt.) If the CF specifies "load and return", SKF+1 and SKF+6 are set as described in section 2.4.2.

In a call to PDSTX, if bit 12 of the transmission control word is set, the transmission is forced (i.e. the following indicators are ignored: busy, broken, backed-up and no completion).

More detailed information on any part of EXEC should be obtained from the listings and flowcharts.

APPENDIX

CONTROL FUNCTIONS FOR FILES FROM IBM TO PDS1 (8 BITS)

00000000	Cold start program or force restart
00xxxxxx	Undefined (escape)
01xxxxxx	Alarm message
10	Binary file follows (for program loading):
01xxxx	Load into core, set SKF
11xxxx	Load into core, enter
00xxxx	Load into core, chain
1010xxxx	* Check-list data file (Hex./A000 or /A800 with cursor)
110	EBCDIC file
0xxxx	Load into buffer, (escape)
10000	*Display via EXEC (Hexadecimal /D000)
10100	*Display via EXEC, with initial delete (Hexadecimal /D400)
11000	*Display via EXEC, delete all graphics (Hexadecimal /D800)
11100	*Display via EXEC, with initial delete, and delete all graphics (Hexadecimal /DC00)
111	Graphical file
0xxxx	Load into buffer, escape
10000	*Display via EXEC (Hexadecimal /F000)
11000	*Display via EXEC, with initial delete (Hexadecimal /F800)

* Processed automatically by EXEC monitor

CONTROL FUNCTIONS FOR FILES FROM PDS1 TO IBM (8 BITS)

00000000	Not allowed
0xxxxxxx	Undefined (free for users to define)
10000000	System use (e.g. abort)
1010000	*Check-list data file (Hex./A000, or /A800 with cursor)
11xxxxx	Undefined

* Provided automatically by EXEC monitor

Control key allocations

↑ ↓ → ←	Cursor movement
CTRL/→ and CTRL/←	Check-list field selection
XMIT	Transmit
REPT	Repeat
CTRL/A	Abort
CTRL/B	Charge keyboards
DEL	*Delete ("pseudo-EBCDIC" value -1)
CR or LF	*Newline ("pseudo-EBCDIC"$)

The keyboard $ is converted into EBCDIC ¢, ' into ", ; into : and & into +.

* Not recognised by check-list handler.