XCRIBL
A Hardcopy Scan Line Graphics System for Document Generation*

R. Reddy, B. Broadley, L. Erman, J. Newcomer
G. Robertson and J. Wright

Computer Science Department
Carnegie -Mellon University
March 28, 1972

*This research was supported in part by Xerox Corporation and in part by the Advanced Research Projects Agency of the Department of Defense under contract no. F44620-70-C-0107 and monitored by Air Force Office of Scientific Research. We would like to thank Bill Gunning, Dave Damouth and Louis Mailloux of Xerox Corp. for their help and assistance.

ABSTRACT

XCRIBL is a system developed at CMU for generating hard copy computer output of arbitrary type fonts, graphics, and grey-scale images using a Xerox Graphic Printer (XGP). XCRIBL can be used to generate documents approaching the quality of printed text with the use of a document generation language (XOFF or PUB) and a character set design program (BILOS). Textual and graphic information to be printed is shipped in its raw form from the host computer (PDP-10) to a mini-computer (PDP-11) which acts as an intelligent channel controlling the XGP. Careful design of the data structures and the hardware interface permit the mini-computer to generate each scan line as needed without having to resort to a brute force solution of generating a bit-image for the whole page (3.5 million bits) for off-line printing. Variable width characters and the ability to mix text and graphics distinguish the present solution from the known simpler schemes for scan line generation.

XCRIBL, A Hardcopy Scan Line Graphics System for Document Generation*

INTRODUCTION

Several problems arise in computer science research in which conventional line printers and graphics terminals prove to be inadequate output devices. Digitized speech and image data are typical of these. They required either a display of arbitrary numbers of vectors flicker free or simulation of a grey scale output. The need for a computer output device capable of producing arbitrary type fonts, graphics, and grey scale images has been obvious to many researchers for some time. Earlier attempts at providing such flexibility in computer output has been largely abandoned because of the high processing and memory requirements or because of slowness of the devices. The XCRIBL system, developed at CMU, using a Xerox Graphic Printer and a PDP-11 represents an inexpensive solution to this problem. Careful design of data structures and the interface permit the PDP-11 to generate each scan line for the XGP as needed without having to resort to brute force solution.

The Xerox Graphic Printer is a facsimile copying machine originally designed for transmission of documents over high bandwidth telephone links. It has a resolution of 192 points per inch which is equivalent to a bit image of approximately 3.5 million bits for an 8 1/2 x 11 page. The paper moves at a rate of 1 inch per second or approximately 5 pages per minute, but because of its high resolution each page can contain the equivalent of two pages of conventional computer listing, which is approximately equal to a rate of 600 lines per minute.

The XGP works on the well-known Xerographic principle but instead of reflecting light off a printed page as in a copier, the XGP uses the image available on a CRT. The image on the CRT can be produced by facsimile transmission or under computer control. Each dot on the CRT is resolvable to 1/200 of an inch (5 mils). Each line drawn on the CRT is transferred to paper through an electrostatic process via a rotating selenium drum. The XGP is roll fed and can handle any thickness of paper and is thus less susceptible to paper jams. It does not require specially sensitized paper.

---

* As this was a group effort, it is appropriate to credit individual contributions. Initial design specifications and general approach were proposed by Reddy. Broadley and Wright developed the PDP-11/XGP interface. Robertson developed the critical real-time support system. Erman programmed the character set editing program and Newcomer modified the PDP-10 Runoff program. Many of the detailed hardware and software design decisions were made by the group as a whole. Clearly this design effort represents only a first approximation to the target system. We expect to modify and improved the system continously.

There are other hard copy devices which can be used for a computer output to achieve similar effects. Typical of these are the hard copy attachment to the Techtronix scope and the Gould electrostatic printer. Main considerations that govern the choice of one or the other of these devices are the dot resolution, the rate of output and the cost of the paper. Our present programs to produce a grey scale image on an ARDS storage scope display take 10-20 minutes to produce a 100 x 100 image. Attempts at providing computer driven high resolution video monitors face the problems of high data rates (100 megabits /sec) which cannot be conveniently handled on the most current computers. Very few electrostatic printers have the high resolution necessary to produce grey scale images and high quality printed output. The cost of paper for the XGP is about 1/3 of a cent per page in comparison to 1/2 to 1 cent for computer listing paper and specially sensitized paper. There have been several high quality photo compositions systems built (RCA Video Comp) but the cost of these devices is very high and they do not usually provide hard copy output. The XGP has been used earlier as a computer output device but many of the solutions were either brute force or did not provide the variable width text, or the mixed text and graphics ability.

A GENERAL SOLUTION

The following figure provides a data flow diagram in which the XGP is used as a printer for a PDP-10 with the PDP-11 mini-computer acting as the controller. The textual information shipped to the 11 is similar to the textual information shipped to any line printer. The graphical information is similar to the description shipped to most graphics terminals. The character set descriptions for various type fonts are stored on a small head-per-track disk connected to the PDP-11. In addition to textual and graphical information, the data from the PDP-10 may also contain special purpose control information such as change of type fonts, margins and other special formatting information.

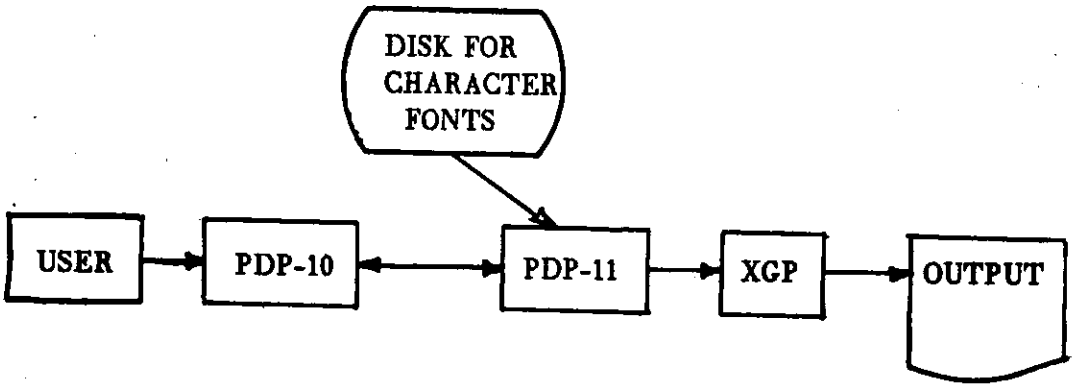


Figure 1. XCRIBL System: Data Flow Digram

Representation of chracters: There are several possibilities for representing character set descriptions. Special representations such as run coding and Huffman coding were rejected because of the increase in computational requirements. The requirement for variable width characters makes it difficult to use special purpose address computations as is common in character display terminals. Characters in our system are represented as a matrix with two parameters specifying the width and the height of the character. Each row of the matrix is filled with enough blank bits to make up an integral number of 8-bit bytes (which is the byte size of the PDP-11). A special mode in the interface permits the system to pick up the specified number of bits from the bit matrix based on the character width. At present, for any given character set, the height is fixed but this restricition is expected to be removed in the later systems.

Representation of Vectors: Vectors are represented in the conventional scan line format. The difference between a scan line representation and a conventional vector representation is illustrated in the following figure.

Conventional representation:

X1	Y1	X2	Y2

Scan line representation:

YTOP	YBOT	XTOP	∂X

where ∂X ← (X2-X1)/(Y2-Y1); *
if Y1<Y2 then
begin YTOP ← Y1 ; YBOT ← Y2 ; XBOT ← X1 end
else
begin YTOP ← Y2 ; YBOT ← Y1 ; XBOT ← X2 end;
* In practice there are problems of division by zero, round off, and position accuracy.

Figure 2. Vector Representation

In a conventional CRT, a vector can start at any point on the screen and move to any other point in a random manner. In video terminals and hard copy scan line output devices, such as the XGP, the data must be presented so that those vectors that intersect with a given scan line can be conveniently determined and the appropriate points of intersection in the scan line are marked. There have been several proposals for both hardware and software solutions to this problem but our requirement for flexible formating of output made it necessary for us to chose a

software solution. This limits the number of vectors that can cross a scan line but that restriction is not expected to be serious for most cases.

Representation of Grey Scale: Grey scale is achieved by dividing the page into 1/25 inch squares (an area of .0016 square inches) in which an appropriate number of bits is set to black to represent increasing darkness. This is achieved at present by using a rectangular spiral representation of increasing darkness. In practice, however, generation of grey scale images turns out to be a special case of textual output in which a special typefont of characters for grey scale is used. For this font the characters are represented as an 8 x 8 matrix. The following figure gives the representation for character '6' and the grey scale value of darkness 10.

```
0 0 0 1 1 0 0 0        0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0        0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0        0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0        0 0 1 1 1 1 0 0
0 1 1 1 1 1 0 0        0 0 1 1 1 0 0 0
0 1 0 0 0 0 1 0        0 0 1 1 1 0 0 0
0 1 0 0 0 0 1 0        0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0        0 0 0 0 0 0 0 0
```

Figure 3. Matrix represenataion of"6" and grey scale of 10.

Mixed text and Graphics: A page consisting of both textual and graphical information is described as two separate files on the PDP-10: one containing the textual part and the other the vector list. The generation of a scan line which contains both textual and graphical sections is not a problem for the PDP-11 if the text and the graphics can be assumed to be nonoverlapping. If the latter is not the case, then one has to resort to an off-line solution of generating the bit image on the PDP-10 or restricting the character set to only fixed width characters within the present system.

THE IMPLEMENTATION

In this section we will provide a description of the overall implementation of the system. More detailed descriptions of various aspects of the system can be found in the appendices.

The Interface

The main purpose of the interface is to accept a coded scan line from the PDP-11 memory and decode it to make up a scan line (bit vector of about 1550 points) every 5 ms. In the present solution, there is no hardware buffer for the whole scan line but rather the bits are generated as needed. The interface has facilities for handling three different modes of data and a means for switching between modes.

The modes are "character mode", "vector mode" and "image mode."

In the character mode the first byte represents the number of valid bits and subsequent bytes contain the data. Thus, if the width of a character is 21, the first byte contains 21 and the subsequent three bytes contain the data with the last byte containing only 5 bits of valid data. When the width count is zero, then the following byte represents a mode change (to either vector mode or image mode) or a stop code.

In the vector mode, each pair of bytes represents a run coding of (part of) the data. The first byte of a pair being the number of white points and the second byte is the number of black points. This mode is especially useful for compact representation of vectors and for representation of margins and blank lines. When two successive byte counts are zero, the mode reverts to character mode.

In the image mode, every byte is treated as data until an error condition occurs, such as overscan, at which point an interrupt is caused for re-start of next scan line. Appendix I gives a more detailed description of the interface.

The Support System

The purpose of the support system is to generate the scan line data needed by the XGP. In reality, however, it consists of several interacting device service routines with critical time constraints. In addition to the XGP, the support system has to service the interrupts from the 10-11 link, examine the incoming data for special control characters, and replace type fonts from the disk as needed.

On the average, the support system has 5 ms. to generate every scan line of data. Some lines are generated much faster than others. A large part of the system represents buffers of the input from the PDP-10 and the output buffers the XGP. At present, the program's size is 1K, with 15K being shared between the character set definitions and the buffers.

An interesting feature of the system is that every aspect of the output device now becomes a variable when compared with the conventional line printers. The character sets, the size of the buffers for output, margins, intercharacter spacing, interline spacing, and the size of the page are all variable for the XGP and can be changed under the control of the user.

The limitations of the PDP-11 as a bit manipulation device, and the difficulties involved in handling a real time device while at the same time servicing a high speed link and a fixed head disk place severe demands on the processing power available on a general purpose mini-computer like a PDP-11. We are currently investigating alternative solutions to this problem. Appendix II contains the details of the real-time support systems for running the XGP.

The Character Set Design System

BILOS is a system for creation and modification of character sets and has many facilities that are common to other interactive editing systems. a line of text is the basic primitive, characters being elements of this primitive. In BILOS the basic primitive is a boolean matrix whose elements may be either on or off.

BILOS has facilities for modifying any bit of this matrix. This is done by moving the cursor to an appropriate point in the grid on a display screen and changing the state of that element.

In addition, the system has conventional facilities for copying, substituting, translating, rotating, stretching, and shrinking of a character represented as a matrix. The system presently uses a storage scope display terminal connected to the PDP-10 Appendix III provides a description of the BILOS character set editing system.

A first approximation to a desired character set is obtained by digitizing that type-font using an image dissector, an image input device available at CMU. Smoothing algorithms reduce the noise of the digitized image. Final improvements to the character set are made manually using BILOS.

Document Generation Languages

We have two languages on the PDP-10 for generation of documents:, RUNOFF and PUB. Both are designed for output of a document on the line printer. While many of the facilities available in these systems are useful for generation of documents using XGP, several new features must be added to provide the user with all the flexibility available while using a XGP. After some initial attempts at modifying the existing systems, we now feel that a complete re-design and implementation of a document generation language is desirable.

A modified version of RUNOFF (XOFF) is now available as a temporary system for use with the XGP and its description is given in Appendix IV. Similar modifications are being made to PUB. Attempts at using both these systems in various document generation tasks is expected to provide us with the necessary experience to define a future system.

RESULTS AND CONCLUSIONS

Figures 4, 5, 6, and 7 provide examples of typical output from the XCRIBL system. While there is nothing new in each of the techniques used, the total system represents a bringing together of several technologies in both hardware and software to make such a system possible. There are several unsolved problems. Most of them deal with the flexibility provided by the system. Typical of these are sophisticated text justification algorithms, number of vectors that can cross a scan line at any given time, size and shape of the characters, and the dot resolution of the printer itself. The design choices were usually dictated by the computing power available in the controller. The present implementation cannot produce inclined text or even text requiring exchange of width and height of a page (such as when you try to produce breadth-wise text). In total, however, the flexibility, quality, and cost per page of computer output provided by the present system is substantially better than other devices available for computer output.

ACKNOWLEDGEMENTS

Several other members of CMU's community have helped to make this project a success. We would like to thank Mack Hicks for his editorial assistance in the preparation of this report; Bunny Kostkas who typed the manuscript; Jim Teter for his help in fabricating the interface; and Don Bihary, Keith Price, Larry Robinson, Leroy Richardson, Mary Shaw, Phil and Janis Karlton, and Ron Tugender for their help in constructing character sets.

XCRIBL SYSTEM
COMPUTER SCIENCE DEPARTMENT
CARNEGIE -MELLON UNIVERSITY

This is computer generated using XCRIBL system at Carnegie -Mellon University using a Xerox Graphic Printer. The XCRIBL system is capable of producing arbitrary *type font output*, graphics and grey scale images.

The Xerox Graphic Printer (XGP) works in a fashion similar to the Xerox office copiers. Instead of reflecting light off a printed page as in a copier the XGP uses a cathode ray tube similar to old television tubes as a source of light. The image is drawn as a series of dots on the CRT with a resolution of [illegible] dots per inch. The line of dots is reflected onto a selenium drum which electrostatically attracts a fine black powder to the exposed selenium areas. The powder is transfered to a moving sheet of paper. Finally a fuser melts the powder onto the paper.

To be able to print any character the pattern of dots which will be printed as that character must be entered into the computer along with an indicator of what the pattern represents. To facilitate this a program has been written to design character sets. This program draws a grid on a display termainal. Each box in the grid represents one dot in the final Xerox output. The dots may be set or unset and the character redrawn on the display as frequently as one might desire. Because of the ease with which this may be done it becomes an enjoyable task to design a character set and then be able to change any part of any character.

The XGP is also capable of drawing lines and gray scale images. The AI group is using the XGP to print pictures of faces and speech spectrograms. The range of possible uses is boundless.

↓∝ß∧¬∈πλ∞∂⊂⊃∩∪∀∃⊗⇔_→∩⊃∪≡∨?'
!" $%'()*+'-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[}←
'abcdefghijklmnopqrstuvwxyz{|}

↓∝ß∧¬∈πλ∞∂⊂⊃∩∪∀∃⊗⇔_→∩⊃∪≡∨?'
! $'( )'-.012345678:;?
ABCDEFGHIJKLMNOPQRSTUVWXYZ[
'abcdefghijklmnopqrstuvwxyz

Figure 4
Example of Textual Output

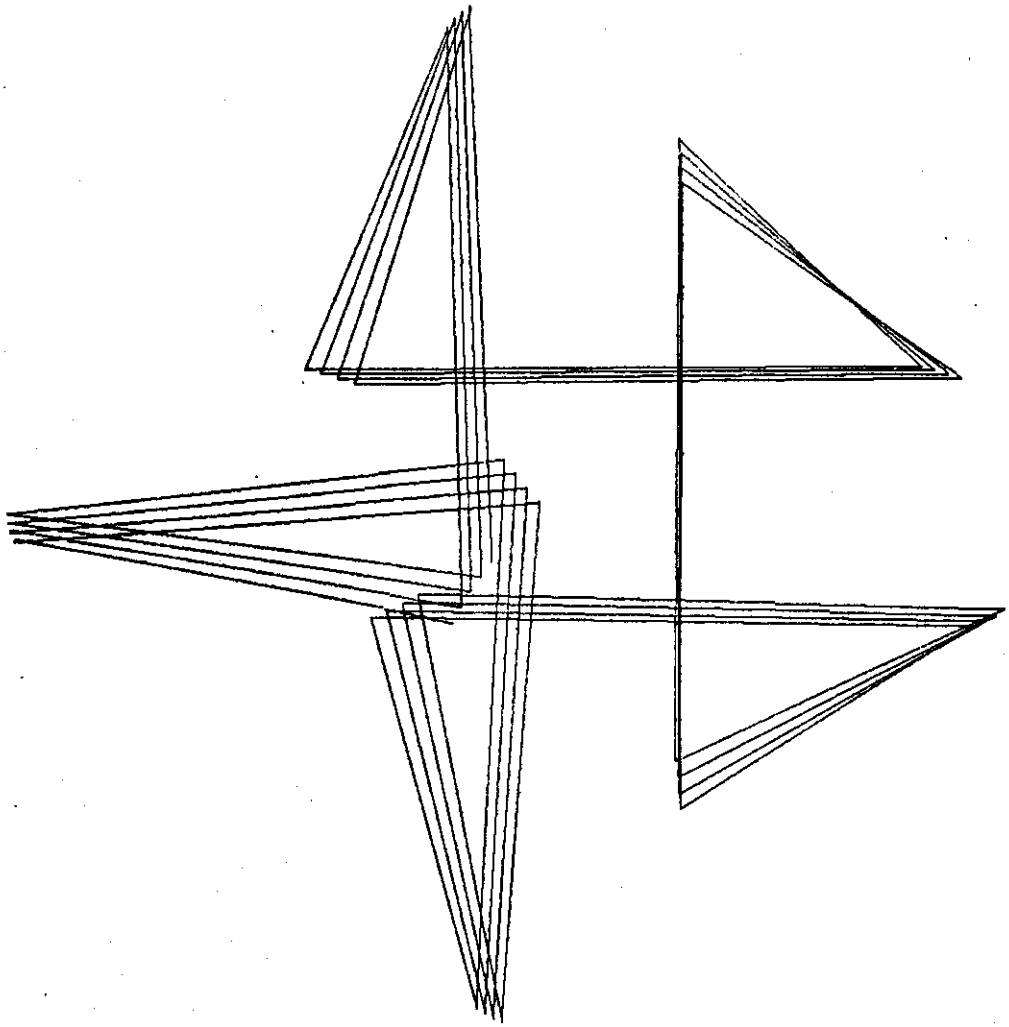


Figure 5. Example of Graphic Output

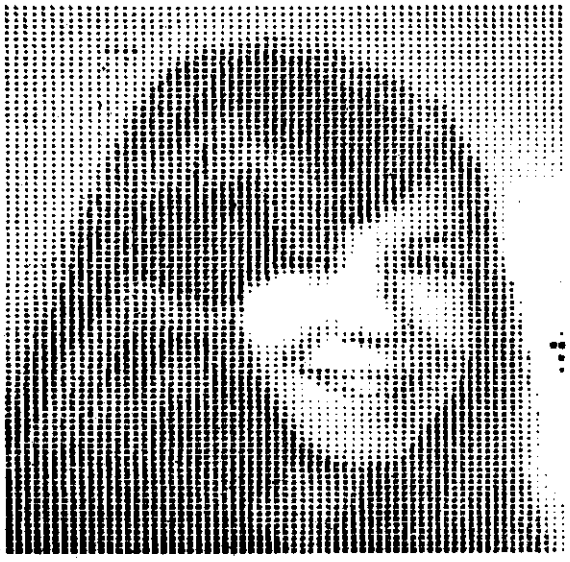


Figure 6. Example of a Grey Scale Image

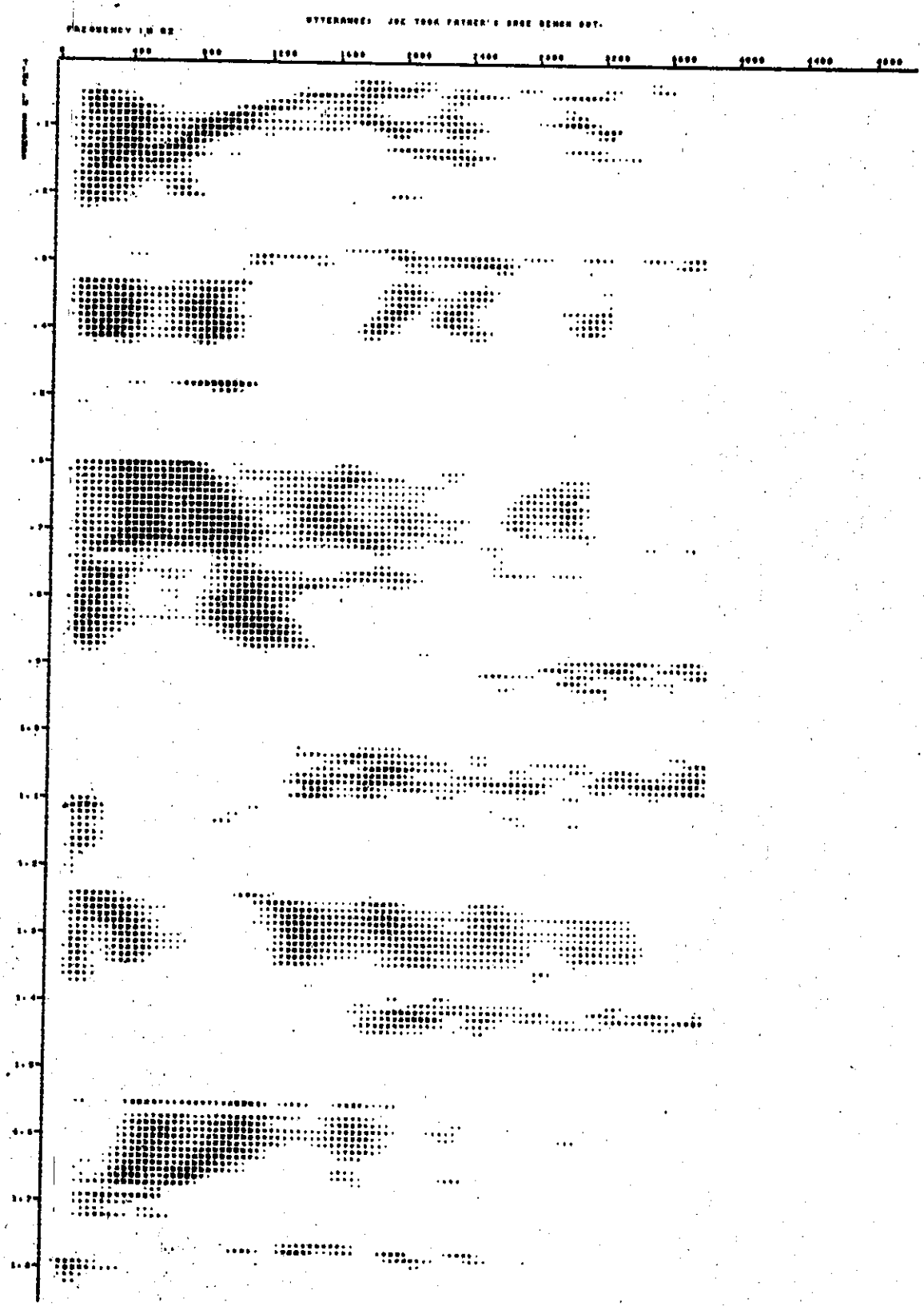


Figure 7. Example of a Speech Spectrogram

APPENDIX I

THE PDP-11 -- XGP INTERFACE

Bill Broadley and Jack Wright

I.1 INTRODUCTION

This appendix describes the interface hardware between the PDP-11 and the high speed hard copy graphics printer This Xerox Graphic Printer is similar in principle to a facsimile machine. Information is written (exposed) onto paper in a raster-scan pattern. A slow vertical scan in done by paper motion at a rate of 200 scan lines per second. A fast horizontal scan (video) is done at 3 us per scan line. Video information is two-state (black/white) only, no grey scale.

Resolution is variable (maximum 300) and is set to about 200 elements per inch at present. Format is 8 1/4" hriz. scan length on 8 1/2" paper with unlimited vertical extent. This gives a writing rate of approximately 300,000 points per second. The printer operates synchronously from page start to page end. Pages are terminated by a programmed "paper cut" command.

The interface provides facilities for data output from the PDP-11 Unibus to the printer video and for control functions. Several data modes are provided, as described below. The data rate is such that interrupt operation can be used, but at a high cost in efficiency; hence Direct Memory Access (channel) mode is provided. Data from the computer is initialized at the start of each horizontal scan, and is logically a stream of 8 bit bytes. These are converted to the video bit stream by the interface.

Figure 1 gives a sketch of the interface organization and the following sections provide the details of various data and control modes available for a user of the system.

I.2 REGISTERS

The LDX interface has 3 registers that are accessible from the UNIBUS (i.e., by program). These are:

Control and Status Register CSR 172100
(can be used as Source or Destination) 16 BITS
16 BITS
Memory Address Register MAR 172102

(can be used as source or destination) 15 BITS MA<01:15>

Special Note: Byte operations on these registers are illegal.

I.2.1 Control and Status Register (CSR)

Certain bits may change autonomously due to interface action; these are indicated below by R or S for reset and set respectively. The table below gives CSR bit designations, followed by explanation:

Control Status Register

BIT	HARDWARE SET/RESET	SOFTWARE SET/RESET	MNEMNOIC NAME
00	R	SR	GO - This is given to start data transfer
01	R	S	CUT PAPER
02			Not used
03		SR	MA EXTENTION - MA16
04		SR	MA EXTENTION - MA17 address extension on the Unibus
05			Not used
06		SR	DONE INTERRUPT ENABLE
07	S	SR	DONE
08	S	R	DMA OVERRUN
09	SR		ACTIVE
10		S	MOTION
11	SR		device ready
12	S	R	NXM - Non existent memory.
13	S	R	SYNC ERROR - next line started before GO
14	S	R	OVERSCAN - too much data
15	S	R	ERROR FLAG (8 V 11 V 12 V 13 V 14)

I.3 PAGE CONTROL

The LDX is essentially a synchronous device, and is intended to run continuously (non-stop paper motion). Gross control is provided by:

CSR10 - MOTION BIT
When set, the printer run signal is asserted, causing the paper to run.
When reset, motion begins to shutdown.

NOTE: When starting, a lay delay (approx. 2 sec.) is needed to allow startup before the first scan is output. This delay must be program generated.

Data is written on continuous roll paper and cut into sheets by program command.
CSR01 # CUT PAPER

When set, causes a "cut mark" to be written. This bit will reset itself after 60 us.

NOTE: The cut mark is a black dash written in the margin. When it reaches the cutter, a sensor causes the cutter to operate. The cut should then be just after the last line written before the mark command occured.

NOTE: Like other Xerox devices, a large amount of paper is in process between writing and stacking. Three feet of paper is automatically run out by hardware.

I,4 LINE CONTROL

In terms of program interaction, the major cycle of the LDX control is the scan line sequence. Program action is required to initiate each line (by interrupt service). This is done by loading both the MA and CSR Registers.

I.5 SCAN TERMINATION

Scan termination is the change from Active to Done states. It normally occurs when a stop code occurs in the data stream (character mode, count zero, control byte = 001).

Any error condition will also cause a scan termination.

I.6 INTERRUPTS

The interface has one interrupt vector location (370) for the DONE condition. All error conditions also set the DONE condition.

I.7 DATA CONTROL

Normally, data is accessed by the interface from a core block on direct Memory Access (NPR) basis. At the initialization of each scan line (before setting GO) the initial core address is specified by loading the interface memory address register (MAR) (address 772102). the first data byte is taken from an even location. bit 00 is ignored. There is no word count register (see "Scan termination").

Due to timing constraints, no provision is made for linking multiple buffer blocks.

I.8 DATA MODES

There are three data modes; character, vector and image. Each scan begins in character mode.

I.8.1 Character mode

In this mode the first byte contains a count; each following byte contains 8 data points:

A byte count of 0 indicates that a single control byte follows. A byte count of 1 through 255 specifies the number of following video bits until the next count byte. The count is represented in true positive form. Each video byte specifies eight resolution elements; 1 = black, 0 = white; BIT 00 is used first.

Control byte - the single byte following a zero count byte in character mode is interpreted as follows:

000 - change to vector mode
001 - terminate scan (sets DONE)
002 - change to image mode
003 - 377 - undefined

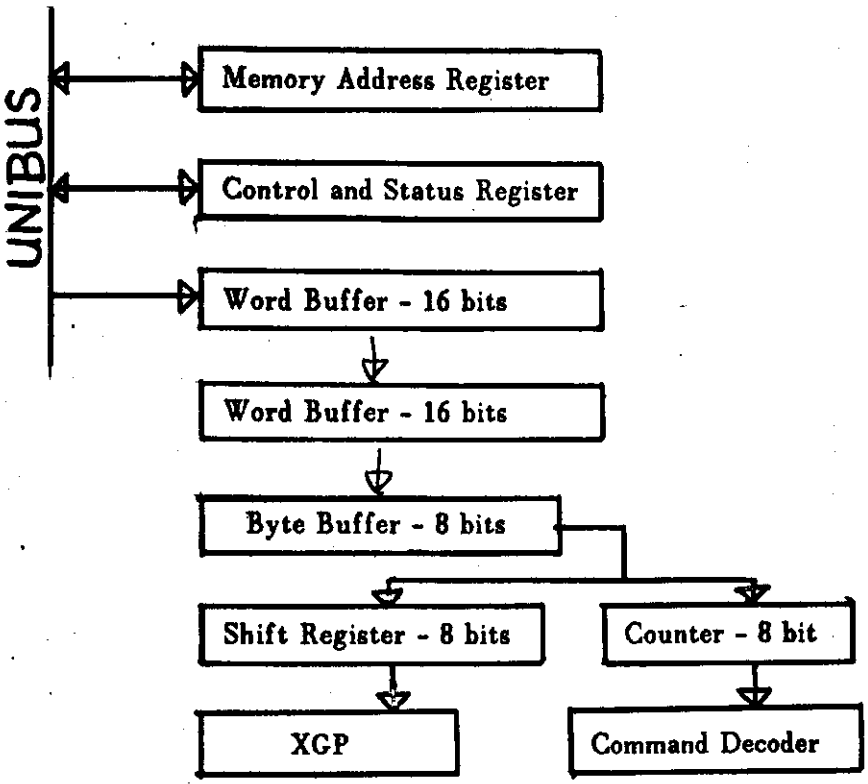
I.8.2 Vector Mode

In this mode, each byte specifies a number of consequtive points of the same same color (black or white). Permissible values are 0 through 255 (true positive form). Color alternates as with each new byte. White is the starting color. To obtain a vector which is longer than 255 points, alternating bytes of zero (for the undesired color) must be used.

Two consequtive bytes of zero cause a change to character mode.

IMAGE MODE

Image Mode is similar to character mode except that there is no count bytes. All bytes after the control byte are data bytes. The only termination in image mode is an error condition (normally OVERSCAN).



Appendix II

The Support System*

George Robertson

II.1 INTRODUCTION

The XGP support system consists of two sub-systems: the LOOK program on the PDP-10 and the LISTEN program on the PDP-11. The purpose of the LOOK program is to accept data from a file or another program in 7-bit ASCII code format of the PDP-10 and translate it to the 8-bit format. In addition, it performs a large number of control functions. The purpose of the LISTEN program is to accept the data from the PDP-10 and to convert it into the scan line format needed by the XGP. In addition, it contains different service routines for handling the XGP, 10-11 link, and the PDP-11 disk. At present both LOOK and LISTEN have to be manually initiated. Facilities in the PDP-10 monitor will be available to activate the system from a remote terminal.

II.2 THE PDP-10 XGP SYSTEM

LOOK is a PDP-10 program for communicating with the PDP-11 and the LDX. LOOK was designed to appear similar to PRINT, except that it has more flexiblity. Currently, LOOK lives as LOOK.SAV[A730GR02].

To run LOOK, issue the monitor command:

```
.RUN DSKB LOOK[A730GR02]
```

II.2.1 Input State

When LOOK is run, it goes into its input state. In input state, the following commands (each terminated by a CR) are legal.

<file>	Use this file for whatever mode is in force (see mode descriptions below).
=<var>	Examine the variable.
=	Display all system variables.

←<var>=<val> Set variable to value.

←<var>= Display alternatives for assignment for this variable.

II.2.2 Variables

The variables which can be set and examined (with their defaults and possible settings) are listed here.

AKSET - 0 - (INCORE, CLEAR, a character set index between 0 and 63) This variable specifies how the "A" partition character set is to be initialized.

BKSET - CLEAR - (INCORE, CLEAR, -1 < KSET < 64) This parameter specifies how the "B" partition character set is to be initialized.

VERT.SPACING - 8 - (non-negative integer) Specifies the spacing between text lines.

LFTMAR - 200 - (non-negative integer) Specifies the left margin in number of bits. (i.e. default is one inch).

TOPMAR - 128 - (non-negative integer) Specifies top margin in number of bits.

BOTMAR - 128 - (non-negative integer) Specifies bottom margin in number of bits.

NLINES - 55 - (positive integer) Specifies number of text lines on a page.

SLINES - 2000 - (positive integer) Specifies number of scan lines on a graph (Vector mode only).

CUT - AUTOMATIC - (MANUAL, AUTOMATIC) Specifies whether automatic page cutting is desired. The beginning of a file (i.e. first page) always gets an initial cut.

LINENUMS - YES - (YES, NO) Specifies whether line numbers if they exist in the file should be sent or not.

JUSTIFY - NO - (YES, NO) Specifies whether or not justification is to be in effect.

JWIDTH - 0 - (non-negative integer) Specifies the number of bits (not counting the left margin) that the user wants on each line. Only lines shorter than this width are justified. Only blanks are justified. The longest blank which will be generated is 255 bits (that is over one inch). This only has effect when JUSTIFY is YES.

JMAX - 0 - (non-negative integer) Specifies the maximum number of bits the justifier is allowed to add to a line. If more than this number of bits are needed, then the line is left unjustified.

LOCKUP - NO - (YES, NO) Specifies whether or not core lockup should be used in real-time modes. Note that lockup cannot be used unless you are running under a project with the lockup priviledge.

MODE - TEXT - (TEXT, VECTOR, INAKSET, INBKSET, OUTAKSET, OUTBKSET, MIXED, IMAGE) Specifies how a file is to be handled. See below for a complete description of the different modes.

II.2.3 Modes

The following modes of operation determine how a user file is to be handled. The mode also determines how communication with the PDP-11 will be handled.

This subsection describes what LOOK sends to the PDP-11 for each mode the user might invoke. It also presents user conventions for file formats.

1. TEXT mode sends the current parameter settings for AKSET, BKSET, VERT.SPACING, LFTMAR, TOPMAR, BOTMAR, NLINES, CUT, and JWIDTH and JMAX if JUSTIFY is set to YES. Then text lines are sent from the specified file, one line at a time, with or without line numbers as specified by LINENUMS.

   Once the parameters have been specified, the text mode deals only with text. It prints text lines (a string of characters terminated by LF) until an EOF character is seen (see escape conventions below). Null characters (octal code 0) are always ignored.

   The escape character is the rubout (octal code 177). The

character following a rubout is interpreted as an escape code. The escape codes (in octal) along with their meanings are listed below.

Code	Meaning
0	End of file
1	Vertical spacing in next two characters
2	Left margin in next two characters
3	Top margin (2 characters)
4	Bottom margin (2 characters)
5	Number of text lines per page (2 characters)
6	Automatic cut
7	Manual cut
10	A partition character set (1 character)
11	B partition character set (1 character)
12	Wait for A character set
13	Wait for B character set
14	Change to A partition character set
15	Change to B partition character set
16	Justify width (2 characters, 0 => no justification)
17	Maximum padding for justification (2 characters)
20	Variable size blank (width of blank , 1 character)
21	Change to image mode
22	Character update
23	Line-feed
24	Form-feed
25	End control line
26	Begin control line
27	Cut immediate
30	Tab to raster position (2 characters)
177	Rubout character

A control line is defined to be a set of escape codes between a begin-control-line and an end-control-line. No text may appear in a control line or between the last line-feed and the begin-control-line. Cut immediate and form-feed are treated as line terminators, but any text on a line with one of these commands is ignored. Tab to raster position and variable sized blank are in effect only when in a non-justify mode.

2. VECTOR mode is a graphics mode. The current setting of SLINES is sent first. Then the specified file is read, sorted, re-formated, and finally sent for printing. The format of the file should be <N, v1, v2, ... , vN> where N is simply the number of vectors. Each vector in the file should be in the form <x1, delta x, y1, delta y> where (x1,y1) is the starting point, delta x is the total change in the x direction, and delta y is the total change in the y direction. The file should be a binary

file with each number in a seperate word. Note that delta x is the only item in a vector which can be negative. Also note the following ranges of item values: 0 < x1 < 1550, 0 < x1 + delta x < 1550, y1 non-negative, delta y non-negative.

Since the user may wish to maintain his vectors in a text file for editing purposes, a conversion program is provided for coverting from a text file to a binary file. The program is VECNVT.SAV[A730GR02]. The text file should contain text lines with one vector per line. The vector format is the same (i.e. <x1, delta x, y1, delta y>) except that a single blank should seperate each item in the vector description.

3. INAKSET mode allows the user to retreive a character set from the A partition of the PDP-11.

4. INBKSET mode allows the user to retreive a character set from the B partition of the PDP-11.

5. OUTAKSET mode allows the user to send a character set or updates to a character set to the A partition of the PDP-11. When the specified file name is looked up, a search is first done of your own disk area. If the file is not found, [A730KS00] is searched. In other words, if you are using any of the standard character sets, you need not specify [A730KS00] as the disk area.

6. OUTBKSET mode allows the user to send a character set or updates to a character set to the B partition of the PDP-11. See the comments on OUTAKSET.

7. MIXED mode allows the user to send mixed text and vectors. The character sets used in this mode must be fixed width with the character width being a multiple of eight (8).

8. IMAGE mode is an off-line mode. The information on the file is presumed to be formatted scan lines and is passed directly to the XGP with no interpretation.

II.3 THE PDP-11 XGP USER SUPPORT SYSTEM

The XGP user support package is a PDP-11 system which handles communication between the PDP-10 and the LDX. The system has six modes of operation which allow for total access to the LDX. Those modes are briefly described here.

The gross structure of the system is intended to allow hierarchical access to the LDX. There are essentially three levels of access. The first level, which is the most restrictive, allows either text processing or vector processing, but not both (Text mode and Vector mode). These two modes are real time modes. The second level allows mixed text and vectors, but has a restriction on the type of character sets used for the text (Mixed mode). The third level allows arbitrary access, but is not done in real time (Image mode).

This section discusses internal formats, conventions, data flow and control flow. It is presented primarily for completeness. The user need not be concerned with the details of this section.

II.3.1 Text Mode

In text mode, the user specifies two 128-character character sets which he wants to use. These character sets may be empty, in core, or on the PDP-11 disk. Switching between character sets is legal within a text line only if the two sets have the same height. A third set may be introduced between text lines (not yet implemented).

The user also specifies top, left, and bottom margins, the number of text lines per page, whether or not automatic page cutting is desired, vertical spacing (i.e. between text lines), and justification information. Justification of variable width characters is done if desired. The user specifies the width of the justified line and the maximum number of points he will allow for padding. The justification is done by expanding blanks within the text line. Any or all of these parameters can be changed from within the text.

Internal formats for control codes are somewhat different from those found in section II.2.3.

II.3.2 Send-Character-Set Mode

In send-character-set mode, the PDP-10 is sending a character set to the PDP-11. The user first specifies how core is to be set up on the PDP-11 before anything is sent. He specifies that core is to be cleared if he is going to ship over a new character set. If he is sending updates to an existing character set, he specifies a character set on the PDP-11 disk or one that is already in core.

A description of the representation for character sets on the

PDP-10 can be found in the documentation for BILOS (see BILOS.DOC[A620LE03]).skip 2

II.3.3 Receive-Character-Set Mode - A Partition

In receive-character-set mode, the PDP-11 is sending a character set to the A partition of the PDP-10. The user specifies what to be sent by specifying how the A partition character setshould be set up (i.e. cleared, in core, on the disk). The character set representation is the same as for send character set mode.

II.3.4 Receive-Character-Set Mode - B Partition

This mode is similar to the mode for the a partition.

II.3.5 Vector Mode

Vector mode takes as input a set of vectors for a page along with the number of scan lines desired for the page. After the vectors are stored in core, a graph is produced in real time. The origin of the graph is in the upper left-hand corner. The positive X-axis is horizontal to the right. The positive Y-axis is vertical down. The vectors are expected to be sorted on their Y coordinates. The form of each vector is <x1, y1, [slope], fraction part of slope, # scan lines> where (x1,y1) is the starting point, [slope] is the greatest integer in the slope, and # scan lines reflects the number of scan lines which this vector will pass through.

II.3.6 Mixed Mode

Not implemented yet.

II.3.7 Image Mode

Not implemented yet.

II.4 OPERATING PROCEDURES - XGP AND PDP-11

II.4.1 Start-up Procedures

1. If the PDP-11 is off, use the key on its console to turn it on.

2. If the XGP is off, press its "ON" button. If that doesn't work, press the "POWER" button on the remote control unit on top of the LDX. The XGP will go into a "STANDBY" state for about 15 minutes while the fuser warms up.

3. Find and mount the dectape labelled "LDX SYSTEM #1". Dial the dectape to unit 0. MAKE SURE THAT THE UNIT IS WRITE LOCKED!!

4. On the PDP-11 console, press HALT, START, set the switch register to 773000, LOAD ADDR, ENABLE, and START. The system will be read from the dectape.

5. While the system is loading, one of two error messages might be typed on the PDP-11 TTY.

   a. "LDX NOT READY" - The system will loop until the XGP goes into its "READY" state.

   b. "DISK CLOBBER! MOUNT BACKUP TAPE AND HIT CONTINUE" - A checksum error was found on the disk directory. Find and mount the dectape labelled "LDX DISK BACKUP" on unit 1. Make sure that it is write locked! Hit CONTINUE.

   When the system is satisfied with the XGP and the disk, it will type "TYPE H FOR HELP".

II.4.2 XGP Operator Commands

The following single letter commands may be typed at the PDP-11 TTY. The command descriptions follow.

A - Abort current listing. Will terminate the current operation, send an error code to the PDP-10, and try a soft restart.

B - Bell on error. This is a toggle. In the on position, it causes a bell to ring whenever an error condition occurs.

D - Disk. This allows the operator to perform disk operations which are illegal from the PDP-10. The current character sets will be written if "C" is typed after the prompt. The disk will be backed up on dectape unit 1 if "T" is typed. The disk directory will be cleared if "Z" is typed. The Disk ("D") command will only work when the system is in its idle state. At other times it is ignored.

H - Help. A text of command descriptions will be typed if the system

is in its idle state. Otherwise, the help command is ignored.

M - Message on errors. This is a toggle. When it is in its on position, error messages will be typed on the PDP-11 TTY when any error occurs.

R - Restart. This is a hard restart. It makes no attempt to communicate with the PDP-10.

S - Stop on errors. This is a toggle. In its on position, it causes the PDP-11 to halt whenever any error condition occurs.

II.4.3 Shut-down Procedures

If someone else is likely to want to use the LDX, don't shut it down. However, if no one else is around, the following procedures should be followed.

1. Power down the XGP by pressing its "OFF" button. Don't use its "POWER" button for this purpose.

2. Dismount all tapes that were used and put them back where you found them.

3. Power down the PDP-11 by pressing HALT and turning the console key to the off position.

4. Clean up whatever mess you and the XGP have left!!

APPENDIX III

BILOS - A Program for Designing Character Sets

Lee Erman and Ron Tugender
Carnegie-Mellon University

BILOS (pronounced "BY-LOHS") is a PDP-10 program for creating and editing character sets (i.e. rectangular bit arrays) for use with the XCRIBL document generation system at CMU. A further description of the XCRIBLsystem can be found in LDX.DOC[A730GR02].

At present, BILOS uses the ARDS display (in the future, BILOS will be adapted for use with the CMU graphic processors).

1. INTRODUCTION

Characters are specified by their octal codes: 1 through 177 are legal. Character codes are always specified in octal --all other numbers are decimal. A character set (abbreviated hereafter as "kst") can have any (or all) of its 127 possible codes used.

Each character is rectangular, and the height of all characters in a kst is a constant for that kst in the range of 1 to 2↑15 - 1 (The height may become variable in the future). The width of a character is in the range of 1 to 255, and widths may vary from character to character in a kst. (Actually, BILOS can handle characters of arbitrary width; it is the XCRIBL system which has the 255 maximum. BILOS can warn if a character is wider than 255 -- see the WBIGTELL variable, below.) A width of 0 is equivalent to non-existence for that character.

BILOS keeps in core the entire kst that it is editing. In general, a character in BILOS is similar to a line in SOS, the text editing system available on the PDP-10. There are in fact many resemblences between BILOS and SOS. Those features of BILOS that correspond to SOS are denoted in this manual by "(SOS)". Familiarity with SOS is, however, not required.

2. START-UP

BILOS, when run, first prompts for a file name. If no name is given, then BILOS requests the (fixed) height for the kst to be created. If a file name is given (and the file is found on the disk), the height of the kst (as it exists on the file) is reported,

and a prompt is given to type in a new height. If no height is typed (i.e. just a <cr> is typed), the height is not changed. If the new typed-in height is less than the height in the file, each character is truncated from the bottom. If the height is greater than that in the file, each character is padded at the bottom with lines of blank points.

Following start-up, BILOS enters "command mode", which it signals by typing a "*" on the user's TTY (SOS). From that point on, BILOS can be in either command mode or "alter mode" (SOS).

3. COMMAND MODE

3.1 BNF

The following describes the BNF used in the various commands below:

```
<range>  ::= <char> / <char> : <char>

<char>   ::= <ascii-code>  / <ascii-code> + <decimalinteger>
                           / <ascii-code> - <decimalinteger>

<ascii-code> ::= . / <octal-code>

<octal-code> ::= 1 / 2 / 3 / 4 / 5 / 6 / 7 / 10
                           / 11 / ... / 177

<file>  ::= <PDP-10  file.ext[PPN]>
```

The semantics of the above are as follows:

If a <range> is a single <char>, then the command affects only the character corresponding to that ASCII code. If the second form of <range> is used, then all currently defined characters between those two ASCII codes will be affected.

If <char> is specified as an <ascii-code> + (or -) an integer N, the octal code used for that <char> is the Nth defined character following (or preceding) the <ascii-code>, as long as that is within the bounds of the defined set of characters (SOS).

The value of "." (pronounced "dot") is defined to be the "current" character, i.e. the one last considered in time. For example, after D13:137, "." would be 137; after C12,21 "." would be 12.

3.2 Actions Available in Command Mode

In command mode, the user can perform various administrative actions. Basically, these actions fall into three classes: file update and session termination, character housekeeping and displaying, and examination and setting of variables. In the following descriptions, all commands are terminated by a <cr>. Those parts of commands enclosed in square brackets are optional.

3.2.1 File Update and Session Termination

3.2.1.1 E[<file>]

Write out the file on the disk with the name <file> (if <file> is omitted, BILOS will use the current file name. If no file name exists yet, BILOS will ask for one) and end the edit (SOS).

3.2.1.2 W[<file>]

Write out the file as above but do not end the edit (to preserve partial editing) (SOS).

3.2.2 Character Housekeeping and Displaying

3.2.2.1 R<file>[,<range>]

Open <file>. Modify the current kst by replacing each character in <range> by the corresponding character in <file>. If <range> is omitted, then 1:177 is used (i.e. the entire current kst will be replaced by the contents of <file>).

3.2.2.2 D<range>

Delete all existing characters in <range> (SOS).

3.2.2.3 C<to-char>,<from-char>

Copy the contents of <from-char> into <to-char>. If <to-char> already exists, BILOS will print a message to that effect and not modify <to-char> (i.e. if <to-char> exists, it must first be deleted) (SOS).

3.2.2.4 T<to-char>,<from-char>

Perform a C<to-char>,<from-char> followed

by a D<from-char>. (If <to-char> already exists, nothing will happen) (SOS).

3.2.2.5 A<range>

Go into alter mode for the characters in <range> (SOS). (see description of alter mode below)

3.2.2.6 L[<range>]

List on the line printer all defined characters in <range>, under the name BILOS.LST. BILOS will prompt for a listing title, an arbitrary string of characters which is typed in and terminated by a <cr>. The variables "LCHAR" and "LBLANK" (defaulted to "X" and <space>) are used to represent the points and background respectively. If <range> is omitted, 1:177 is used.

3.2.2.7 SC<range>

The width will be scaled by a factor of the current value of the variable "WSCALE"; the height will be scaled by a factor of the current value of "HSCALE". The width of the character is adjusted appropriately, according to "WSCALE". The height is unchanged -- truncation or padding on the bottom is done if "HSCALE" is greater than or less than 1.0, respectively.

3.2.2.8 P[<range>]

Type on the user's terminal the heights and widths of all defined characters in <range>. If <range> is omitted, 1:177 is used.

3.2.3 Variable Examination and Setting

Variables are used in BILOS to control various actions. In general, a variable can be read or set and has an initial default value (except as noted). When typing the name of a <variable>, only enough of the name is needed to disambiguate it from the other names.

3.2.3.1 Variable Manipulation Commands

3.2.3.1.1 =

List the names of all the variables.

3.2.3.1.2 ←<variable>=<value>

Set the value of <variable> to <value> (SOS). If a <variable> is set to null, it is reset to its default.

3.2.3.1.3 =<variable>

Type the current value of <variable> (SOS).

3.2.3.2 Variables (defaults are listed in square brackets)

3.2.3.2.1 FILE [input file]

The current file name. This file will be used for the next E or W command if no explicit file name is given in the command (SOS).

3.2.3.2.2 VERSION

BILOS' version number.

3.2.3.2.3 WBIGTELL [YES]

If "YES", then warn the user when writing out the kst if there is a character wider than 255 (which is the widest that the XCRIBL system accepts). If "NO", don't warn.

3.2.3.2.4 WMAX [whatever is appropriate]

The width of the widest current character - cannot be set.

3.2.3.2.5 HEIGHT [kst height after start-up]

Height of the kst.

3.2.3.2.6 BUMP [5]

The number of points the cursor will move on big jumps (<>+?) in alter mode.

3.2.3.2.7 . ("dot") [0]
The "current" character (SOS).

3.2.3.2.8 LCHAR ["X"]
The character used to represent set points when the kst is listed on the line printer.

3.2.3.2.9 LBLANK [<space>]
The background character for listings on the line printer.

3.2.3.2.10 LWIDTH [130]
Length of line for listing on LPT.

3.2.3.2.11 KSTID [0]
Not used at present.

3.2.3.2.12 WSCALE [1.0]
The scaling factor of the width for the SCALE command.

3.2.3.2.13 HSCALE [1.0]
The scaling factor of the height for the SCALE command.

3.2.3.2.14 DISPLAY [1]
The default display format (used when no explicit display format is used -- see alter mode).

3.2.3.2.15 FORMAT[<digit>]

Set or examine the value of format number <digit>. FORMAT is used to set the variables controlling the output format of characters when displayed in alter mode (see Alter Mode). There are 9 possible formats, with digit codes 1-9. A format consists of two values: a scale and a display caracter. The scale determines the number of ARDS points between character points in displaying. Its default is 11. The display character is the character which prints on the ARDS for filled points. Its default is the "delete"

character, which prints on the ARDS as a solid box.

When setting the display format, the scale is given first, followed by a comma, followed by the display character. If either is omitted in a set command, its value is set to that of the default.

e.g. ←F04=15,@ sets format 4's scale to 15 and display character to "@".

←F05=, sets format 5's scale to 11 and display character to "delete" character.

←F06= sets format 6's scale to 11 and display character to "delete" character.

When examining, if a <digit> is given (e.g. =F03) only the values of scale and display character for that format is printed. If no <digit> is given (i.e. =F0), all currently defined formats are listed. The current default format (controlled by the variable "DISPLAY") is appropriately marked.

For convenience, formats 1, 2, and 3 are predefined. (They can, however, be redefined by the user).

FORMAT	SCALE	DISPLAY CHAR
1	11	"delete"
2	7	"*"
3	3	"."

3.2.3.2.16 ARDS ["YES" if ARDS is user's TTY, else "NO"]
Returns "yes" if ARDS is assigned to user, "no" otherwise. If set to "YES", BILOS will try to get the ARDS; if set to "NO", BILOS will release the ARDS.

4. ALTER MODE

In alter mode, each character in <range> is, in turn, copied into a scratch buffer (SOS), where the user can then display the character, as well as modify it.

4.1 Start-up in Alter Mode

When entering alter mode for a character, BILOS first types out the character code and the current width of the character (a width of 0 means the character is not currently defined), then prompts for a new width.

a) A reply of <cr> leaves the width unchanged and continues into edit mode for the character.

b) A positive integer response changes the width to that value, truncating or padding with zeroes on the right as necessary, and goes on into edit mode.

c) A reply of <alt-mode> <cr> leaves the current character unchanged, and goes on to the next character in <range>.

d) A response of E <cr> returns to command mode, with no change to the current character.

4.2 Displaying a Character in Alter Mode

A character is displayed on the ARDS by printing one point on the ARDS for each point in the character. The appearance of the character is controlled by the display format. Each display format represents two quantities. The scale factor of the format determines how far apart the points are placed on the ARDS. The display character is the character which prints for those points which are set in the character (unset points print as blank). Three formats are predefined for the user (as explained in the description of the "FORMAT" variable in the section on command mode). Which format is used in displaying the character is determined by the "DISPLAY" variable and the "D" command in alter mode (see below). Any format can be redefined by the user (in command mode).

4.3 Editing a Character in Alter Mode

After a non-terminating response (i.e. other than <alt-mode> <cr> or E <cr>) has been entered to the prompt for the width, BILOS is ready to edit the character. BILOS clears the screen, draws the character on the ARDS using the current default format (see "D" command below), prints the character code, current character width,

and "MODE = N" at the top of the screen, and positions the ARDS cursor at the top left point of the character. BILOS is then ready to accept alter mode commands.

4.3.1 Editing States

When editing a character, BILOS is in one of three states: N (neutral), M (mark), or X (unmark). These states affect the action BILOS takes when the cursor is moved. In "N" state, the points to which the cursor moves are not affected. In "M" state, the cursor marks (turns on) each point it visits, while in "X" state, the cursor unmarks everything it touches.

The user is informed of the current state by the "MODE =" at the top of the screen. The rightmost letter following the "MODE =" denotes the current state. BILOS is initialized to the "N" state when the edit is started, and after each "D" command in alter mode. The state is changed by the "N", "M", and "X" commands (see below).

4.3.2 Editing Commands in Alter Mode

Each command is entered by hitting the appropriate key. Commands are not echoed. Letter commands may be entered in upper or lower case. This, of course, is not true with special characters.

4.3.2.1 <space>

Set (fill in) the point where the cursor is. Echoes as the display character of the last "D" command.

4.3.2.2 U

Unset (erase) the point where the cursor is. If the point was set, echoes a "/" at that position.

4.3.2.3 .

Move the cursor one point to the right (For this, as for all cursor movement commands, the effect on the point(s) to which the cursor moves is determined by the current state of the editor. See section 4.3.1 Editing states.) The cursor always remains within the confines of the character's rectangle.

4.3.2.4 >

Move the cursor to the right a number of points equal to the current value of the "BUMP" variable (initially 5). All points between the original position of the cursor and the position to which it moves are affected by the editor state.

4.3.2.5 ,

Move the cursor one point to the left.

4.3.2.6 <

Move the cursor "BUMP" points to the left.

4.3.2.7 ;

Move the cursor one point up.

4.3.2.8 +

Move the cursor "BUMP" points up.

4.3.2.9 /

Move the cursor one point down.

4.3.2.10 ?

Move the cursor "BUMP" points down.

4.3.2.11 C

Clear the screen.

4.3.2.12 [<digit>]D

Draw the character as it currently appears in the editing buffer. If only "D" is issued, the format used will be the default format, which is the value of the "DISPLAY" variable (set in command mode). If a <digit> precedes the "D", then format <digit> is used. If this format is not currently defined, the message "?FORMAT NOT DEFINED" is printed.

4.3.2.13 B

Draw a box on the screen around the character with width and height appropriate to those of the current character, using the scale of the format used in the last "D" command.

4.3.2.14 G

Draw a grid on the screen over the character, with each square representing a point, and dimensions same as "B" command.

4.3.5.15 <cr>

Update the character as now altered in the scratch buffer, and leave alter mode for the current character (SOS).

4.3.2.16 W

Update the character as now altered in the scratch buffer, but do not leave alter mode. (Note that "W" in alter mode updates the current character in core; "W" in command mode writes out the entire file on the disk.) (SOS does not have the "W" command in alter mode.)

4.3.2.17 Q

Leave alter mode, but do not update the character as altered (SOS). However, if one or more "W" commands have been issued in alter mode, the character is left in the state it was in after the last "W" command.

4.3.2.18 N

Go to "N" state. (see 4.3.1)

4.3.2.19 M

Go to "M" state. (see 4.3.1)

4.3.2.20 X

Go to "X" state. (see 4.3.1)

4.3.2.21 S. (S, S; S/)

Within the character's grid, shift all the bits one unit right (left, up, down). Bits shifted off are lost, bits added are zeroes.

4.3.2.22 ←

Delete the column that the cursor is in (this command has the same effect for any point in the column). Shift all columns right of the cursor one unit to the left, shifting in a column of zeroes on the

right.

4.3.2.23 ↑

Delete the row that the cursor is in, and shift the rows below the cursor up one unit, filling in with zeroes at the bottom.

Appendix IV
XOFF: A System for Generation of Documents

R. Clements, Digital Equipment Corporation
J. Newcomer, Carnegie Mellon University

This document describes a program for document generation, XOFF. This system is based on RUNOFF, a program distributed by Digital Equipment Corporation. The CMU version of RUNOFF has been modified to interface to the XCRIBL system for use with the Xerox Graphic Printer (XGP). All references to "RUNOFF" in this manual refer to the modified version, XOFF, unless explicitly stated otherwise.

IV.1. Introduction

RUNOFF is a PDP-10 program to facilitate the preparation of typed or printed manuscripts, such as memos, manuals, etc. The user prepares his material on any regular PDP-10. The file includes case and formatting information as well as textual material. RUNOFF then takes the file and outputs it in final form on a line printer, teletype, or other device to produce a final file image. RUNOFF performs the formatting and case shifting as directed, and will also perform line justification, page numbering and titling, etc., as desired. A file which can be shipped to the XGP may be produced. Such files contain control bytes for the XCRIBL system, allowing full use of all the XGP special features.

The principal benefit of such a program is that files prepared for use with it may be edited and corrected easily. Small or large amounts of material may be added or deleted, and unchanged material need not be retyped. After a set of changes, the program may be operated to produce a new copy which is properly paged and formatted. Documentation may thus be updated as necessary without requiring extensive retyping.

XOFF is based on RUNOFF which was written by R. Clements of Digital Equipment Corporation. XOFF contains several new features for controlling the XGP, and also represents an improvement to the original RUNOFF for ordinary output devices.

The user who is familiar with document production systems need only review the commands described in the later sections. The XGP special feature commands which are specific to CMU can be ignored if the output is not to be sent to the XGP.

IV.2. Source File Format

The source file contains the textual material which will appear on the final copy, plus information which specifies formatting. Also, upper and lower case information may also be supplied so that copy can be prepared on the teletype or other such device which can normally input only upper case letters. All command information consists of regular ASCII printing characters so that a listing of the source file may be examined if the final copy is not exactly as desired.

All material in the source file is taken to be source text except those lines beginning with a period. A line beginning with a period is assumed to contain commands which must match one of those listed in this document. The commands provide the formatting information, and control various optional modes of operation.

Usually the text is filled and justified as it is processed. That is, the program *fills* a line by adding successive words from the source text until one more word would cause the right margin to be exceeded. The line is then *justified* by making the word spacings larger until the last word in the line exactly meets the right margin.

The source text can be reproduced exactly by disabling filling and justification. The program may be set to fill but not to justify, in which case the output will be normal except that lines will not be justified to the right margin. The program may also be set to justify but not to fill, although this would probably produce peculiar results and is not recommended.

When the fill mode is on, spaces and carriage returns occurring in the source text are treated only as word separators. Multiple spaces are ignored. Multiple carriage returns are ordinarily ignored, but this is under control of the RETAIN mode.

Some of the commands cause a *b*reak in the output. A break means that the current line is output without justification, and the next word goes at the beginning of the next line. This occurs at the end of paragraphs.

The program will advance to new pages as necessary, placing the title (if given) and the page number at the top of each page. The user may explicitly call for a page advance where desired, and may inhibit the occurrance of a page advance within specified material.

IV.2.1 Case Information

Files are normally prepared in one of two ways: with upper and lower case, or entirely in upper case. Files may be prepared with upper and lower case shifts from model 33 teletypes by using the line editor SOS. The "question-mark escape" convention is used in SOS to obtain case shifts, e.g., "A" will go in as "A" and "?A" will go in as "a". A mode may be used to invert this convention, e.g., "?A" becomes "A" and "A" becomes "a"; to accomplish this for teletypes set mode LOWER by the SOS command "←LOWER". For other input devices which have lower case, use the "TTY LC" monitor command to inhibit translation of lower case to upper case by the monitor, and then use any editor (use SOS in mode M37). Note that the mode for RUNOFF should be set to UPPER CASE.

Specification of cases for files prepared entirely in upper case (e.g., from a teletype using TECO or some other editor which does not have escape conventions for lower case) is done with two characters, up-arrow (↑), and back-slash (\). The appearance of an up-arrow causes the upper case letter immediately following to be transmitted in upper case. The appearance of a back-slash causes the upper case letter immediately following to be converted to lower case. Any letter not preceded by one of these characters is transmitted in the current mode. The mode is initially lower case, and is changed by the occurrance of two successive case control characters or by commands. Two up-arrows (↑↑), or the UPPER CASE command, cause the mode to be set to upper case, and two back-slashes (\\), or the LOWER CASE command, cause the mode to be set to lower case.

The use of ↑, \, ↑↑ and \\ corresponds to the use of the shift and shift-lock keys on a typewriter. Usually, typing appears in lower case. To type one letter in upper case, the shift key (↑) is used. The shift-lock (↑↑) is set to type a series of upper case letters, after which it is released (\\).

The following shows the uses of the case control characters:

↑HERE IS A ↑SENTENCE IN ↑↑UPPER CASE\\ AND LOWER CASE.

becomes:

Here is a Sentence in UPPER CASE and lower case.

Note: Case conversion takes place only on ASCII codes 101 to 132 octal, that is, the upper case letters. Any actual lower case letters (codes 141 to 172 octal) appearing in the source will be transmitted unchanged.

IV.2.2 Special Characters

2.2.1 Underscore shift

The character ampersand (&) will cause the character following it to be underscored, e.g. &f&o&o becomes *foo*. (Note: when this document is output on the XGP, characters which are underscored print in italics).

Underlining of a string of characters can also be specified, in a manner analogous to the use of the shift lock operations described in section 2.1. An appearance of ampersand preceded by up-arrow (↑&) will cause underlining of all following characters except space. An appearance of ampersand preceded by backslash (\&) will disable this mode.

When outputting to devices other than the XGP, any character may be used to print an underline, and several modes may be used depending upon whether the output device can backspace, overprint, or print a non-spacing character. The character used may be selected by the UNDERLINE CHARACTER command. The initial value of the underline character is octal 137, an ASCII left-arrow, which prints as an underscore on many devices.

2.2.2 Non expanding blank (quoted space)

It is occasionally necessary to include spaces in the text which should not be treated as word separators. For this purpose, RUNOFF treats numbersign (#) as a quoted space; i.e. it will print as exactly one space in the output, will never be expanded nor changed to a carriage return.

The XCRIBL system will expand spaces to achieve justification of lines. Since quoted spaces are ordinarily transmitted to an output device as normal spaces, the effect of using a quoted space is lost. To avoid this, the user may define another character code to be a space in his XGP character set, and specify what this character is by the NON EXPANDABLE BLANK command.

2.2.3 Quoting character

To allow the appearance of the special characters (ampersand, number-sign, up-arrow, or back-slash) in the output, the character left-arrow (←, or underscore on some devices) is used as a quote character. The character immediately following a left-arrow will be transmitted to the output with no formatting effect. The left arrow itself is thus another case requiring quoting. The following five cases occur: ←&, ←↑, ←\, ←←, and ←#.

IV.2.2 Special Characters

The quote character applied to any other character gives just that character, e.g., "←A" is just "A". This feature is useful for beginning a text line with a period (e.g., in preparing a RUNOFF manual); just use "←.".

2.2.4 NOCONTROL mode

Note that in NOCONTROL mode (see the NOCONTROL command), all special characters are implicitly quoted, and thus appear as themselves in the output file, and cause no special actions.

IV.3. Commands

The following commands will be recognized if they are on a line started with a period. Any such line is assumed to contain one or more of these commands. If a command on the line cannot be found in the command table, or the parameter is in error or omitted, an error diagnostic will be typed. For SOS and LINED files, the line and page number of the line containing the error is typed; for other files the page number of the page containing the output text is typed. When multiple commands appear on a line, only the erroneous command is typed, and the command scanner attempts to resume scanning at the next semicolon. Some commands take one or more decimal numbers or a text string following the command name. These are separated from the command name by a space. Optional parameters which may be omitted are noted in the manual by enclosing them in brackets [].

Commands, with the exception of TITLE, SUBTITLE, FOOTNOTE, CENTER and INDEX, may appear several per line, separated by semicolons. These commands, when they appear on a line, must be the *last* commands on the line. For TITLE, SUBTITLE, and INDEX, the information up to the end of the line is considered part of the command. For the CENTER command, everything up to the end of the line is simply ignored, and the number of lines specified by the optional parameter (1, if omitted) are centered. For the FOOTNOTE command, everything up to the end of the line is ignored, and footnote processing begins with the next line.

Commands which control the XGP will be ignored if an XGP-compatible output file is not being produced; however, all such commands which break will do so regardless of the mode (XCRIBL or normal).

When abbreviations are used, some illegal commands may be processed as other commands, e.g., "INEDNT" will be recognized as "I EDNT", i.e., it will index the word "ednt". The convenience gained by abbreviations was considered more valuable than the accidental recognition of an invalid command, since such errors show up very quickly.

IV.3.1 Formatting commands

.BREAK
.BK
.B

Causes a break, i.e. the current line will be output with no justification, and the next word of the source text will be placed at the beginning of the next line.

.SKIP [n]

Causes a break after which n×(line spacing) lines are left blank. If the skip would leave room for less than two printed lines on the page (i.e. if there are less than n+2×(line spacing) lines left), the output is advanced to the top of the next page. If n is omitted, 1 is assumed. If a SKIP command occurs at the top of an empty page, it is ignored. To force blank space at the top of a page, use FIGURE.

.BLANK [n]

Exactly like SKIP, except that n (rather than n×(line spacing)) lines are specified. BLANK is used where space is to be left independent of the line spacing; SKIP, where the space should be relative to the size of line space. If n is omitted, 1 is assumed. If a BLANK command occurs at the top of an empty page, it is ignored. To force blank space at the top of a page, use FIGURE.

.FIGURE n
.FIG n

Like BLANK, except that if less than n lines remain on the current page, the page will be advanced, and n blank lines will be left at the top of the new page. Principally used where it is desired to leave room for a figure to be drawn in manually.

.INDENT n
.IND n

Causes a break and sets the next line to begin n spaces to the right of the left margin. n may be negative to cause the line to begin to the left of the left margin (useful for numbered paragraphs).

.PARAGRAPH [n]
.PARA [n]
.P [n]

The number is optional and, if present, sets the number of spaces which paragraphs are to be indented. The initial setting is 5. The command causes a break and leaves (m+1)/2 blank lines, where m is the regular line spacing. The next line will be indented as indicated above.

.PAGE
.PG

Causes a break and an advance to a new page. Does nothing if the current page is empty. Titles and numbering as for automatic page advance.

.TEST PAGE n
.TP n
.NEED n

Causes a break followed by a conditional page advance. If there are n or more lines remaining on the current page, no advance is made and no lines are skipped. Otherwise, the page is advanced as for PAGE. This command should be used to guarantee that the following n lines are all output on the same page.

.NUMBER [n]

Turns on page numbering (normal) and, if n is supplied, sets the current page number to n.

.NONUMBER

Turns off page numbering. Pages will continue to be counted, so the normal page number will appear if numbering is re-enabled.

IV.3.2 Mode Setting Commands

.XCRIBL

Sets the XCRIBL switch. Same as "Y" reply to "XCRIBL FILE?" or file extension .XOF. All underlining modes are turned off, no pause occurs between pages, and no form feed simulation is performed.

This command must appear before any text is processed, and preferrably before any commands which influence the XGP (such as the PAPER SIZE command) occur.

Note that files which have extension .XOF or contain a XCRIBL command will *always* produce XGP formatted output files. In order to produce an ASCII output file (for LPT listing), the file must not contain a XCRIBL command or have extension .XOF.

.JUSTIFY
.JUST
.J

Causes a break and sets the mode so that subsequent output lines will be justified. (Initial setting)

.NOJUSTIFY
.NOJUST
.NJ

Causes a break and prevents justification of subsequent output lines.

.FILL

Causes a break and specifies that subsequent output lines be filled. Sets the justification mode to be that specified by the last appearance of JUSTIFY or NOJUSTIFY. (Initial setting)

.NOFILL

Causes a break and prevents filling of subsequent output lines. ***Also turns off justification.***

Note:

1. The nofill-nojustify mode need be used only where there are several lines of material to be copied exactly. A single line would not require using these commands if

there are breaks before and after.

2. Normally FILL and NOFILL are used to turn both filling and justification on and off. It is usually desirable to do both. However, a subsequent appearance of a justification command will override the fill command.

3. Because of the action of FILL, a single occurrance of NOJUSTIFY will cause the remainder of the file to be unjustified, with filling as specified. In order to justify but not to fill (not recommended), a JUSTIFY command must follow every NOFILL command.

4. When a file is generated in XCRIBL mode, no justifying is done by RUNOFF. Instead, XGP control commands are inserted in the file to instruct the XCRIBL system to justify or not to justify. This is necessary since RUNOFF cannot justify properly for variable-width type fonts.

.CUT
.NOCUT

Sets the automatic-cut switch for the XGP. When automatic cut is on, each page is cut after being printed. When off, no cutting is performed and the output is on a continuous sheet. LOOK sets the cut switch initially on.

.RETAIN
.NORETAIN

These commands control the retain mode. Normally, blank lines are discarded. When in retain mode, blank lines encountered in NOFILL mode are retained, and blank lines encountered in

ordinary text. Case shifting will still take place, as determined by the commands UPPER CASE, LOWER CASE, or the last shift-lock (↑↑ or \\) done in CONTROL mode.

.LOWER CASE

Indicates the text coming in is in upper case (input by TTY or similar device) and is to be shifted to lower case. Capitalization will be indicated by the use of the shift character, ↑.

.UPPER CASE

Indicates the text coming in is in upper and lower case. Upper case letters will not be downshifted.

IV.3.3 Parameter Setting Commands

.LEFT MARGIN n
.LM n

Causes a break after which the left margin is set to n. n must be less than the right margin, but not less than 0. The initial setting is 0. The amount of any indent plus the left margin must not be less than 0.

.RIGHT MARGIN n
.RM [n]

Causes a break after which the right margin is set to n. n must be greater than the left margin. The initial setting is 60.

The number of characters on a line will be equal to or less than the right margin minus the left margin minus any indenting which may be specified. Even if filling has been disabled, lines will not be extended past the right margin.

.ALTER RIGHT MARGIN n
.ARM n
.ALTER LEFT MARGIN n
.ALM n

Adds the value of n to the current value of the left or right margin. The value of n may be negative. These commands break.

.SPACING n

Causes a break after which the line spacing will be set to n. n must be within the range 1 to 5. Single spacing is 1, double spacing is 2, etc. The initial value of spacing is 2.

.PAPER SIZE n [,m]

Sets the number of lines per page to n. n must be greater than 10. The initial setting is 58. n includes the top margin of 5 lines. The page number and title appear on the second line. The second argument, m, is optional. If present, it sets the paper width in columns (initially 60). It must be greater than the left margin, and it is set into the right margin as if a RIGHT MARGIN m command had also been typed. This command is usually used only at the beginning of a file, but may be used throughout if needed.

.ALTER PAPER SIZE n [,m]
.APS n [,m]

Adds the value of n to the number of lines per page, and m (if present) to the maximum right margin. This command breaks.

.TAB STOPS [n [,n ...]]
.TABS [n [,n ...]]

Clears all previous tab stops and sets new tab stops as specified. The several n (max 32) must be greater than zero and in increasing order. They are the positions of tab stops independent of the setting of the left margin, although any which are less than the left margin will not be seen. There are no tab stops initially.

Tabs should be used only in lines which will be unjustified and unfilled, i.e. where filling is disabled or a break immediately follows. Clearly, the spaces specified by a tab character are not expanded to justify the line--this would defeat the effect of tab formatting. The appearance of a tab in the source text will be expanded to one or more spaces, the amount necessary to advance to the next tab stop. If a tab appears at a point where no further tab stops have been set on a line, the tab will be treated as though it had been a space.

For XGP output files tabs will not be expanded to spaces to accomplish TAB expansion when in a non-justify mode. Instead the XGP is positioned to raster position TABSTOP[i]×CW (for the i-th tab stop). If this position has been passed, no action takes place. Note that tab stops will work properly only on *unjustifed* lines. A request to tab which appears on a justified line will be expanded to a number of spaces, as before. This will generally produce unsatisfactory results when using variable-width fonts; it is a compromise since the XGP will not process tab requests when in JUSTIFY mode.

.VERTICAL SPACE n
.VS n

Sets the between-line vertical spacing of the XGP to be n raster units. LOOK initializes this value at 8. This command breaks.

.TOP MARGIN n
.TM n

Sets the top margin of the XGP page to be n raster units. LOOK initializes this value at 128.

.BOTTOM MARGIN n
.BM n

Sets the bottom margin of the XGP page to be n raster units. LOOK initializes this value at 128.

.XGP LEFT MARGIN n
.XLM n

Sets the left margin of the XGP page to be n raster units. This left margin defines the zero position for ordinary LEFT MARGIN and INDENT commands. This command breaks.

.PAD [n]

Sets the padding parameter to the XGP to be n raster units. If n is omitted, the padding is set to (RM×CW/2). The padding value is initialized to this computation for any XCRIBL output file. It is not changed when the right or left margins are changed, unless another PAD command is issued.

Note: if the PAD value is set too small (e.g., 0), some (or all) lines of the output file will be printed unjustified, regardless of the desired mode.

IV.3.4 Character Set Control

.CHARACTER WIDTH n
.CW n

Sets the character width to the value n. This value is used to compute the right margin, padding values, and tab positions for the XGP. It is used as a best guess for variable-width sets. The character width is initialized at 20, for a set which prints 10 fixed-width characters per inch.

.SELECT A SET AS n
.SA n

.SELECT B SET AS n
.SB n

Selects the character set in the A or B partition to be the set defined by the integer n. This set must be on the disk at the time the file is output on the XGP. These commands break.

As of this writing the disk character set feature of the XCRIBL system is not implemented, and these commands are ignored.

.WAIT FOR A
.WAIT FOR B

After a SELECT operation, one of these commands will assure the selected set is in core before any attempt is made to use it.

As of this writing the disk character set feature of the XCRIBL system is not implemented, and these commands are ignored.

.USE A
.USE B

Chooses which of the two character set partitions, A or B, will be used to perform output.

.NON EXPANDABLE BLANK /x/
.NXB /x/
.NON EXPANDABLE BLANK n
.NXB n

The character between the slashes is used in place of a blank any time a non-expandable blank (#) is to be output. Alternately, the *decimal* number representing the character code may be written as the parameter. For most standard sets, this character will be the delete character, octal code 177 (decimal 127). Outputting a character other than a normal blank (octal 40) insures that the justification algorithms of the XCRIBL system will not attempt to justify by expanding the blank. The character set being used must have this character defined as a blank (i.e., a character of non-zero width with no points turned on). The non-expandable blank is initialized to a normal blank (octal 40).

.UNDERLINE SET A
.USA
.UNDERLINE SET B
.USB

Selects which of the two partitions (A or B) will be used to simulate underscores. Typically one would have an italic set in the partition selected. The selection is independent of the character set selected by a USE A or USE B command. The underline set is initialized to 0, indicating no special set will be selected for underlining. Note that the heights of the character sets must be the same in both partitions, or chaos quickly follows.

IV.3.5 Miscellaneous Commands

.TITLE tttt ... tttt

This command takes the remaining text on the line as the title. This text will appear at the top of all subsequent pages, at position 0, on the second line with the page number. The title is initially blank.

Note that the page number, which usually appears in the center of the page, is always forced to be beyond the title. Thus the page number may be forced to the right of the page by inserting non-expandable blanks (#) to the right of the title text. The word "PAGE" may be affixed by making it the last word of the title, appropriately separated from the rest of the title for readability.

.SUBTITLE tttt ... tttt
.SUB tttt ... tttt

This command takes the remaining text on the line as the subtitle. This text will appear on the line immediately following the title and page number. The subtitle is initially blank. The subtitle is not indented, but may contain leading spaces to achieve the same effect, if desired.

.CENTER n [,m]
.CEN n [,m]
.C n [,m]

This command causes a break after which it centers the next line following in the source file. The centering is over the column n/2, independent of the setting of the left and right margins. If n is missing, n is assumed to be the paper width, initially 60. (See PAPER SIZE command.)

The optional argument, m, specifies that the next m lines (including blank lines) are to be centered. If m is omitted, it is assumed that one line (i.e., the next line) is to be centered.

.FOOTNOTE n
.FOOT n

Allocates n*(line spacing) lines at the bottom of the current page for a footnote*. If insufficient room remains on the current page, space will be allocated at the bottom of the following page. The text for the footnote begins on the line following the command, and it may contain any appropriate

commands (e.g. CENTER, SKIP) necessary to format the footnote.. The footnote is terminated by a line beginning with an exclamation point (!) (the remainder of which is ignored). The lines delimited by the FOOTNOTE command and this line are put into a buffer to be processed when the output moves to within the stated distance of the bottom of the page. If a page has multiple footnotes, the allocated space is the sum of the allocations for all footnotes assigned to the page. The user must include his choice of footnote-designating symbols within the text.

The current values of such information as left and right margin and line spacing are saved and restored after processing of footnotes. Therefore, a footnote may contain commands which change these parameters, and the effect will be limited to the footnote text.

The actual space taken by the footnote may be more or less than that specified by n. The n merely allocates space and should be the user's best guess. If it is considerably off, the footnote lines may overflow the page, or extra space may be left at the bottom. The user may wish to adjust n after examining a first draft printout.

.TAB CHARACTER c
.TC c

Uses the character c in the input to signal a tab operation. This allows the input file to be more compact than when the ordinary tab (octal 11) is used. The tab character is expanded to the number of blanks required to achieve the next tab stop (set by the TAB STOPS command), or generates an XGP "TAB" command. The tab character disappears from the input. Any character except blank and semicolon may be used for a tab character. The tab character is initialized to the ASCII tab (octal 11).

.UNDERLINE CHARACTER c
.ULC c

Specifies the character to be used for outputting an underline for non-XCRIBL devices (e.g., Datels). The underline character is initialized to left arrow (←) which prints as an underscore on most devices. Any character except blank and

---

[1] This is a footnote. This text and the dividing line above were specified by text and commands following a FOOTNOTE 5 command.

semicolon may be specified.

.INDEX tttt ... tttt
.I tttt ... tttt

This command takes the remaining text on the line as a key word or words and adds it, along with the current page number, to the internal index buffer. The command does not cause a break. It should appear immediately before the item to be indexed. A key word or words may be indexed more than once.

.PRINT INDEX

Causes a break after which it prints the entire contents of the index buffer. Entries are printed in alphabetical order, and are set against the left margin. Regular line spacing is used, except that a blank line is left between entries of different first letters. The number of the first page on which each entry appeared is put on the same line as the entry, beginning at the middle of the line (midway between the left and right margins). Additional page numbers for multiple entries follow, separated by commas. The index buffer is left empty.

When an index is produced in XCRIBL mode, periods will not be output between the item name and its page numbers. Instead, a tab to raster position (RM×CW)/2 is made before outputting the numbers.

.BEGIN [tttt ... tttt]
.END [tttt ... tttt]

The idea of block structure is that any changes to margins, spacing, etc. are local to the block and will be restored to their original values upon exit from the block. The RUNOFF block makes all variables except the TITLE, SUBTITLE and page number local to the block. In addition, any XCRIBL commands issued in the block will be undone when the block is exited. Blocks may be nested, and an implicit block surrounds footnotes.

If a file contains and END for which there was no matching BEGIN then an error message is issued for that END command. If, upon completion of processing of a file, there were BEGINs which had no matching ENDs, an error message to this effect is issued.

As an aid to the user in assuring that blocks are properly nested, any string of characters (not including a semicolon)

may be included after a BEGIN. This is the ***block name*** for the block defined by that BEGIN. The END which matches the BEGIN must have the same block name following it as well, or it is flagged as an illegal command.

As of this writing, the block structure feature is not implemented. Although a block surrounds each footnote, if any commands issued within that block change the XGP status, the original status may not be restored.

IV.4. Program operation

RUNOFF is a sharable program much like most PDP-10 cusps. monitor.

The program will first respond with its title and version number, and then ask:

INPUT FILE:

The source file descriptor (device, name, extension, and project-programmer number) should then be typed. The default extension for RUNOFF source files is .RNO. If the file is intended only for the XGP, the extension .XOF may be used; this informs RUNOFF that the file is intended for the XGP and saves some unnecessary questions, i.e., those below. RUNOFF will always look for a .RNO file before looking for an .XOF file. If the name is terminated with an altmode, the output file will be written on the disk in normal mode (ASCII for .RNO files; XCRIBL for .XOF files) and operation will begin. The extension of the output file will be .DOC for .RNO files and .XGO for .XOF files.

Note that the project-programmer number may be either octal (DEC PPN) or in CMU format.

If a carriage return is typed (instead of an altmode), the program will then ask:

OUTPUT FILE:

The destination file descriptor should be typed. The destination device may be TTY:, the teletype, which is limited in that all letters are upper case, and underlining is not available. (These restrictions do not apply to some more elaborate teletype-like terminals.) Alternately, a disk file name or other device and file name may be given, and the final image will be written onto it. If the output is to be shipped to the XGP, it must be written on the disk, and later shipped to the XGP via LOOK. If no explicit extension is given, the extension .DOC will be used for input files with .RNO extensions, and .XGO for input files with .XOF extensions.

If the output file name is terminated by an altmode, the following questions will not be asked, and normal mode will be assumed (ASCII output for .RNO files and XGP output for .XOF files). If the name is terminated with a carriage return, the program will ask:

XCRIBL FILE?

to which the user replies Y or N. If the reply is Y, and the reply is not terminated with an altmode, the program then asks:

UNDERLINE PARTITION (A,B,N)?

The user may then specify which character set partition is to be used to simulate underlining, A, B, or Neither (for a complete description of this feature, see the UNDERLINE SET command in section 3.4). The option selected will be overridden by any UNDERLINE SET command appearing in the file.

If the reply to the XCRIBL FILE question is N, the program then asks:

UNDERSCORE (L, C, B OR N)?

to which the user must reply L, C, B, or N. L (*Line*) is normal, and is used for a line printer with underlining or for files which will eventually be copied to such a line printer. It causes underscoring (where specified by &) to be done by overprinting a line. C (*Charcter*) is used for files which will be printed with a flexowriter or any other device which has a non-spacing underscore character. B (*Backspace*) causes each underline character in the output file to be followed by a backspace character. This mode should be used for terminals which have both a spacing underscore and a backspace function. N (*No*) causes underlining requests to be discarded.

If the last reply was not terminated with an altmode, the program will next ask:

SIMULATE FORM FEED (Y OR N)?

to which the user must reply Y or N. N is normal and means that a form-feed character (control-L) will be used to advance the listing to the top of a page. This mode should be used for the line printer and for files intended for the flexowriter. Y means that the program will advance to a new page by typing an appropriate number of line-feed characters. It must be used with any device which does not have mechanical page advance built in.

If the last reply was not terminated with an altmode, the program will then ask:

PAUSE?

A Y answer means that the program will stop at the top of each page to allow the paper to be positioned manually. The user should type a space to cause printing to be resumed. This mode is appropriate for typing the output onto pages of duplicating masters or the ARDS

display. (Note: a special program, TYPER, also provides this facility, as well as the ability to omit pages and selectively retype others).

When the program has completed processing, it will type DONE, and return to monitor level.

Note that if the file contains a XCRIBL command (described in the command section), then underscoring, form feed simulation, and pausing are all turned off. Note, however, that XCRIBL output files may simulate underscoring by use of a second character set (e.g., and italic set); see the UNDERLINE SET command.

IV.5. Output file format

The differences between normal ASCII output files and XCRIBL output files are:

1. "ASCII" or ordinary output files contain only ASCII characters and may be printed on any suitable output device. Such files may also be printed on the XGP but special actions such as justification may not operate properly. Such features as font change are not available.

2. "XGP" format files also contain ASCII characters but in addition contain commands to the XCRIBL system (delete characters followed by 1 to 3 bytes of control information). Since these commands and control bytes are interpreted as characters by ordinary output devices, the file may not print properly on devices other than the XGP.

   Note that lines in an XGP format file are not justified; justification is done by the XCRIBL system at the time the file is output.

   A special program, TYPER (Version 3) is available. Use the /L switch to read XGP format files. The output will contain the interpretation of the XCRIBL commands enclosed in double brackets [[]]. This program is primarily a diagnostic aid to those involved in XCRIBL system development.

IV.6 Index

INDEX
(Entries entirely in UPPER CASE are commands)

IV.6 Index

IV.6 Index

IV.6 Index

Security Classification

DOCUMENT CONTROL DATA - R & D

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

1. ORIGINATING ACTIVITY *(Corporate author)* Computer Science Department Carnegie-Mellon University Pittsburgh, Pennsylvania 15213	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

XCRIBL: A HARDCOPY SCAN LINE GRAPHICS SYSTEM FOR DOCUMENT GENERATION

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

Scientific Final

5. AUTHOR(S) *(First name, middle initial, last name)*

R. Reddy, B. Broadley, L. Erman, J. Newcomer, G. Robertson, J. Wright

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 28, 1972	68	0

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F44620-70-C-0107 b. PROJECT NO. 9769 c. 61102F d. 681304	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Research(NM) 1400 Wilson Blvd. Arlington, Virginia 22209

13. ABSTRACT

XCRIBL is a system developed at CMU for generating hard copy computer output of arbitrary type fonts, graphics, and grey-scale images using a Xerox Graphic Printer (XGP). XCRIBL can be used to generate documents approaching the quality of printed text with the use of a document generation language (XOFF or PUB) and a character set design program (BILOS). Textual and graphic information to be printed is shipped in its raw form from the host computer (PDP-10) to a mini-computer (PDP-11) which acts as an intelligent channel controlling the XGP. Careful design of the data structures and the hardware interface permit the mini-computer to generate each scan line as needed without having to resort to a brute force solution of generating a bit-image for the whole page (3.5 million bits) for off-line printing. Variable width characters and the ability to mix text and graphics distinguish the present solution from the known simpler schemes for scan line generation.

DD FORM 1 NOV 68 1473

Security Classification