Xerox 9700 Electronic Printing System
Reference Manual

XEROX

9700

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

Xerox 9700 Electronic Printing System

Reference Manual

91 00 02B

September 1978

Printed in U.S.A.

NOTICE

This publication is a major revision of the Xerox 9700 Electronic Printing System Reference Manual (Preliminary), dated June 1978.

RELATED PUBLICATIONS

Title	Publication No.
Xerox 9700 Electronic Printing System Forms Creation Guide	91 00 01
Xerox 9700 Electronic Printing System Font Users Guide	91 00 03
Xerox 9700 Electronic Printing System Tape Formats	91 00 04
Xerox 9700 Electronic Printing System Operator's Guide	600P81096

CONTENTS

APPENDICES

TABLES

1. INTRODUCTION

General

The Xerox 9700 Electronic Printing System is a versatile, high-performance, general purpose, programmable printing system which may be used either as a stand-alone off-line printing system to process a variety of computer-generated magnetic tapes or as a sophisticated on-line printing subsystem.

High-performance operating characteristics of the Xerox 9700 are achieved by the effective combination and application of xerographic, laser, and computer technologies to control the flow of digital data, as well as the use of pre-cut standard size (8.5 x 11 inch) sheets of paper.

Features

- Standard 8.5 x 11 cut sheet output, using plain or pre-printed paper.
- Executive quality output (300 x 300 bits per inch resolution).
- Large variety of paper weights (16-pound bond to 110-pound index) can be intermixed.
- Wide range of paper types – multicolored, 3-hole drilled, perforated.
- 2 sheets per second – up to 18,000 LPM.
- Continuous operation – dual input paper trays, dual output stackers, disk buffering of variable data.
- Electronic forms – 9700 generated capability for forms, logos, shading. (A forms library is stored on the system disk.)
- Highlighting of data – copy modification entries (CME).
- Landscape or portrait orientation.
- Multiple character sets (fonts) per page, including OCR, typewriter, italics, script.
- Fonts are selectable on a character-to-character basis.
- Multiple forms per report.
- Varying page densities by use of different font styles and sizes.
- Variable character size (4-24 point) and spacing (4-30 CPI) and proportional character spacing based on font style.
- Variable line spacing (3-18 LPI).
- Overprint capability
- Sample print tray (25 page capacity when using 20-pound bond paper).
- Copy set integrity maintained.
- Stacked reports.
- Interspersed reports.
- Selective processing of portions of reports.
- User-initiated offsetting of portions of reports.
- Extended job accounting.
- Job routing capability.
- Containerized output.
- Multiple copies, collate option.
- On-line to IBM 360/30 and up, and IBM 370/135 and up.
- Selectable input mode (on-line or off-line).
- Extensive tape formats.
- System status and operator prompting via text sequences.

- Keyboard display, for use as operator control and editing functions.
- Disk, for use as secondary memory.
- 2 bin output 1500 page/bin capacity (using 20-pound bond paper).
- 2500 page main input feeder, 400 page auxiliary input feeder (using 20-pound bond paper).

Off-line Host System Support

The Xerox 9700 supports tapes from the following host systems, as defined in the 9700 Tape Formats manual:

- Burroughs medium systems (B2500, B2700, B3500, B3700, and B4700) printer backup and standard tapes.
- Burroughs large systems (B6700) ANSI, printer backup, and standard tapes.
- Honeywell 200/2000 series SPR, standard and COBOL print tape formats.
- Honeywell 6000 series SSF print tapes (packed format with embedded normal edit mode carriage control).
- OS Writer support, including recognition of banner pages for report offsetting.
- IBM DOS/360 and DOS/VS/370 POWER
- ANSI
- IBM OS/360 and DOS/360 Standard Tape Formats
- IBM DOS/360 GRASP
- UNIVAC SERIES 70
- UNIVAC 1100 Standard Data Files

Upward Compatability from Xerox 1200 Computer Printing System

Xerox 9700 provides an upward compatible path for Xerox 1200 (EPCP B01) users. The following compatibility requirements are met:

- Standard 9-track, 1600 bpi tape formats accepted by Xerox 1200 (as listed above) are accepted by Xerox 9700.
- Format control specifications supported on Xerox 1200 are available on Xerox 9700.
- Options for stacked reports, interspersed reports, multiple copies, logical file separators, etc., are specifiable for Xerox 9700.
- Xerox created user interface conventions established for the Xerox 1200 are supported by Xerox 9700.
- On-line 1200 facilities are available on Xerox 9700.

2. SYSTEM OVERVIEW

Introduction

This chapter provides a functional overview of both the hardware and software aspects of the Xerox 9700 Electronic Printing System.

Hardware Functional Description

Major hardware units of this Xerox 9700 Electronic Printing System are: a xerographic printer, an imaging unit, an output stacker, a system disk, a controller (mini-computer), and a keyboard/display unit. Additionally, for on-line operations a channel interface controller is required, for off-line operations a magnetic tape unit and controller are required, and for selectable on-line/off-line operations both controllers and the magnetic tape unit are required. As shown in Figure 2-1, the printer, imaging unit, and output stacker are physically joined. The controller, system disk, magnetic tape unit, and the keyboard/display unit are connected via cables to the system.

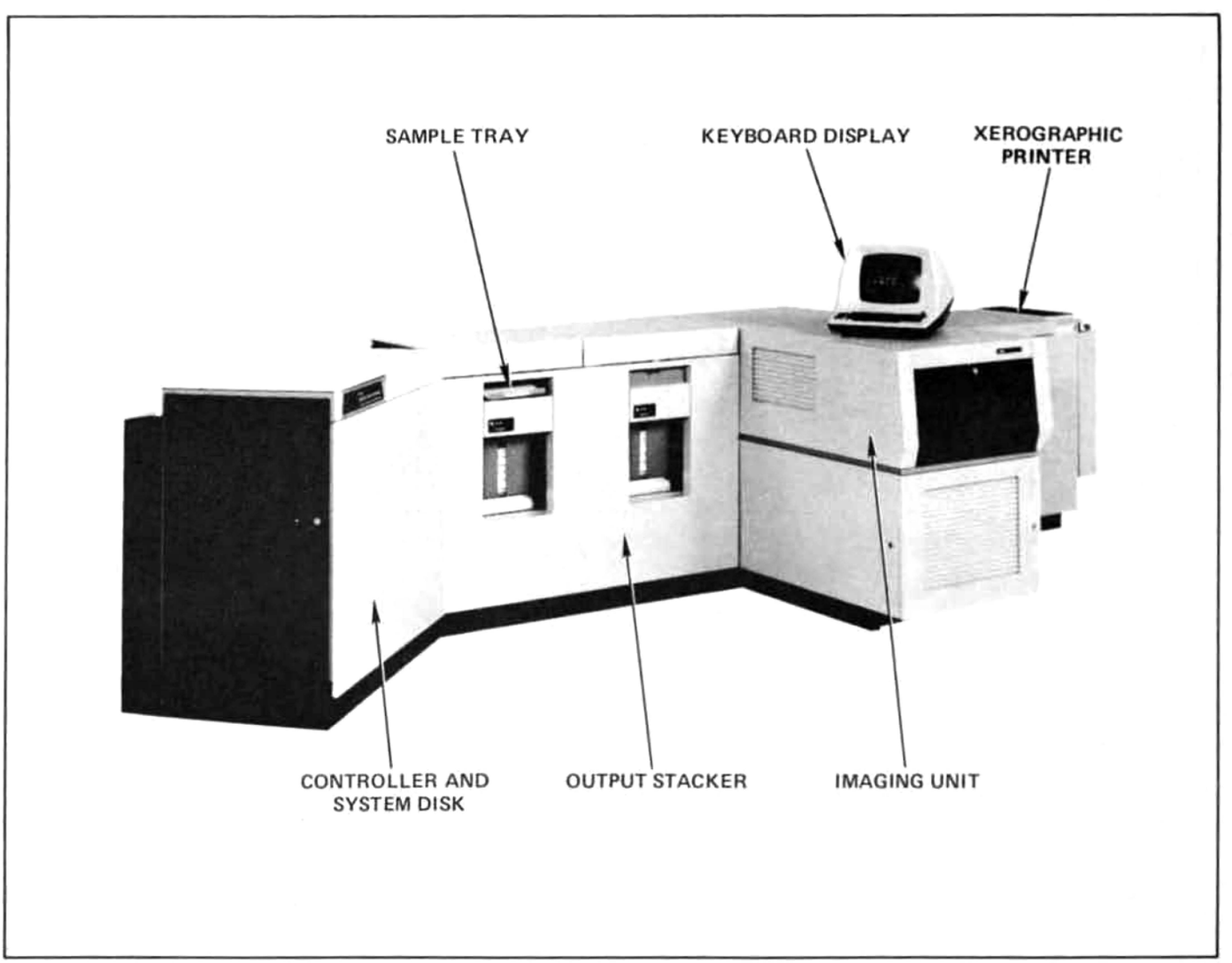


Figure 2-1. Typical Xerox 9700 Electronic Printing System

Functionally, the Electronic Printing System may be considered as five subsystems which perform the tasks necessary to meet the system's requirements. These subsystems (see Figure 2-2) are: Input, Control, Imaging, Xerographic, and Output.

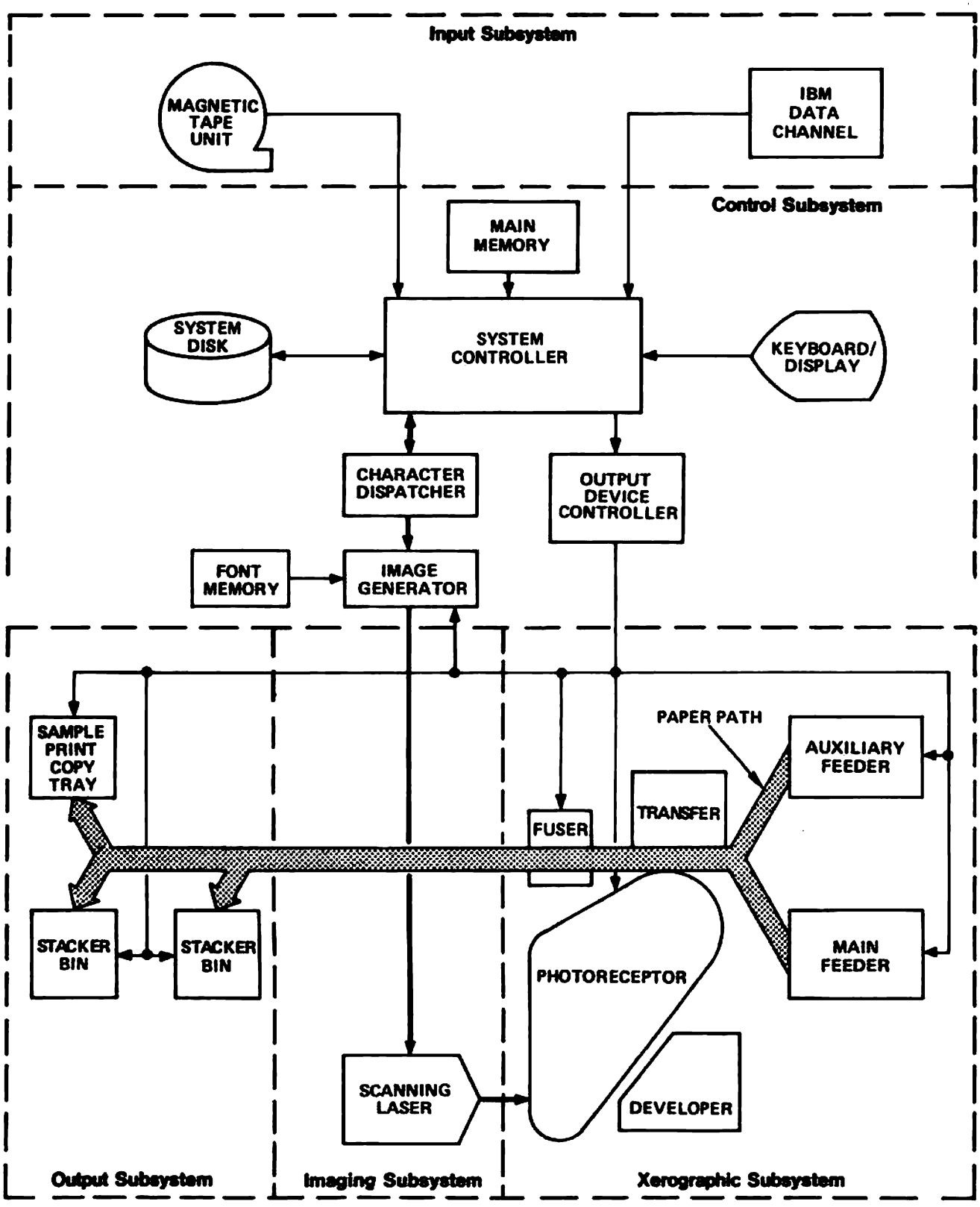


Figure 2-2. Functional Diagram of Xerox 9700 Electronic Printing System

Input Subsystem

The input subsystem provides print data to the system. In off-line operation, the print data is obtained from a 9-track 1600 bpi magnetic tape unit. In on-line operations, the print data is obtained via a data channel connected to either an IBM 360/30 and up or an IBM 370/135 and up. A selectable system allows either mode of operation.

Control Subsystem

The control subsystem performs all of the data handling, formatting, buffering and operator control of the system. It consists of a system controller (a mini-computer), main memory, a keyboard/display unit, a disk unit, and interface and control electronics. Stored in the system controller main memory is a software task as selected from the operating system by the executive task. The software tasks direct page formatting, system control, forms compilation. and system diagnostics.

The interface and control electronics handle controller-to-device communications and control of specific device functions. They provide the necessary interfacing between the I/O circuits and the device circuits. They also accept control orders from the system controller and exercise the indicated functional control over the devices. Device status is communicated to the computer software via the control and interface electronics.

The keyboard/display unit permits the operator to exercise control over the system. The control software communicates with the operator via easy-to-use English language instructions, status, and informational displays. Operator commands are also in English and easy to learn.

Imaging Subsystem

The imaging subsystem accepts the formatted page of data, which has already been merged with the forms data, and utilizes a scanning laser beam to generate lines of modulated light for the continuously moving photoreceptor of the xerographic system, creating a latent image of the output page. The raster scanning laser beam gives wide flexibility in formatting the output page, since composition, size, style, and orientation of character sets, electronic forms, and logos can be programmably modified. Electronic forms, logos, and character sets which have been previously stored on the disk may be called as needed and merged electronically resulting in perfect registration between the variable data and the forms data. The orientation of the printed data may be either landscape (parallel to the long dimension of the paper) or portrait (parallel to the short dimension).

Xerographic Subsystem

The xerographic printing subsystem incorporates all the typical functions of the xerographic printer including paper handling and development of the latent image of the output page.

All information to be printed, including forms and logos, as well as text, is received by the xerographic processor as a series of binary light signals from the scanning laser beam. Xerographic properties of an endless belt permit its surface to be uniformily charged electrically and then selectively discharged (or "exposed") by a modulated laser beam to create a latent image of the desired output. The latent image is subsequently "developed" by using dry ink (or toner) and then transferred to a sheet of plain paper. The final image is "fixed" when the paper passes through a fusing station on its way to the output

stacker. The endless belt is cleaned and recharged in preparation for the next cycle of operation. This imaging and printing process is performed with a resolution of 300 bits (or dots) per inch in the horizontal and vertical directions and at the rate of two sheets per second, where each sheet is 8.5 x 11 inches.

The main paper feeder may hold up to 2500 sheets and the auxiliary paper feeder may hold up to 400 sheets. When both paper feeders are filled with the same type of paper the printer may be operated continuously by utilizing the auxiliary paper feeder while the main paper feeder is being filled. Alternatively, the auxiliary paper feeder may be filled with paper (as heavy as 110-pound index) to be used as covers or separators. Transfer between main and auxiliary paper feeders is programmably controlled. Note that paper weights may be intermixed in either feeder.

Under program control, individual sheets of paper are obtained either from the main or auxiliary paper hopper, passed over a transfer station (where the image is received on the underside of the sheet, i.e., the sheet is in a face down position), passed through the fuser station (where the image is fixed), and routed to the appropriate output bin or sample print tray (still face down). Thus, successive sheets of a multi-page report are in the proper sequence in the output bins or sample print tray. Since the individual sheets are moved in a straight, direct, and short path, without being inverted, the probability of a paper handling error is very low.

Output Subsystem

The output subsystem provides the capabilities for paper stacking, report collating and sample prints. The two-bin output stacker is a modular unit with its own paper transport mechanism. It is physically joined to the imaging unit and operates as an extension to the printer. Pages delivered by the printer are carried across the top of the stacker, routed to the appropriate bin, and stacked face down. Therefore, the first page of a job or run appears on the bottom of the bin and the last page appears at the top. The pages are in the correct order because of the face-down stacking. As a convenience to the operator in distinguishing between output sets, separation is provided by offsetting each copy set approximately 0.5 inches from the previous one. The output is containerized to help retain offset integrity of the stock while being moved to the offline finishing station or work area.

Each stacker bin may hold up to 1500 pages. Automatic bin switching to the inactive bin occurs when the active bin has been filled. If the inactive bin is not ready when the active bin reaches its capacity, the system will stop printing. Normal processing may be continued after a bin has been made ready. Bins can be unloaded without stopping the printer, as long as one bin is in the ready mode.

The operator may select either bin for loading. If both bins are empty and initialized, and neither is manually selected, bin 1 is selected automatically. Bin selection may be switched by the operator at any time, provided the alternate bin is in the ready mode.

The sample print tray permits delivery of sample-print pages directly to the user. Sample print pages may include printouts of information stored in the system (i.e., specified forms, font sets, and logos) or a duplicate copy of the current output page or a sample page, if no report is being processed. Logs and Dumps, generated at operator request or automatically by the system, are also delivered to the sample print tray. This sample print feature allows the user to examine any job page.

Unless the sheet is intended for delivery to the sample print tray only, each sample print sheet is reprinted and delivered to the active stacker bin. Thus, the integrity of the output copy set is maintained.

Software Functional Description

Functionally, the Electronic Printing System software, called the Xerox 9700 Operating System Software (OSS) consists of several identifiable components as illustrated in Figure 2-3. These components are:

- The EXECUTIVE which schedules all processing for the Operating System and the Operating System Diagnostic Software.
- An INPUT PROCESSOR which decodes input data from an off-line magnetic tape or from an on-line interface. Further details about the INPUT PROCESSOR functions are described below.
- An OUTPUT PROCESSOR which controls the operation of the xerographic printer, automatically merges forms data with the computer-generated data, and outputs this data in a coded format to be printed. Further details about the OUTPUT PROCESSOR functions are described below.
- A PRINT DESCRIPTION LANGUAGE (PDL) processor which permits the specification of general printing environment characteristics.
- A FORMS DESCRIPTION LANGUAGE (FDL) processor which allows for the description of forms on the system using simple, English language commands. Refer to Forms Creation Guide Xerox Publication No. 91 00 01 for further details on FDL.
- An EDITOR which facilitates editing source files and manipulating disk files.
- A FILE MANAGEMENT SUBSYSTEM which creates and maintains disk files.
- OPERATOR COMMUNICATION SUBSYSTEM which provides the interface between the 9700 operator and other subsystems.
- Various UTILITIES which do such tasks as produce accounting summaries and control font loading.

Input Processor Functions

Input data is processed and written to disk for subsequent printing. During processing, the following functions are performed:

- Block selection/deletion
- Extraction of records within blocks
- Record selection/deletion
- Detection of user-defined offset points
- Detection of report delimiter records
- Detection of parts of reports to print
- Detection of Dynamic Job Descriptor Entry (DJDE) records
- Detection of user-specified font usage requests
- Character translation
- Printer carriage control processing
- Replacement of input text with pre-specified static text
- Font change information

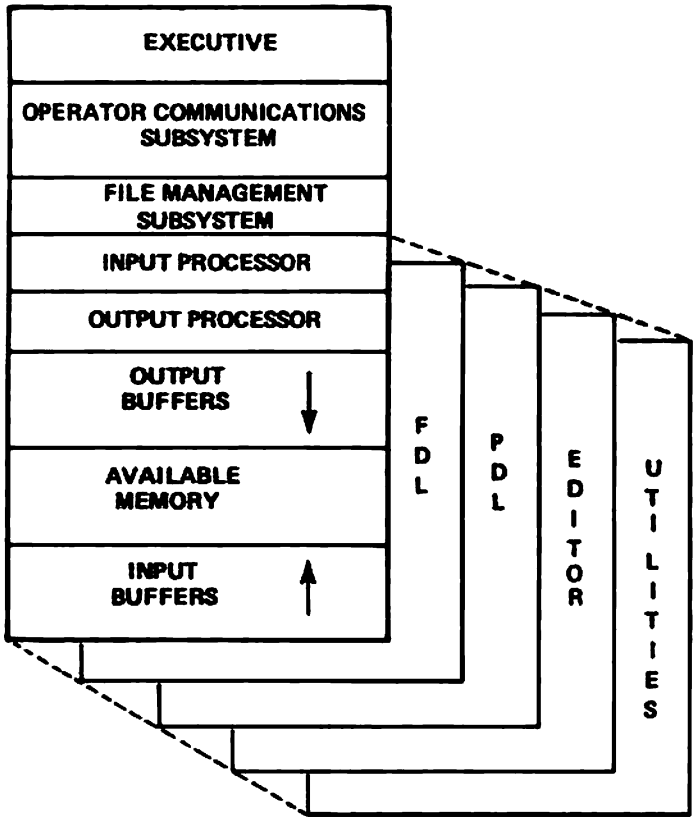


Figure 2-3. 9700 OSS Components

Block Selection/Deletion. The user may specify that certain data blocks are to be deleted for processing purposes. These options are specified through the use of the BSELECT or BDELETE commands (see Logical Processing Commands). If no option is specified all data blocks are selected for processing.

Record Extraction. Record extraction is performed according to the structure of the input data. In the case of data input from tape, blocks may be variable or fixed in length, and records within blocks may be of fixed, variable or undefined length (see "RECORD" Command). Records may be separated by constant delimiters, in which case a search process is required to locate records within blocks.

Record Selection/Deletion. The user may specify that only certain data records are to be processed for printing. Alternatively, the user may specify that certain data records are to be deleted from the printed report. These options are specified through the use of the RSELECT or RDELETE command (see Logical Processing Commands). If no option is specified all data records are selected for printing. All selected data records are moved from the input buffers to disk blocking buffers in preparation for writing to disk.

Detection of Report Delimiter Records. In general, end-of-report is synonymous with end-of-file. However, the user may specify that the end-of-report is to occur upon detection of certain data within an input record. This is done through the use of the RSTACK command. In this case, detection of the specified data will cause end-of-report processing to be performed.

Detection of Dynamic Job Descriptor Entry (DJDE) Records. If an identifier has been specified for DJDE Commands (see IDEN command), then each record is interrogated to determine if it contains DJDE information. DJDE information is processed as encountered and modifies current job characteristics. The effect of this is to permit the user to tailor the printing environment according to the particular needs of a report. DJDE information is applied to control the printing of data occurring after the DJDE information. DJDE records may occur within report delimiter records and in this case the DJDE information will apply to the next report following.

Detection of User-Specified Font Usage Requests. The user may specify that selected data records contain a byte index indicating the font to be used in printing data contained in the record. The index byte is specified as a number in the range 1 to n, where n is the number of fonts specified in the PDL FONTS option of the OUTPUT command. The index specified is an index into the ordered list of fonts specified in the FONTS option. The user may intermix the use of fonts within a print line by specifying different fonts within different input data records, each of which contains a printer carriage control that causes normal upspacing to be inhibited.

Character Translation. As selected data records are moved from the input buffers to the disk blocking buffers, character translation is performed as necessary. The user specifies the input format through the use of the CODE option of the PDL VOLUME command.

Printer Carriage Control Processing. A printer carriage control (PCC) type is specified by the user according to the type of host system (see PCC command). Each PCC value specified corresponds to a particular action to be taken (i.e., space or skip before printing, print or not, and space or skip after printing). Channels are associated with page print line positions by the PDL VFU command. Spacing and skipping actions are processed by associating the appropriate laser scan line or dot position with the beginning of each data line to be printed. The starting print line position so generated is part of the control information associated with each print line and directs the character dispatcher in the printing of the line.

Copy Modification Entries. The user may specify that certain text is to be used to replace selected portions of printed page data. This is done through the use of Copy Modification Entries (CMEs). A CME permits replacement of selected portions of page data with pre-specified static data. Different CMEs may be associated with different copy plies of a report. CMEs may also be used to create font changes in variable data.

Output Processor Functions

Output data is read from the disk and dispatched to the printer via the image generator a page at a time. During processing the following functions are performed:

- Page log read
- Fonts loaded into font memory
- Font specifications loaded into main memory
- Form loaded
- Paper feeding control
- Report integrity control

Page Log Read. The page log is read into memory to provide the output task with the font and form information required for imaging the page. It also provides a look ahead capability for determining what new fonts or forms will be required for the next page.

Fonts Loaded into Font Memory. Fonts are loaded into font memory as required by the page log for the next page to be imaged (i.e., if different from the fonts currently being used and not currently loaded for previous pages).

Font Specifications Loaded into Main Memory. The font specifications are loaded into main memory to provide the necessary line and character spacing information for imaging the page using that font.

Form Loaded. A form is loaded into memory for the next page to be printed, if different from the form currently being printed and not currently loaded for previous pages.

Paper Feeding Control. The paper feeding from the main and auxiliary paper trays is controlled by the output task. When the same stock is loaded into both trays, an automatic switch can occur to the full tray when the tray currently in use becomes empty. When cover stock is loaded into the auxiliary tray, it will be picked at the beginning and/or end of the report if specified by the job descriptor information.

Report Integrity. The integrity of the report is maintained by the system through character parity checking, automatic page tracking along the paper path, aborting any page which may be affected by hardware error, and automatic page recovery of all affected pages after a paper jam is cleared.

3. PRINT DESCRIPTION LANGUAGE

Introduction

The Print Description Language (PDL) is a keyword oriented language used to specify the characteristics of the printing environment on the Xerox 9700. PDL source statements are coded by the user to define job tape formats, processing features, and output formats to be used in the printing of tapes. These PDL coded statements describing a particular set of jobs to be run are called a Job Descriptor Library (JDL). Within the JDL there may be one or more unique definitions for different tape formats, processing features, and output formats. These definitions represent a job, and are individually called Job Descriptor Entries (JDE).

A source statement Job Descriptor Library is compiled by the PDL processor (a system task) and a JDL control file is written to system disk. To print a particular job tape, the operator will request via a START command the JDL control file and the desired JDE within it.

This chapter defines the syntax of the PDL statements and provides a summary and index to all PDL commands. It also illustrates how Job Descriptor Libraries are created, compiled and initiated.

Format and Procedures for Coding PDL Statements

Command Format

Figure 3-1 illustrates the composition of a typical PDL statement. PDL statements consist of records of up to 133 characters, of which only characters 1-72 may be used for command information. Records of less than 72 characters are acceptable. Each PDL statement consists of a command keyword, and one or more left/right parts. Within a statement's left/right parts commas (or spaces) are used for separators.

Commands may be continued on successive records. Cross over is performed from one record to the next when column 73 is reached or at the end-of-record for records of less than 72 characters. Multiple commands may appear on one record if separated by semicolons.

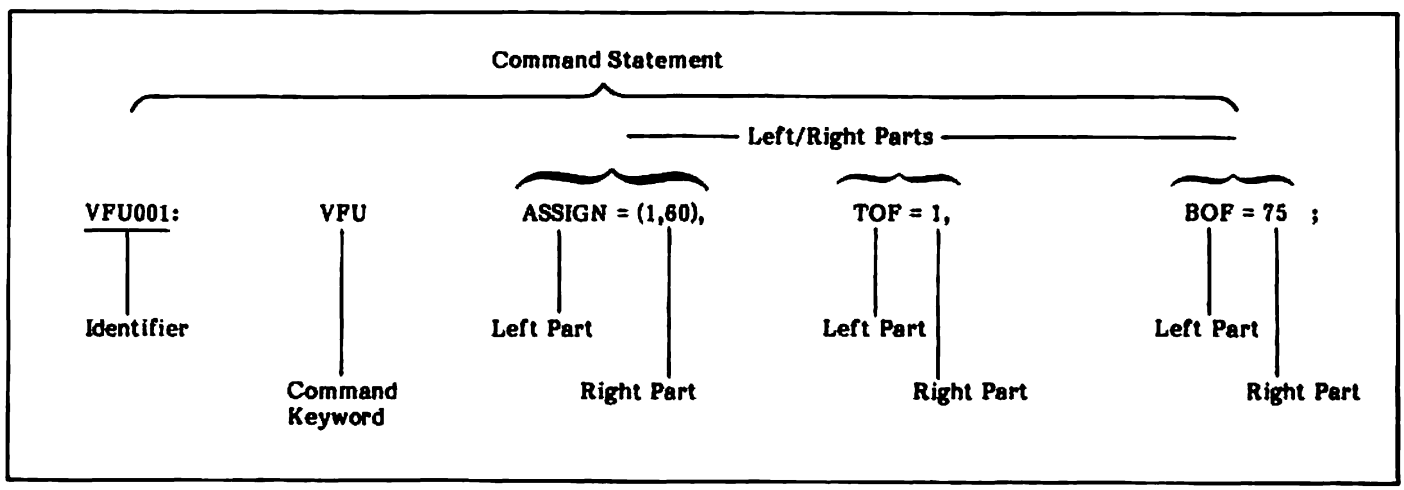


Figure 3-1. Print Description Language Statement Format

IDENTIFIERS

Some commands have identifiers for reference by other commands. Statement identifiers have one of two formats, depending upon the command with which the identifier is associated. All identifiers will consist of 1 to 6 alphanumeric characters (A-Z and 0-9) followed by a colon (:). At least one of the characters must be an alphabetic character except in the case of the SYSTEM (or JDL) and JOB (or JDE) commands which may have all numeric identifiers.

Identifiers are shown in the PDL statement Summary (Table 3-1) as 'ac' (alphanumeric characters with at least 1 alpha character) and 'dd' (alphanumeric characters with no restriction of having one alpha character).

COMMANDS

Every statement must have a command. A command must be written as shown in the PDL Statement Summary (Table 3- 1).

LEFT/RIGHT PARTS

Left/Right parts are permitted in most statements. as shown in the PDL Statement Summary (Tables 3-1 and 3-2). With the exception of logical processing statements, left/right parts are optional; they may be written in any order within the statement. A set of left/right-parts consists of a left part, an equals (=) sign, and one or more right parts.

Left Parts. Left parts are always keywords, which must be written as shown in the PDL Statement Summary (Table 3-1).

Right Parts. Right parts may consist of keywords, identifiers, value constants, string constants, or combinations of these various parts. If more than one right part is associated with a left part, the right parts must be separated by commas and enclosed in parenthesis. See the PDL Statement Summary (Table 3-1) for the permissible combinations of right and left parts for a given command. Keywords used as right parts must be written as shown.

Identifiers used as right parts refer to statement identifiers defined in preceding statements. For example:

```
Statement Identifier
     |
VFU001: VFU BOF = 66;
        LINE VFU =  VFU001;
                    |
                 Right Part
```

Value constants are constants which have an arithmetic value. They should be expressed as decimal numbers. They may be expressed as hexadecimal values, octal values. or even character values, but these expressions are not recommended.

String constants are constants which are normally used to specify strings of characters. The width of string constants is important. String constants may be expressed as hexadecimal values, octal values, or character values, but not as decimal numbers.

String constants may be preceded by an optional repeat count. A repeat count is enclosed in parentheses and must be in the range of 1 to 255. The repeat count itself may be a hexadecimal, octal, or character constant, but a decimal constant is recommended. for example, the statement:

T1: TABLE CONSTANT = (3)'*'; is equivalent to: T1: TABLE CONSTANT = ('***');

Other examples of the use of a repeat count are:

```
T1: TABLE CONSTANT = (3)O'27';
T2: TABLE CONSTANT = (4)X'C1';
```

PDL CONSTANTS

Decimal. Decimal constants are used whenever an arithmetic value is to be specified. They may be used only when defining value constants. Leading zeros, where coded, are ignored. Decimal constants may be signed and in some cases may have fractional digits.

For example:

```
OUTPUT COPIES = 1,
OUTPUT COPIES = 01,        (equivalent decimal constant)

RECORD OFFSET = +60,   ADJUST = -50;
                 |             |
                 Decimal Constants
```

Hexadecimal. Hexadecimal constants are normally used as string constants, but they may also be used as value constants. A hexadecimal constant must be immediately preceded by the characters X apostrophe (X'), and immediately followed by the apostrophe (') character.

For example:

```
                    Hexadecimal Constant
                   ~~~~~~~~~~~~~~~~~~~~
IDEN PREFIX =      X'C1C2C3C4';
```

Character. Character constants are normally used as string constants, but they may also be used as value constants. A character constant must be immediately preceded and immediately followed by the apostrophe (') character.

For example:

```
                              Character Constant
                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
IDEN PREFIX =      'THIS IS A CHARACTER CONSTANT' ;
```

If the apostrophe character (') is required in a character constant, it must be defined in some other fashion, such as the hexadecimal constant X '7D'.

When creating character strings in upper case (as with the 9700 EDITOR) the "#" character can be used as an upper/lower case toggle switch. When a "#" character is encountered, the lower case mode is invoked and all letters thereafter will be considered lower case until another "#" character is encountered. Two successive "#" characters (##) are necessary to represent one actual "#" character.

Octal. Octal constants should be used only as string constants, because of the control program conversion process. Their use as value constants is not prohibited, however. Each 6-bit octal character is converted to an 8-bit octal character, internally, by prefixing two binary zeros. Thus, the arithmetic value of a multiple-character octal constant may be difficult to determine because each digit in the constant has been altered. An octal constant must be immediately preceded by the characters O apostrophe (O'), and immediately followed by the apostrophe (') character.

For example:

```
                        Octal Constant
                        ~~~~~~~~~~~~~~
IDEN PREFIX =  O'21222324' ;
```

Points to Note

If string constants are used as value constants, the width of the string constant (as defined by the number of characters between the leading and trailing apostrophe (') characters) must be 16 bits or less. Therefore, character constants used as value constants must be two characters or less; hexadecimal or octal constants must be four characters or less. Thus, for example, the statement:

```
BLOCK LTHFLD = X'12345';
```

is invalid and produces an error message.

The following statements are equivalent:

```
OUTPUT COPIES = 193;
OUTPUT COPIES = X'C1';
OUTPUT COPIES = 'A';
```

Statement Format

The Print Description Language (PDL) records may consist of up to 133 characters. Columns 1 through 72 of a record are used for PDL statements. Columns 73 through 133 may be used for identification or sequence information; however, this type of information is not processed by PDL or shown on PDL listings. Statements may span as many records as required.

Keywords may not be continued from one record to the next. That is, they must be completely contained within one record. Constants and identifiers may be continued from one record to the next. However, if a constant or identifier does not end at column 72 of one record, it must continue in column 1 of the next record.

It is necessary to specify a statement only if changes to the system defaults associated with a given statement are desired. Within a statement, it is only necessary to specify the left/right parts that represent changes to the associated system defaults. Thus, the statements

```
OUTPUT;
OUTPUT COPIES = 10;
OUTPUT COPIES = 10,  COLLATE = YES;
```

are all valid forms of the OUTPUT command.

Statement Syntax

A statement identifier, where required, must be followed by the colon symbol (:). For example:

```
Identifier
TABLE1:  TABLE CONSTANT =('//$$');
```

Commands and keywords must be written as shown. No embedded blanks are permitted. The equals symbol (=) is used to join left parts to right parts; it is required where shown. Commas are used to separate right parts as well as sets of left/right parts. Blanks may be used to separate left/right parts. Parentheses are used to enclose multiple right parts and are required where shown. Superfluous pairs of parentheses are permitted. Right parts of a multiple right part set must be in the order shown. For example:

```
LINE OVERPRINT = (IGNORE,NODISP);      (valid statement)
LINE OVERPRINT = (NODISP,IGNORE);      (invalid statement)
```

Annotational comments are allowed, and they may appear anywhere within the job descriptor library. They must be preceded by the character sequence slash and asterisk (/*) and terminated by the character sequence asterisk and slash (*/).

Blanks may appear anywhere except in commands, keywords, hexadecimal or octal constants. Blanks terminate identifiers. For example:

```
V1  :VFU BOF = 65  ;
LINE PCC = (10,NOTRAN), VFU = V1  ;
```

A semicolon (;) must be used to explicitly terminate a statement. Multiple statements may be contained within one record, but each statement must be terminated by a semicolon.

PDL Conventions

The following conventions are used in the PDL Statement Summary (Tables 3-1 and 3-2).

1. Lower case letters identify an item that must be replaced by a user selected value.

2. Keywords in capital letters must be entered as shown.

3. An element in brackets [ ] is optional.

4. Elements placed inside a pair of braces { } indicate a required choice.

5. Ellipsis inside parentheses indicate that the element preceding the ellipsis may be repeated.

PDL Statement Summary

Tables 3-1 and 3-2 can be used as a quick reference to PDL statement syntax. Each PDL statement is discussed in detail on the page shown in the table column titled "Ref. Page".

The following definitions describe the symbols used in the PDL summary tables.

1. ac = statement identifier. Consists of 1 to 6 alphanumeric characters (A-Z and 0-9) followed by a colon (:). One of the characters must be an alpha.

2. dd = statement identifier. Same as 'ac' except restriction of having one alpha character is removed. This only applies to SYSTEM and JOB commands.

3. sc = string constant. A hexadecimal, octal or character constant.

4. value = constant. It is preferably a decimal constant, but it may also be a string constant.

5. ccln = ({ TOF / OVR / IGN } , ck) or ccln = ck,

   where ck is a compound keyword and must contain at least one of the following elements (in brackets);

   [SPm / SKn] [P / N] [SPm / SKn]

   For example:

   ```
   PCC DEFAULT = (OVR,SP1PSK2);
   PCC DEFAULT = PSP3;
   PCC DEFAULT = SK4N;
   PCC DEFAULT = SP5SP6;
   PCC DEFAULT = SP2;
   PCC DEFAULT = P;
   ```

6. id = a reference to a statement identified by an "ac" statement identifier.

Table 3-1. PDL Statement Summary

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
dd: {SYSTEM / JDL} ;				Command set used to identify the beginning of a job descriptor library.	3-17
ac: CATALOG;				Command set used to group statements which are to be included in more than one job descriptor entry.	3-18
dd: {JOB / JDE}	[INCLUDE =	(id_1, ..., id_n)] ;		Start of job descriptor entry. "INCLUDE" specified when using catalog command set. "id" refers to the 'ac' identifier of a CATALOG command set.	3-18
END;				Command used to terminate a job descriptor library.	3-19
[ac:] CODE				Command used to assign an input-code to output-code correspondence.	4-14
	DEFAULT =	{ ASCII / BCD / BCL / CBCD / EBCDIC / PEBCDIC / H2BCD / H6BCD / IBMBCD / value } ,	EBCDIC		4-14
	ASSIGN =	{(input,output), / (input,(output, ..., output))};			4-14
[ac:] PCC				Command used to assign one-byte printer carriage control codes and define their actions.	5-13
	INITIAL =	{BOF / TOF} ,	TOF		5-13
	DEFAULT =	{ccln / pcctype} ,	PSP1		5-13
	MASK =	value,	X'FF'		5-15

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
	ADVTAPE =	{ YES / NO } ,	YES		5-15
	ASSIGN =	{ (byte, ccln) / (byte, (ccln, ..., ccln)) } ;			5-14
VOLUME				Command used to specify parameters which describe the format of the job tape.	
	HOST =	{ ANSI, B2500, B2700, B3500, B3700, B4700, B6700, DUMP, GRASP, H2000, H6000, IBMOS, IBMDOS, OCTDUMP, OSWTR, POWER, POWERVS, UNIVAC, US70, XEROX } ,	IBMOS		4-9
	UNPACK =	{ H6000, T4X3, T4X3H2, UNIVAC, NONE } ,	NONE		4-9
	LABEL =	{ ANSI, COBOL, NONE, SPR, STANDARD } ,	STANDARD		4-9
	CODE =	{ ASCII, BCD, BCL, CBCD, EBCDIC, H2BCD, H6BCD, IBMBCD, PEBCDIC, USER, id } ,	EBCDIC		4-10

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
	LCODE =	{ ASCII | BCD | BCL | CBCD | EBCDIC | H2BCD | H6BCD | IBMBCD | PEBCDIC | USER | id },	EBCDIC		4-10
	EOV =	({ PAUSE | NOPAUSE } , { EOF | NOEOF }),	(NOPAUSE, NOEOF)		4-10
	PLABEL. =	{ YES | NO } ,	NO		4-11
	OSCHN =	value,	9	OSCHN, OSHDP, and OSTLP are meaningful only for HOST = OSWTR.	4-11
	OSHDP =	value,	0		
	OSTLP =	value,	0		
	BMULT =	value,	1		4-12
	RMULT =	value,	1		4-12
	RMODE =	{ S | M } ;	M		4-12
BLOCK				Command used to specify the parameters which describe the physical structure of the tape blocks.	
	LENGTH =	value,	1330		4-15
	LTHFLD =	size,	0		4-15
	OFFSET =	value,	0		4-15
	FORMAT =	{ BIN | DEC | PACK | PKSG } ,	BIN		4-15
	ADJUST =	value,	0		4-15
	PREAMBLE =	length,	0		4-16
	POSTAMBLE =	length,	0		4-16
	CONSTANT =	sc,	0		4-16
	ZERO =	{ YES | NO } ,	NO		4-16
	LMULT =	value;	1		4-16

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
RECORD				Command used to specify parameters which describe the characteristics of the logical tape records.	
	STRUCTURE =	{ F / FB / U / UB / V / VB } ,	FB		4-18
	LENGTH =	value,	133		4-18
	LTHFLD =	size,	0		4-18
	OFFSET =	value,	0		4-18
	FORMAT =	{ BIN / DEC / PACK / PKSG } ,	BIN		4-18
	ADJUST =	value,	0		4-19
	PREAMBLE =	length,	0		4-19
	POSTAMBLE =	length,	0		4-19
	CONSTANT =	sc,	0		4-19
	LMULT =	value;	1		4-19
LINE				Command used to structure output lines.	
	MARGIN =	{ value / (value { b / , } { IN / CM / POS }) } ,	(1, POS)		5-7
	DATA =	(offset, length),	(1, 132)		5-7
	PCC =	(offset, { TRAN / NOTRAN }),	(0, NOTRAN)		5-8
	PCCTYPE =	{ ANSI / B2500 / B2700 / B3500 / B3700 / B4700 / B6700 / H2000 / H6000 / IBM1401 / IBM1403 / NONE / UNIVAC / USER / US70 / XEROX / id } ,	ANSI		5-8

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
	OVERPRINT =	({ PRINT / MERGE / IGNORE } , { DISP / NODISP }),	(PRINT, NODISP)		5-9
	VFU =	vfu-id,	none		5-9
	FONTINDEX =	{ offset / NONE } ;	NONE		5-9
OUTPUT				Command used to specify the copy count and collate/ uncollate mode, CMEs and offset control.	
	COPIES =	value,	1		5-2
	COLLATE =	{ YES / NO } ,	YES		5-2
	MODIFY =	{ cme-id / (cme-id, init [,copies]) / NONE },	NONE		5-2
	OFFSET =	{ ALL / FIRST / NONE } ,	ALL		5-2
	COVER =	{ FRONT / (FRONT, SEP) / BACK / BOTH / (BOTH, SEP) / NONE } ,	NONE		5-3
	FORMAT =	pde-id,	FMT1		5-4
	NUMBER =	{ (pnum, lnum, cnum) / NO } ,	NO		5-5
	FORMS =	{ form-id / (form-id, init[,copies]) / NONE } ;	NONE		5-4
[ac:] VFU				Command used to specify the line position to be associated with each channel used in the carriage control convention (PCCTYPE left part in LINE command) and the top- and bottom-of-form physical margins.	
	ASSIGN =	{ (channo, lineno) / (channo, (lineno,..., lineno)) },			5-10

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
	TOF =	value,	1		5-11
	BOF =	value;	66		5-11
ACCT				Command to request an accounting summary and to specify where the summary will be delivered.	5-29
	USER =	{ NONE BIN TRAY (BIN,TRAY) (TRAY, BIN) },	BIN		5-29
	DEPT =	sc;	none		5-29
IDEN				Command used to identify a dynamic job descriptor entry.	
	PREFIX =	sc,	none		6-4
	SKIP =	value,	1		6-4
	OFFSET =	value,	0		6-4
	OPRINFO =	{ YES NO } ;	NO		6-4
ABNORMAL				Command used to enhance output integrity. Abnormal conditions reported, forces job abort for specified status code and imposes operator restrictions on system use.	5-31
	REP =	{ YES NO } ,	NO		5-32
	RES =	sev-level,	9999		5-31
	BLKSP =	{ YES NO } ,	YES		5-31
	SECURITY =	{ YES NO } ;	NO		5-31

Table 3-1. PDL Statement Summary (Con't)

Identifier/Command	Left Part	Right Parts	Default	Comments	Ref. Page
ac: CME				Command used to specify a copy modification entry that is referenced by the MODIFY portion of the OUTPUT command.	5-17
	LINE =	{ n / (n, m) / (n, -) } ,	none		5-17
	POS =	p,	1		5-17
	CONSTANT =	sc,	none		5-18
	FONT =	value;	1		5-18
ac: PDE				Command used to identify the Page Descriptor Entry which describes formatting information for each page of the report.	5-21
	PMODE =	{ PORTRAIT / LANDSCAPE } ,	LANDSCAPE		5-21
	FONTS =	{ (f1 [,f2] ...), / ((f1, s1) [,(f2, s2)] ...) } ,	none		5-21
	BEGIN =	(vpos { CM / IN }, hpos { CM / IN });	(0, 0)		5-22
MESSAGE				Command used to cause a message to be displayed to the operator during job processing or a job printing.	5-26
	ITEXT =	{ sc / (sc ,passnum) } ,	none		5-26
	OTEXT =	{ sc / (sc [,passnum][,WAIT]) } ;	none		5-26
ROUTE				Command to print a message on a separate sheet of paper preceding a report ply or an entire report.	5-27
	RTEXT =	{ sc / (sc ,passnum [,line [,col]]) } ,	none		5-27
	RFORM =	form-id;	none		5-27

Table 3-2. Logical Processing Command Summary

Function	Command	Left Part Right Parts	Default	Comments	Ref. Page
Value Tables	ac: TABLE	CONSTANT = (sc, ..., sc);		Command to build a table of constants for use by the logical processing functions.	4-37
Constant Criteria Tables	ac: CRITERIA	CONSTANT = (offset, length, $\left\{\begin{matrix}\text{EQ}\\\text{NE}\end{matrix}\right\}$, id_2);		Defines test of subfield of a record or block against entries in a value table. The subfield is located at offset "offset" and is of length "length". id_2 is the name of the value table. If EQ is coded, the criteria is satisfied if the subfield matches one of the entries in the table. If NE is coded, the criteria is satisfied if the subfield does not match any of the table entries.	4-22
Change Criteria Tables	ac: CRITERIA	CHANGE = (offset, length, NE, LAST);		Defines a subfield of each record or block which is to be compared with the same subfield of the previous record or block. "offset" and "length" are as above. The criteria is satisfied if the subfield values from the records are not equal.	4-22
Record Selection	RSELECT	TEST = $\left\{\begin{matrix}id_1\\(id_1, \left\{\begin{matrix}\text{AND}\\\text{OR}\end{matrix}\right\}, id_2)\end{matrix}\right\}$;		Defines test expression for selecting records for printing. id_1 and id_2 may be either change-mode or constant-mode CRITERIA statements. If only id is present, then the test is satisfied if the record satisfies the criteria in id_1. If id_1 and id_2 are both present and AND is coded, the test is satisfied only if the record satisfies the criteria in both id_1 and id_2. If OR is coded, the test is satisfied if the record satisfies the criteria in either id_1 or id_2. If the test is satisfied, the record is selected for printing.	4-26

Table 3-2. Logical Processing Command Summary (Con't)

Function	Command	Left Part	Right Parts	Default	Comments	Ref. Page
Record Deletion	RDELETE	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$;		id_1, id_2, AND, and OR as above. If the test is satisfied, the record is deleted from the printed output.	4-26
Block Selection	BSELECT	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$;		id_1 id_2, AND, and OR as above. If the test is satisfied, the block is selected for processing.	4-24
Block Deletion	BDELETE	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$;		id_1, id_2, AND, and OR as above. If the test is satisfied, the block is deleted from the printed output.	4-24
Offset Control	ROFFSET	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$,		id_1, id_2, AND, and OR as above. If the test is satisfied, the output will be offset in the stacker bin.	4-28
		PASSES =	$\left\{ \begin{array}{l} FIRST \\ ALL \end{array} \right\}$;	ALL	If FIRST is coded, the record will cause an off-set only on the first pass of a collated print run. If ALL is coded, offsetting will occur in every print pass.	4-28
Suspend Printing	RSUSPEND	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$,		id_1, id_2, AND, and OR as above. If the test is satisfied, printing will be suspended until a record satisfying the criteria for an RRESUME command is encountered.	4-30
		BEGIN =	$\left\{ \begin{array}{l} CURRENT \\ NEXT \end{array} \right\}$;	NEXT	If CURRENT is coded, the record satisfying the suspension criteria is not printed. If NEXT is coded, the record satisfying the criteria is printed and "non-printing" begins with the following record.	4-30
Resume Printing	RRESUME	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$,		id_1, id_2, AND, and OR as above. If test is satisfied and printing has been suspended, then printing will be resumed.	4-31

Table 3-2. Logical Processing Command Summary (Con't)

Function	Command	Left Part Right Parts	Default	Comments	Ref. Page
		BEGIN = { CURRENT / NEXT } ;	NEXT	If CURRENT is coded, the record satisfying the resumption criteria is printed. If NEXT is coded, printing is resumed beginning with the following record.	4-31
Delimiter Definition	RSTACK	TEST = { id_1 / (id_1, { AND / OR }, id_2) },		id_1, id_2, AND, and OR as above. If the test is satisfied, an end-of-report condition has been encountered.	4-34
		DELIMITER = { YES / NO },	NO	If YES is coded, this record and all subsequent records which satisfy the test constitute a delimiter packet. This delimiter packet is removed from the report and can only be printed on the delimiter page as controlled by the left part PRINT below. If NO is coded, the record is the first record of the next page.	4-34
		PRINT = { NONE / BIN / TRAY / BOTH },	NONE	Defines the destination(s) of printed copies of the delimiter page, provided that DELIMITER = YES. If DELIMITER = NO, no page will be printed.	4-34
		ACCTINFO = (offset, length);	(0, 0)	If coded, specifies that the subfield beginning at offset "offset" and for a length of "length" in the FIRST (or only) delimiter record (or the first data record if DELIMITER = NO) is to be saved for printing on the accounting log at the end of the report. "length" is limited to a maximum value of 128. The default is (0, 0); that is, no accounting information is to be extracted and printed.	4-35

Coding a Job Descriptor Library

Introduction

To simplify the coding of Print Description Language statements in the creation of a Job Descriptor Library (JDL), the statements may be grouped into Command Sets. The three types of Command Sets are

SYSTEM Command Set
CATALOG Command Set
JOB Command Set

which are defined below.

SYSTEM Command Set

A SYSTEM[t] Command Set must appear first and only once in a Job Descriptor Library. It defines installation dependent requirements and default values for Job Descriptor Entries. Each SYSTEM command results in the creation of a Job Descriptor Library. No other PDL statements may be placed on the SYSTEM statement.

A SYSTEM Command Set is terminated by the detection of another SYSTEM (or JDL) command, a CATALOG command, a JOB (or JDE) command, an END command, or physical end-of-input. Detection of an END command, another SYSTEM command, or physical end-of-input causes the PDL task to stop building the current JDL. Detection of two consecutive END commands is interpreted as end-of-input.

The command has the form:

jdl-name: {SYSTEM / JDL};

where

jdl-name is a 1-6 character identifier specifying the name of the JDL to be created. It may be numeric or alphanumeric.

For example:

IBMDOS: SYSTEM;

identifies the start of a SYSTEM command set and the beginning of a JDL. The identifier corresponds to the name used by the operator (via the START command on the keyboard/display) to indicate the Job Descriptor Library to be used when printing a tape.

The default values for job descriptor entries are specified via the same statements which may appear in the CATALOG and JOB command sets; however, when a statement appears in the SYSTEM command set, it establishes a default value which may be overridden by other usage of the statement within the JDL. This system of overriding parameters is discussed in the section of this chapter entitled "Hierarchy of Replacement".

---

[t] JDL may be used interchangeably with SYSTEM.

CATALOG Command Set

A CATALOG command set defines a group of statements which may be subsequently referenced by one or more JDEs for the convenience of the programmer writing the Job Descriptor Library. A CATALOG command set is identified by the CATALOG statement and ends with the appearance of another CATALOG statement or a JOB statement. CATALOG command sets may contain the same statements as appear in JOB command sets.

The CATALOG command has the form

cat-name: CATALOG;

where

cat-name is a 1-6 character alphanumeric identifier of which one character must be alpha. The name will be referenced by JDEs following the CATALOG set.

For example, the first statement in a CATALOG Command Set might be:

POWER: CATALOG;

where POWER is the CATALOG identifier used in the INCLUDE left/right pair of the JOB (or JDE) statement.

JOB Command Set

The JOB[t] (or JDE) command set defines how particular print tapes are to be processed. These command sets are identified by the JOB (or JDE) command and have the form:

jde-name: $\left\{ \begin{matrix} \text{JOB} \\ \text{JDE} \end{matrix} \right\}$ [INCLUDE = (cat-name-1, ... ,cat-name-n)] ;

where

jde-name is a 1-6 character identifier specifying the name of the JDE being defined. It may be numeric or alpha-numeric.

cat-name-i is a 1-6 character identifier of a previously defined Catalog Command Set. It must contain at least one alpha character.

---

[t] JDE may be used interchangeable with JOB.

Examples:

```
JOB3: JOB;
or
JOB3: JOB INCLUDE = (POWER);
and
2001: JOB INCLUDE = (POWER, POWERA);
```

are all valid JOB commands.

The JOB command set continues until another JOB statement or an END statement is encountered. The identifier in a JOB command set (JOB3 or 2001 in above examples) is used by the operator (along with the identifier on the SYSTEM command set) to initiate a print job from the keyboard/display.

Since the JOB command set specifies values for job descriptors, directly references CATALOGs, and indirectly includes SYSTEM defaults, it is considered to be a complete job description; hence, the name Job Descriptor Entry.

END Statement

A JDL terminates with the following statement:

```
END;
```

If one Job Descriptor Library is to follow another, then the next statement should be a SYSTEM command. The end of all JDLs to be processed is indicated by two consecutive END statements.

Figure 3-2 provides an overview of a Job Descriptor Library illustrating the use of the SYSTEM, CATALOG and JOB command sets. Figure 3-3 is an example of an actual coded Job Descriptor Library.

```
dd:  SYSTEM;/*      dd IS A 1 TO 6 CHARACTER ALPHANUMERIC THAT IDENTIFIES THIS PARTICULAR JOB
                    DESCRIPTOR LIBRARY.                                                    */

STATEMENT;
STATEMENT;          /* A 'STATEMENT' HERE MEANS ANY PDL COMMAND                           */
STATEMENT;

/*                  THE THREE STATEMENTS ABOVE ARE USED TO CHANGE THE DEFAULTS FOR THIS CON-
                    FIGURATION.                                                            */

CATA: CATALOG;
    STATEMENT;      /*THESE THREE STATEMENTS WILL BE CATALOGED UNDER THE NAME 'CATA'. THEY MAY
    STATEMENT;      BE USED IN ANY JDE IN THIS SYSTEM BY USE OF THE NAME CATA              */
    STATEMENT;

CATB: CATALOG;
    STATEMENT;
    STATEMENT;      /*A CATALOG IDENTIFIER MUST BE A 1 TO 6 CHARACTER ALPHANUMERIC.      */
    STATEMENT;

CATC: CATALOG;
    STATEMENT;

CATD: CATALOG;
    STATEMENT;      STATEMENT; /* MULTIPLE STATEMENTS ON ONE CARD ARE VALID              */

JOB1:  JOB INCLUDE = (CATA, CATB, CATC);

STATEMENT;          /*JOB1 IS THE NAME OF THIS JDE.  IT IS MADE UP OF CATALOG SETS CATA, CATB, AND
STATEMENT;          CATC AND THE TWO STATEMENTS THAT FOLLOW                                */

JOB2:  JOB INCLUDE = (CATA, CATB, CATD);/* JOB2 IS TOTALLY DEFINED BY CATALOG SETS, CATA, CATB,
                      AND CATD                                                             */

2001:  JDE;         /*AN IDENTIFIER MAY BE ALL NUMERIC.  JDE CAN BE USED INSTEAD OF JOB    */

STATEMENT;
STATEMENT;          /*JOB 2001 DOES NOT USE CATALOGED COMMAND SETS*/
STATEMENT;

JOB10: JOB;         /*JOB10 IS MADE UP OF THE DEFAULTS FOR THIS JOB DESCRIPTOR LIBRARY*/

END;                /*THIS TERMINATES THE JDL*/
```

Figure 3-2. Overview of PDL Coding

```
IBMPDL: SYSTEM;

/*                  THIS JOB DESCRIPTOR LIBRARY CONTAINS JOB DESCRIPTOR ENTRIES FOR PROCESSING
                    GRASP, POWER AND POWERVS JOB TAPES


                    THE SYSTEM COMMAND SET DEFINES CONSTANTS AND PROCESSING PROCEDURES THAT
                    WILL APPLY TO ALL JOBS PROCESSED USING THIS LIBRARY UNLESS OVERRIDDEN BY
                    THE CATALOG OR JOB COMMAND SETS.
                                                                                        */

VFU001:  VFU        ASSIGN = (1,5), ASSIGN = (2,10), ASSIGN = (3,15),
                    ASSIGN = (4,20), ASSIGN = (5,25), ASSIGN = (6,30),
                    ASSIGN = (7,35), ASSIGN = (8,40), ASSIGN = (9,45),
                    ASSIGN = (10,50), ASSIGN = (11,55), ASSIGN = (12,60),
                    TOF = 5, BOF = 66;

/*                  TABLES AND CRITERIA FOR LOGICAL PROCESSING FOR GRASP INTERLEAVED TAPES
                                                                                        */

T1:      TABLE      CONSTANT = ('B');
T2:      TABLE      CONSTANT = ('C');


C1:      CRITERIA   CONSTANT = (27, 1, EQ, T1);
C2:      CRITERIA   CONSTANT = (27, 1, EQ, T2);

/*
                    SYSTEM FOR POWER VS
                                                                                        */

         VOLUME     HOST = POWERVS, PLABEL = YES;
         BLOCK      LENGTH = 2048;
         RECORD     LENGTH = 136, STRUCTURE = VB, LTHFLD = 2, OFFSET = 0,
                    ADJUST = 0, FORMAT = BIN, PREAMBLE = 3;
         LINE       DATA = (1, 132), PCCTYPE = IBM1403, PCC = (0, NOTRAN),
                    OVERPRINT = (PRINT, NODISP), VFU = VFU001;
         ACCT       USER = (BIN, TRAY);


/*                  CATALOG COMMAND SET FOR POWER VERSIONS

                    A CATALOG COMMAND SET IS USED TO GROUP A SERIES OF COMMANDS THAT MIGHT BE
                    INCLUDED IN MORE THAN ONE JOB COMMAND SET (JOB DESCRIPTOR ENTRY). CATALOG
                    COMMAND SETS MUST PRECEDE THE JOB COMMAND SETS.
                                                                                        */

CATPOW:CATALOG;
         VOLUME     HOST = POWER;
         BLOCK      LENGTH = 2048, PREAMBLE = 6, LTHFLD = 2, FORMAT = BIN, OFFSET = 4;
         RECORD     LENGTH = 135, STRUCTURE = VB, PREAMBLE = 2, LTHFLD = 2,
                    FORMAT = BIN, OFFSET = 0, ADJUST = 3;

/*                  CATALOG COMMAND SET FOR GRASP */

CATGRP:CATALOG;
         VOLUME     HOST = GRASP;
         BLOCK      LENGTH = 4096, PREAMBLE = 0, ZERO = YES;
         RECORD     LENGTH = 135, STRUCTURE = VB, PREAMBLE = 1,
                    LTHFLD = 1, FORMAT = BIN, OFFSET = 0, ADJUST = 2;
         RSTACK     PRINT = TRAY;

/*                  A JOB COMMAND SET INDICATES THE BEGINNING OF A JOB DESCRIPTOR ENTRY. THE
                    SYNTAX "INCLUDE = (NN)", MEANS TO INCLUDE THE CATALOG COMMAND SET LABELED
                    "NN" AS PART OF THE JOB DESCRIPTOR ENTRY.                           */
```

Figure 3-3. Job Description Library, Programming Example

```
/*                    POWER VS, POWER 4.0, AND POWER 4.1/4.2

                THE FOLLOWING JDES PROVIDE SUPPORT FOR IBM POWER VS TAPES, POWER VERSION 4.0
                TAPES, AND POWER VERSIONS 4.1/4.2 TAPES

                CHARACTERISTICS                    JDE #

                POWER VS TAPES                       1
                POWER VERSION 4.0 TAPES              2
                POWER VERSION 4.1/4.2 TAPES          3                                    */

1:JOB;
                VOLUME    HOST =POWERVS;
2:JOB           INCLUDE = (CATPOW);
                VOLUME    HOST = POWER;
                RECORD    LTHFLD = 1, PREAMBLE = 1, ADJUST = 2;
3:JOB           INCLUDE = (CATPOW);
                VOLUME    HOST = POWER;

/*                              IBM DOS GRASP TAPES

                THE FOLLOWING JDES PROVIDE SUPPORT FOR GRASP TAPES FOR 1600 BPI TAPES.

                CHARACTERISTICS                                    JDE #

                NORMAL GRASP TAPES                                   21
                GRASP WITH INTERSPERSED REPORTS - SELECT             22
                  REPORTS FROM PHANTOM DEVICE B
                GRASP WITH INTERSPERSED REPORTS - SELECT             23
                  REPORTS FROM PHANTOM DEVICE C
                                                                                         */

21:JOB          INCLUDE = (CATGRP);
                VOLUME    HOST = GRASP;
22:JOB          INCLUDE = (CATGRP);
                VOLUME    HOST = GRASP;
                BSELECT   TEST = (C1);
23:JOB          INCLUDE = (CATGRP);
                VOLUME    HOST = GRASP;
                BSELECT   TEST = (C2);

END;
```

Figure 3-3. Job Descriptor Library, Programming Example (Continued)

Hierarchy of Replacement

Introduction

The system job descriptor entry default values shown in Tables 3-1 and 3-2 are the more commonly used values in job tape processing; they can be thought of as a "basic" job descriptor entry. As stated previously, the user needs to code PDL statements only for those parameters which must be changed to process the installation's job tapes. Likewise, this coding process may be further simplified by placing parameters common to more than one job in a CATALOG command set. When these coding features are properly implemented, it is possible for the same statement to be used in more than one command set within a library. The PDL processor evaluates these multiple statements and applies the "highest order", error-free definition to the job tape being printed. This process, which is termed the hierarchy of replacement, is demonstrated in the subsequent paragraph.

Statement Processing

Figure 3-4 shows a coded job descriptor library which contains four JOB command sets (job descriptor entries). Note that a statement to specify the recording code of the input tape appears in three places:

1. According to the SYSTEM command set, the recording code of the input tape is ASCII

   VOLUME CODE = ASCII;

2. According to the CATALOG command set, the recording code of the input tape is EBCDIC

   VOLUME CODE = EBCDIC;

3. According to the JOB command sets for jobs one and three, the recording code of the input tape is Printable EBCDIC (PEBCDIC). The PDL statement

   VOLUME CODE = PEBCDIC;

   overrides both the CATALOG and SYSTEM command set definitions.

   For job two, the recording code of the input tape is EBCDIC, as specified in the CATALOG command.

   For job four, the recording code of the input tape is ASCII, since neither the CATALOG nor JOB command set overrides have been coded.

```
EXAMP2: SYSTEM;
            VOLUME CODE = ASCII;
CATAA:  CATALOG;
            VOLUME CODE = EBCDIC;
            OUTPUT COPIES = 100;
            LINE DATA = (2, 130);
JOB1:  JOB INCLUDE = (CATAA);
            VOLUME CODE = PEBCDIC;
JOB2:  JOB INCLUDE = (CATAA);
            OUTPUT COPIES = 50;
JOB3:  JOB INCLUDE = (CATAA);
            VOLUME CODE = PEBCDIC;
JOB4:  JOB;
            OUTPUT COLLATE = NO;
END;
```

Figure 3-4. Hierarchy of Replacement Example

The last error-free PDL statement that is encountered ("highest order" definition) will be applied to job tapes processed under a given JDE in a Job Descriptor Library.

The next level of parameter replacement (see Figure 3-5) above the JOB command set is the Dynamic Job Descriptor Entry (DJDE) command file (see "Command File DJDEs" in Chapter 6) which is initiated via the START command (see "Starting a Job" in Chapter 7). Parameters coded within the command file in DJDE format override parameters within the Job Descriptor Library.

If the job tapes to be processed are labeled tapes, then the appropriate tape label information may override programmed values in the Job Descriptor Library. To determine which specific tape label fields are applied, refer to the appropriate vendor format in Xerox Tape Formats Manual No. 910004.

Final modification can be made to job descriptor parameters via DJDE commands contained on the job input tape. These DJDE commands are discussed in Chapter 6.

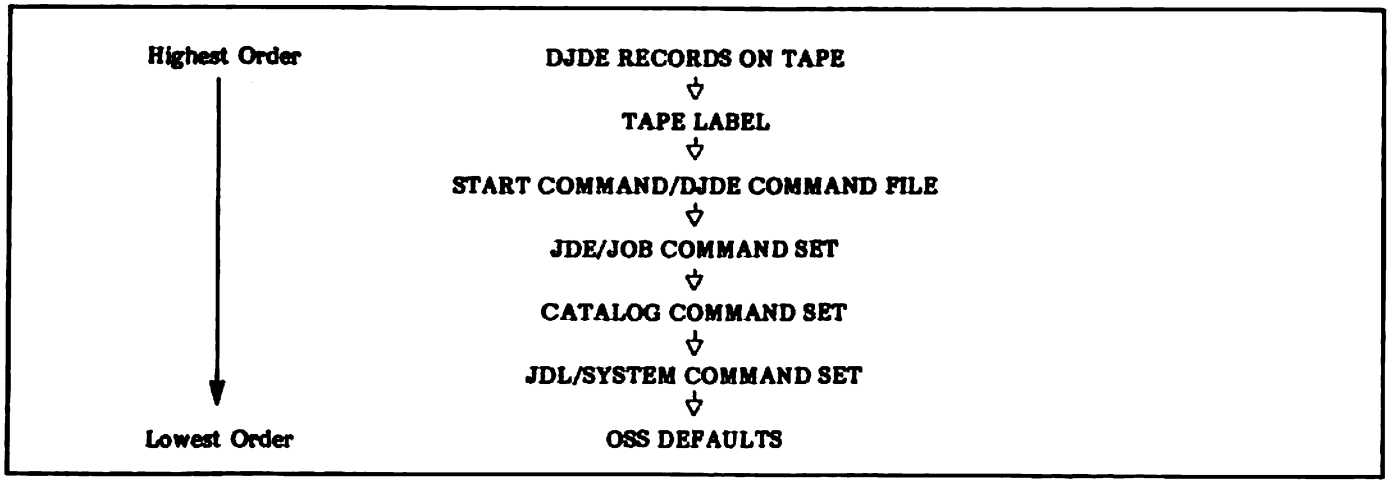


Figure 3-5. Hierarchy of Replacement

In those instances where PDL encounters statements with errors, the hierarchy of replacement is altered somewhat. "Error Processing" is discussed in the subsequent paragraphs.

Error Processing

Introduction

The PDL processor checks each statement in a job descriptor library to ensure that is is syntactically correct. It further checks each job descriptor entry to ensure that it is internally consistent. In performing these error-processing functions, PDL determines if an error is "fatal", or "non-fatal". When a fatal error is encountered, the job descriptor library processing will be aborted.

When non-fatal errors are encountered, PDL will validate the job descriptor entry; thus, the entry will be included as part of the job descriptor library. Since those statements which contain errors are ignored, the highest order, error-free statement is applied. Appropriate diagnostic messages are output to inform the system programmer (and operator) that the job descriptor entry which was processed by PDL did not conform to the entry that was programmed due to errors encountered in one or more statements.

If the job descriptor entry adequately describes the job tapes that are to be processed, the errors need not be correeted. Otherwise, it will be necessary to correct the erroneous statements and recompile the library. The diagnostic messages which are output during library processing are contained in Appendix B.

PDL Statement Errors

The PDL processor evaluates a statement from left to right. When a syntax error is encountered in a statement's "left part", PDL stops evaluating the statement and outputs an appropriate error message. Thus, additional errors which may be present in the statement will not be reported. On the other hand, if a "right part" option is out of range, or otherwise invalid, an appropriate error message is output, but evaluation of the statement continues.

Hierarchy of Replacement in an Errored JDL

Figure 3-6 shows the effect of printer control statement errors on the hierarchy of replacement when the PDL is evaluating a coded job descriptor library. Each example shown in Figure 3-6 is discussed briefly below.

Example 1. The VOLUME CODE statement in the SYSTEM command set contains a syntax error (CODE = ASCIII). The system default value, EBCDIC, will be applied to the JOB command set (job descriptor entry JOB1).

Example 2. The VOLUME CODE statement in the CATALOG command set contains a syntax error (VOLUME CODE = EBDIC). The code default, ASCII, specified in the SYSTEM command set will be applied to the JOB command set (job descriptor entry JOB1).

Example 3. The VOLUME CODE statement in the JOB command set contains a syntax error (VOLUME CODE = PEBDDIC). The volume code, EBCDIC, specified in the CATALOG command set (catalog AA) will be applied to the JOB command set (job descriptor entry JOB1).

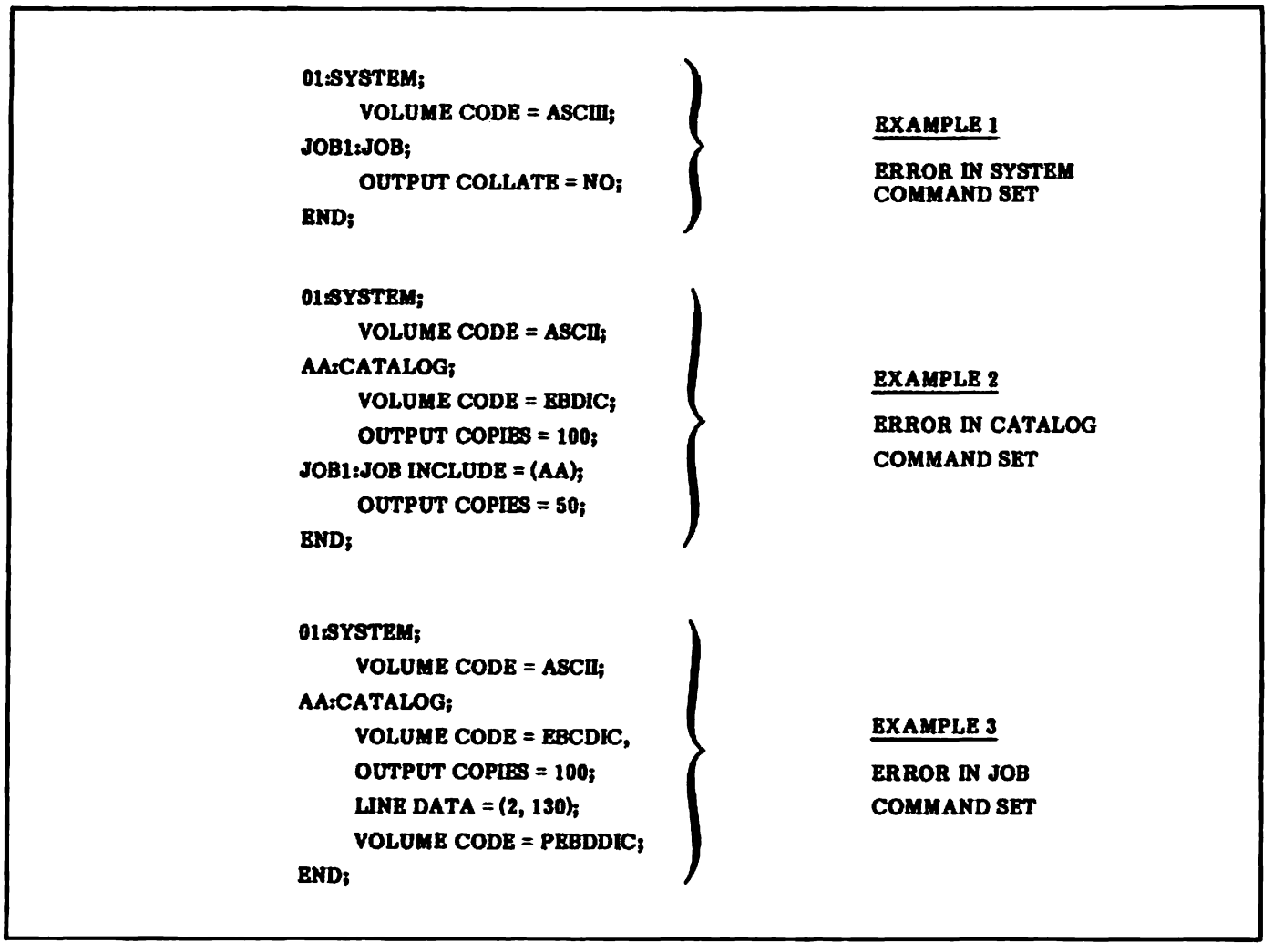


Figure 3-6. Command Coding with Errors, Hierarchy of Replacement

JDL Creation and Compilation

JDL Creation

Once a JDL has been defined and coded it can be written to tape on a host system and then processed by the PDL processor. It may also be entered directly into the 9700 via the EDITOR (as described in Chapter 8).

JDL files on tape must be fixed, unblocked format with no carriage control and a maximum block/record length of 133 characters. The JDL (or JDLs) must be followed by two tape marks, except where the host system cannot produce tape marks (Honeywell 2000); then two END; cards must be at the end of the JDLs. A JDL on tape may be copied (see COPY command in Chapter 7) into a 9700 disk file and then modified (if necessary) with the EDITOR as with EDITOR created files.

JDL Compilation

A Job Descriptor Library can be compiled by the PDL processor from a disk file or from tape as illustrated in Figure 3-7. If the JDL is on tape, PDL will first copy the source JDL file to disk and then compile it. Source JDL files from tape are cataloged on disk in the Job Source Library (file type JSL). The outputs of the compilation are 1) a listing of the source statements with error diagnostics and 2) a Job Descriptor Library control file that can be selected by the operator (on the START command) to print a particular job tape.

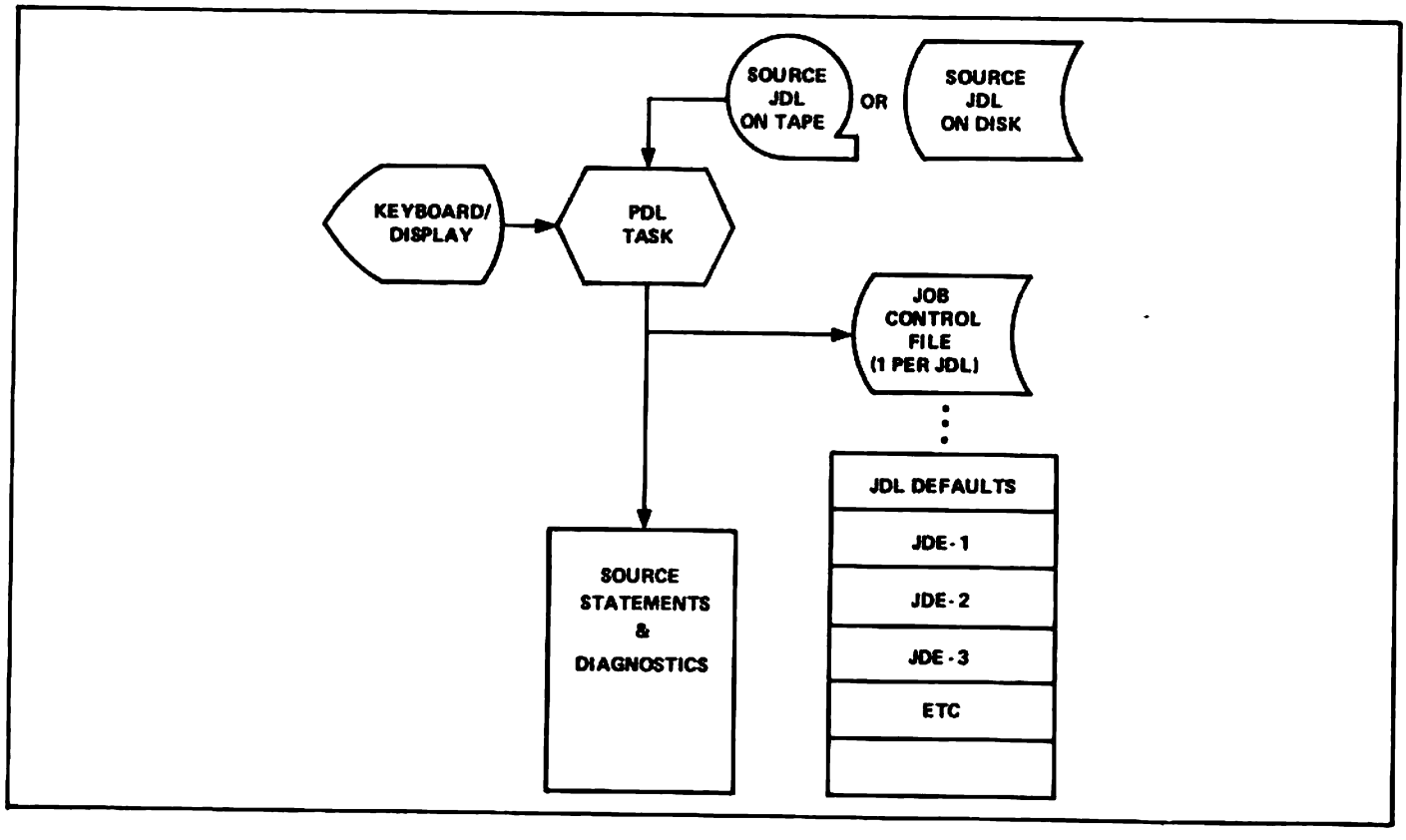


Figure 3-7. PDL Creates JDL Control File

For the PDL processor to compile the JDL form tape or a disk file the following command is entered on the keyboard/display.

PDL [file-name] [,NOPRINT] [,NOSOURCE] [,REPLACE / ,NOREPLACE] [,TRAY] [,NRWD]

where

file-name — specifies the name of a source Job Descriptor Library file which is to be input to the PDL task. If no filename is specified, source input for PDL will be read from tape and a JDL disk file will be created for each SYSTEM command encountered.

NOPRINT — specifies that only source records which contain errors, the diagnostics which apply to those lines, and the PDL summary reports are to be printed during compilation. The default is to print all of the above plus the PDL source records.

NOSOURCE — specifies that no JDL source files are to be created when the input is from tape. This option has no effect when input is not from tape.

REPLACE — specifies that the PDL task may replace an existing job control file with a new output file of the same name. This is the default.

NOREPLACE — specifies that the PDL task is to abort the writing of a file if it has the same name as an existing file. REPLACE is the default.

TRAY — specifies that the listing and diagnostics from the PDL compilation are to go to the sample print tray. Bin is the default.

NRWD — specifies that no rewind is to occur after a tape file is processed. The default is to rewind.

For example, the command:

```
PDL     ,NOSOURCE, NOPRINT
```

will compile JDL file(s) from tape, create JDL control file(s) or replace previously existing job control files (of those already on disk with same SYSTEM name), print diagnostic listings (not the source) to the bin, and rewind the tape after the JDL file(s) are processed.

The command;

```
PDL     H2SYS, TRAY
```

initiated from the keyboard/display will compile the disk file H2SYS (file type JSL), create a JDL control file in the JDL file directory (or replace a previously existing control file) and print the source listing and diagnostics to the sample print tray.

Default JDE

A default Job Descriptor Entry (JDE) can be defined by the user in a Job Descriptor Library (JDL). If the user does not specify a default then one will be created in the JDL control file when PDL compiles the user's JDL. This default JDE is useful when initiating a print job with the START command in that the "jde" parameter (see "Starting a Job" in Chapter 7) need not be specified (as well as other options on the START command for other defaults).

The default JDE created by PDL is named "DFLT". It will contain all the left/right part values that are specified at the SYSTEM level of the JDL or SYSTEM level defaults for those left/right parts not specified. The identifier of a user created default JDE must be "DFLT". This default JDE will be handled in the JDL as any other user defined JDE (as defined previously in "JOB Command Set" of this chapter).

PDE and CME Compilation

The PDL processor also compiles PDE and CME source files. PDE and CME source statements are created by the user, cataloged in the JSL directory and processed by PDL just as with previously discussed JDL files. After PDL processing, a control file is created and cataloged on disk in the CME or PDE file directory. These control files are referenced in a job descriptor library the same as if the CME or PDE statements were coded into it. When the system finds the CMEs or PDEs are not part of the job descriptor library it will go to the control files on disk for the information. CMEs are referenced by the MODIFY left part of the OUTPUT command. PDEs are referenced by the FORMAT left part of the OUTPUT command. Further details on PDEs and CMEs are contained in Chapter 5.

Running a Job Tape

The START command is used to initiate the printing of a job tape. This command is entered by the operator on the keyboard/display console along with other optional parameters. An example of the START command is as follows:

```
START   JDE10,H2SYS
```

where

H2SYS is a job control file created by PDL as a result of compiling the source job descriptor library file H2SYS.

JDE10 is the name of Job Descriptor Entry within the JDL file H2SYS.

The syntactical details of the START command (see "Starting a Job") are defined in Chapter 7 along with other commands which enable the operator to control the flow of print jobs through the system.

4. INPUT PROCESSING FUNCTIONS

Introduction

The input medium to the 9700 is magnetic tape which may be recorded in one of a variety of standard vendor formats. The PDL programmer defines the tape input deblocking and record format parameters which reduce physical tape blocks first to logical records, then to print lines. Further, special processing options can be selected which facilitate report processing or enhance the appearance of the report output.

Prior to selecting the tape options which are to be applied to a job descriptor entry to describe a job tape, certain introductory tape structure concepts must be understood. These concepts, discussed in the first section of this chapter, will enable the PDL programmer to readily define the following job tape characteristics:

- Tape Code (the CODE and LCODE left/right parts).
- Host Computer Formats (the HOST left/right parts).
- Record Structure (includes STRUCTURE, BLOCK, and RECORD options in a sample job tape).
- Packed Data Formats (interpretation and use).

A discussion of the PDL statements used to actually select these options follows the introductory material.

Tape Structure Concepts

Tape Codes

The tape codes recognized by the printing system are EBCDIC, ASCII, several versions of BCD, and USER-defined translation. Tables showing the correspondence between standard recorded codes and printed characters are contained in Appendix C.

Host Computer Formats

The printing system processes the various types of standard host input tapes listed below. The format of each of these types of tapes is described within the 9700 Tape Formats Manual (Publication No. 91 00 04).

- American National Standards Institute (ANSI)
- IBM OS/360 and DOS/360 Standard Labels
- IBM DOS/360 GRASP.
- IBM DOS/360 POWER
- IBM POWER VS
- Xerox Tapes with Xerox Carriage Control
- Univac Series 70 Standard Labels
- Honeywell 200/2000 Labeled Tapes
- Honeywell 600/6000 Standard System Format Labels
- Burroughs Medium System Labeled Tapes
- Burroughs Large System ANSI Labeled Tapes
- Univac SDF Tape Format
- IBM/OS Writer Stacked Report Output

Record Structure

All records input to the printing system are either blocked or unblocked with the fixed length, variable length, or undefined format. These record formats are illustrated in Figures 4-1, 4-2, and 4-3, respectively. In some cases, variations of the record formats can be processed on the printing system, providing the variation is rigorously observed. The GRASP tape format is an example of such a variation. Tape label contents may also describe blocking and record structure. In some cases, tape label contents override BLOCK and RECORD parameters specified in the JDL. These labels are described in the 9700 Tape Formats Manual.

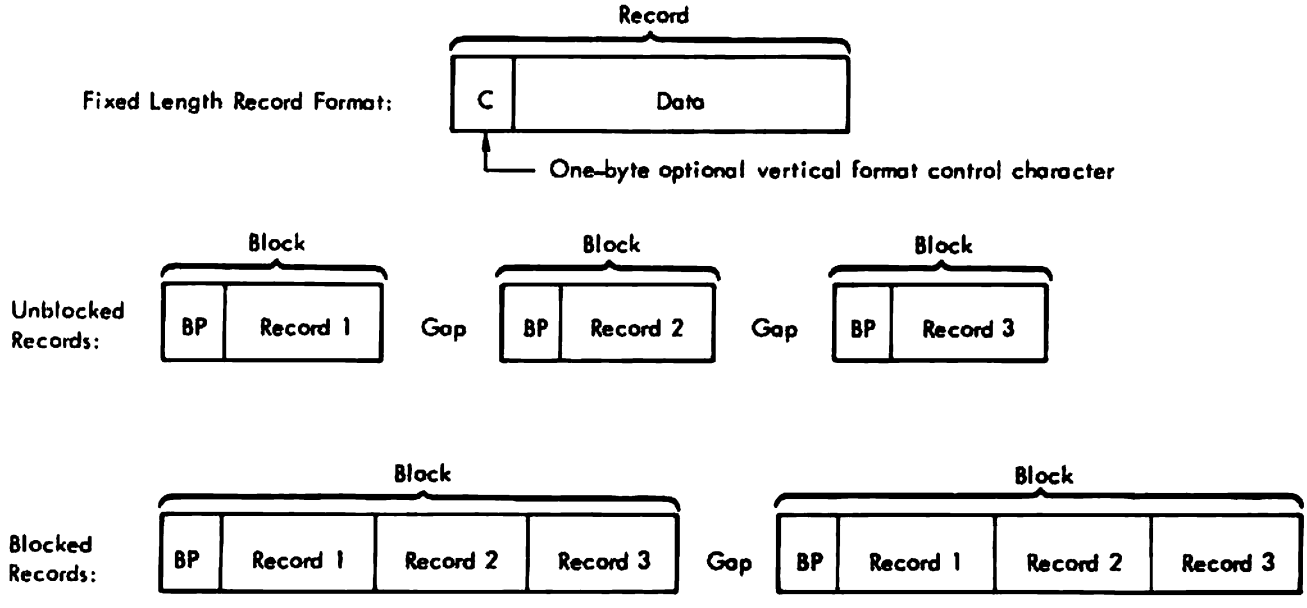
A record is arbitrarily divided into two portions. The operating system's portion of the record contains information supplied by the host operating or spooling system. The user's portion of the record contains information provided by the applications or user's program running on the host system. The boundary between the two portions of the record is traditionally between the record length and printer carriage control field, such that, if there is no record length field, there is no operating system portion of the record.

A record is arbitrarily divided to simplify job descriptor entry definition of relatively constant information such as offsets to carriage control, print data, and/or stacked report fields from a boundary which will not change as a result of a small change to the program job control language (or as a result of an operator keyin to start-up a writer or spooler). In other words, operating system portion of a record changes a great deal in the fixed blocked to variable blocked format transition, but the user portion of the record does not change at all.

A sample input block is broken down into its component parts in Figure 4-4. PDL statements used to define each of the components shown in this record are described in the syntax and interpretation tables for the BLOCK, RECORD, and logical processing statements in this chapter, and the LINE statement in Chapter 5, "Output Processing".

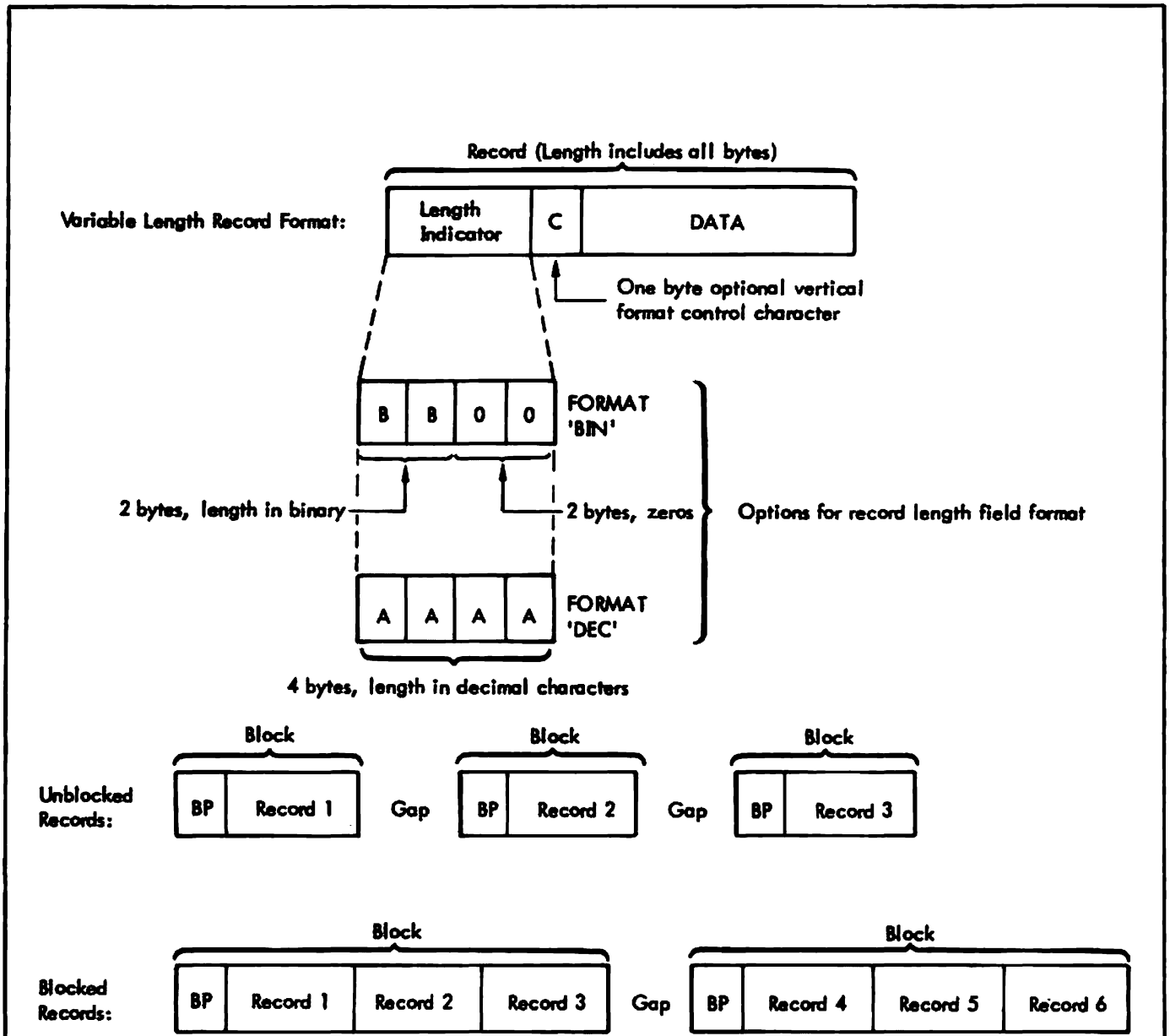
For any tape, labeled or unlabeled, the following criteria must be adhered to:

- Maximum physical block size is limited to 4096 bytes.
- The format specification for the file must remain constant within the file.
- Block length fields must be within the blocks to which they refer.
- Record length fields must be within the records to which they refer.
- Print lines must not span records.
- Any adjustments to the block or record lengths must be a file constant.



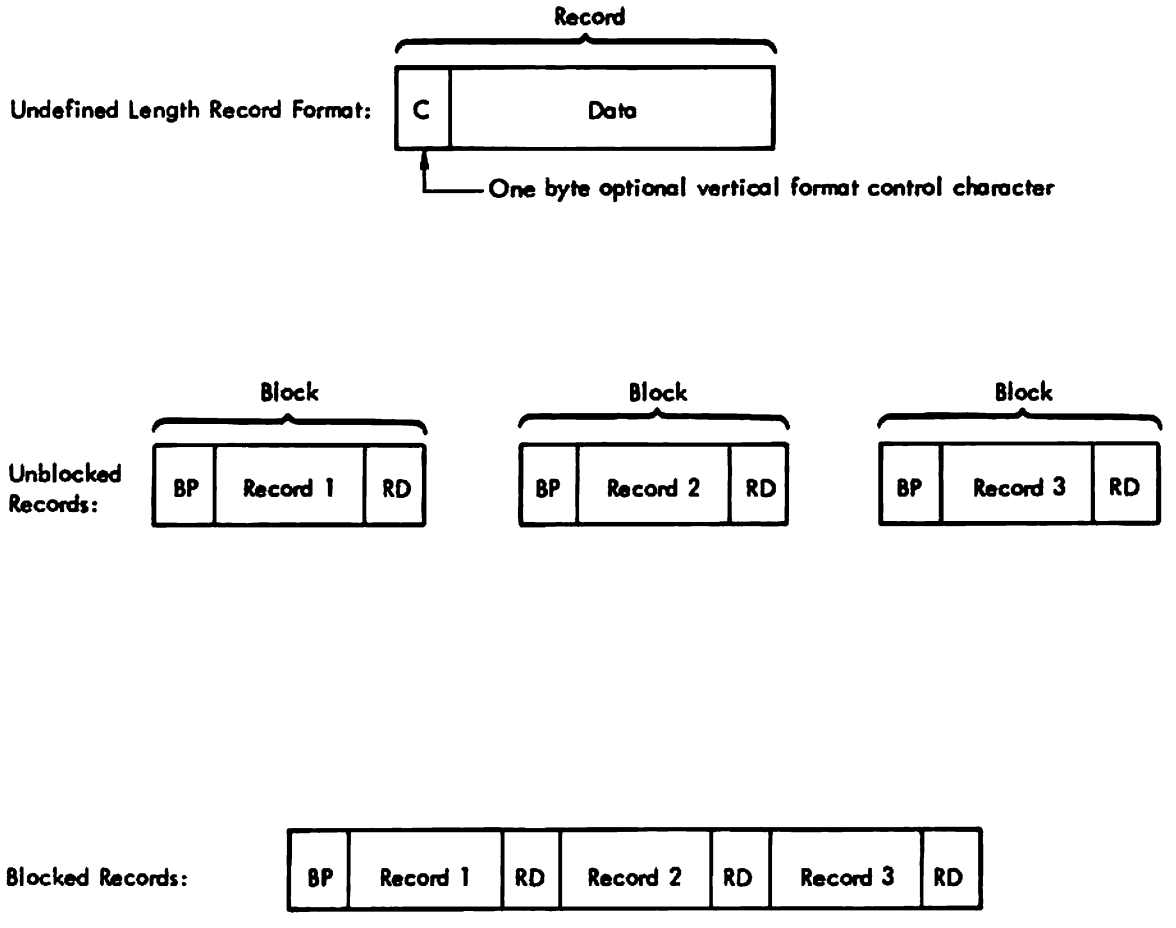
Notes:
1. BP=Optional Block Prefix. (Note: The block postamble is also optional.)
2. All records within a file are of the same length.
3. Blocks may be truncated, but only at the end of a record. Truncated blocks must be an even multiple of the record length plus the block preamble and postamble (if any).
4. Gap=Interblock Tape Gap.

Figure 4-1. Fixed Length Records



Notes:
1. BP = Block Prefix. The block prefix may contain a block length field of the same format as the record length field.
2. Blocks may be truncated, but only at the end of a record.
3. The block may optionally contain a postamble.

Figure 4-2. Variable Length Records

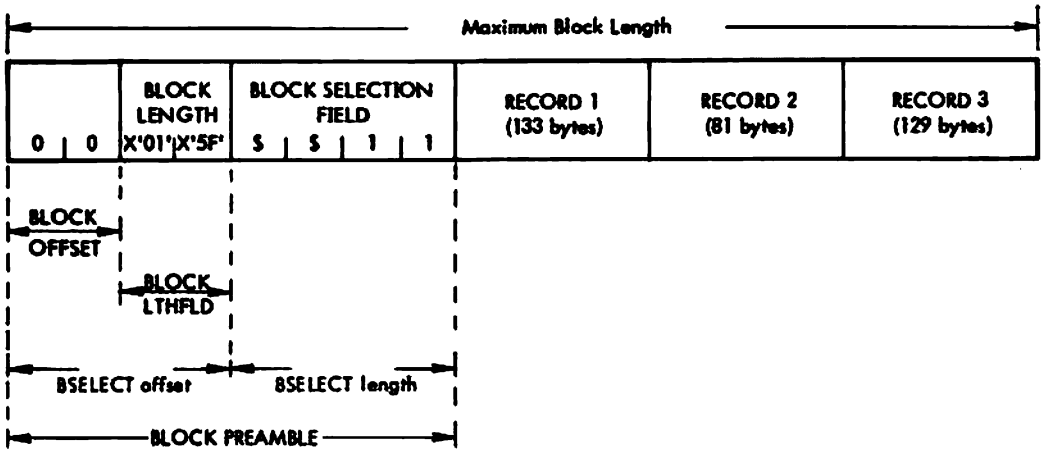


Notes: BP = Optional Block Prefix. (Note: Block postamble is also optional.)

RD = Record Delimiter. The record delimiter is a string of from one to four characters (optional for unblocked format).

Figure 4-3. Undefined Length Records

BLOCK



Interpretation

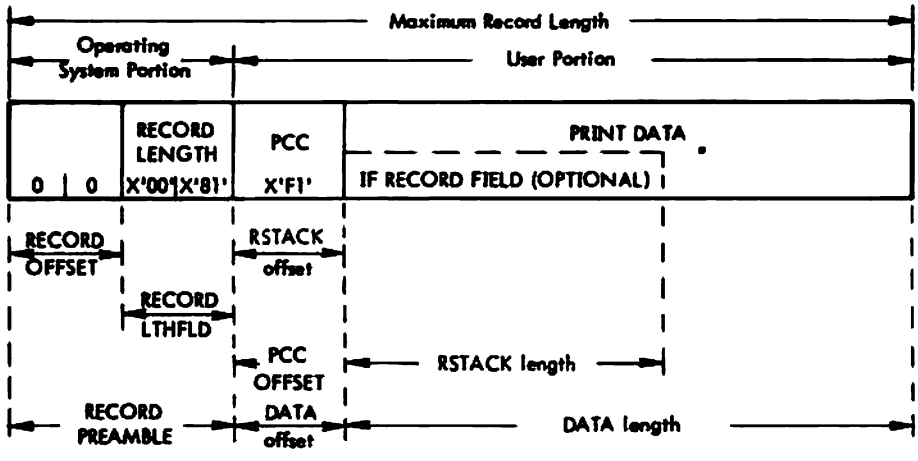
1. The BLOCK statement would be coded as follows:

```
BLOCK LENGTH = 351,
      OFFSET = 2, ADJUST = 0,
      LTHFLD = 2,
      PREAMBLE = 8,
      FORMAT = BIN;
```

2. The logical processing statements for block selection would be coded as follows:

```
T1: TABLE     CONSTANT = ('$$11');
C1: CRITERIA  CONSTANT = (4, 4, EQ, T1);
    BSELECT   TEST = C1;
```

RECORD



Interpretation

1. The RECORD statement would be coded as follows:

```
RECORD LENGTH = 133, OFFSET = 2,
LTHFLD = 2, PREAMBLE = 4, ADJUST = 4,
FORMAT = BIN;
```

2. The logical processing statements for stacked reports would be coded as follows:

```
T1: TABLE     CONSTANT = ('IF RECORD FIELD');
C1: CRITERIA  CONSTANT = (1, 15, EQ, T1);
    RSTACK    TEST = C1, DELIMITER = YES, PRINT = BOTH;
```

3. The LINE statement (discussed in Chapter 5, "Output Processing") would be coded as follows:

```
LINE PCC = (0, NOTRAN), PCCTYPE = ANSI,
DATA = (1, 132),
VFU = V1;
```

Figure 4-4. Components of a Block and a Record

Packed Data Formats

Six-bit characters may be written onto a 9-track tape in a 4X3 packed (or compressed) format. That is, four 6-bit data bytes are compressed into three 8-bit data bytes. Two methods of packing these bits together exist. One method is used by Honeywell 6000 users and is specified by the PDL left/right parts UNPACK = T4X3. Honeywell 2000 users employ a slightly different method of packing which is specified by UNPACK = T4X3H2

Whenever an unpacking method is included in the job descriptor entry, the system unpacks the characters prior to data processing. Each 6-bit character is extracted and 2 high order zeros are appended. This unpacked character may then be translated if the CODE = "right part" option is also coded.

Figure 4-5 shows an example of how 6-bit characters packed in the T4X3 method are unpacked and then translated to ASCII by the system. Figure 4-6 shows an example of unpacking and translating the T4X3H2 method.

Normally after data is unpacked it must be translated. The character code set is defined in the CODE = "right part" option. For a 4X3 unpacking method, the data is generally BCD and either of the 3 standard BCD sets (H2BCD, H6BCD. and IBM-BCD) can be used. These character sets are listed in Appendix C.

Four host BCD characters have the following binary representation:

BCD Character	A	1	5	*
Binary Representation	010001	000001	000101	101100
Represented on 9T Tape by:	1 byte 01000100	1 byte 00010001	1 byte 01101100	
	A	1	5	*
After Unpacking by OSS :	00010001	00000001	00000101	00101100
	A	1	5	*
After Translation to ASCII by OSS :	01000001	00110001	00110101	00101010
	A	1	5	*

Figure 4-5. Pictorial Representation of T4X3 Packing

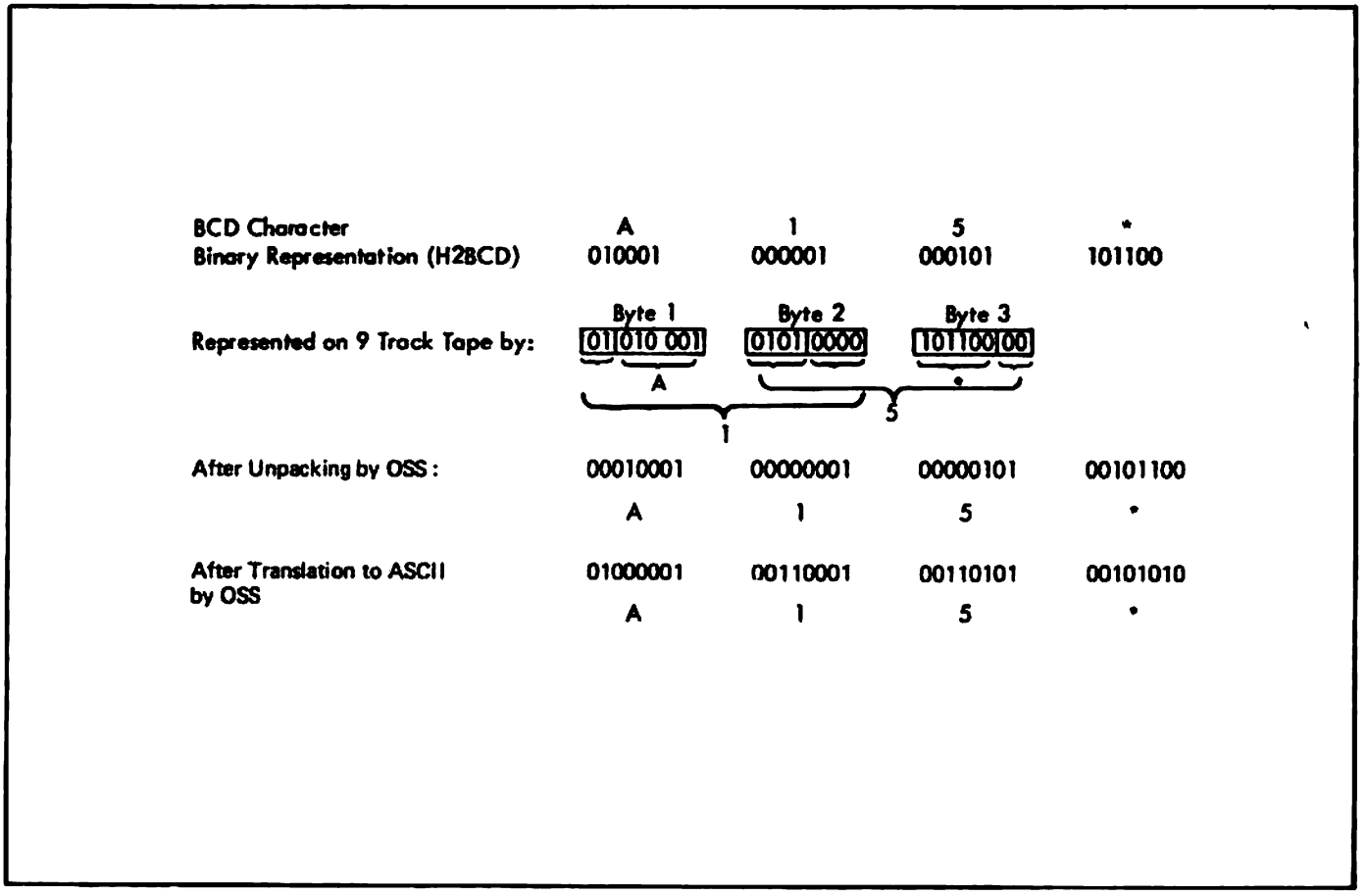


Figure 4-6. Pictorial Representation of T4X3H2 Packing

Data Tape Characteristics

The data tape characteristics are defined by the VOLUME CORE, BLOCK, and RECORD statements. These statements, in effect, enable the system to extract output print lines from the input data stream. They are defined on the following pages.

VOLUME Statement

Command	Left Part	Right Parts	Default	Interpretation
VOLUME	HOST =	{ ANSI IBMOS IBMDOS GRASP POWER POWERVS OSWTR US70 B2500 B2700 B3500 B3700 B4700 B6700 H2000 H6000 DUMP OCTDUMP UNIVAC XEROX } ,	IBMOS	The right part can be any one of the host options.
	UNPACK =	{ T4X3 T4X3H2 UNIVAC NONE } ,	NONE	The right parts "T4X3" and "T4X3H2", specify different unpacking routines, whereby 6-bit characters are extracted and then restored as 8-bit bytes with the new left-most bits set to zero. "T4X3" is used primarily for Honeywell 6000 tapes, "T4X3H2" is used for Honeywell 2000 tapes, and UNIVAC is used for UNIVAC tapes, although any of these routines can be specified independently of the HOST type. These routines are described in a previous section of this chapter. The right part "NONE" indicates that no unpacking operation is to be performed.
	LABEL =	label,	STANDARD	The right part "label" can be any one of the following tape label options: NONE (the input tape is unlabeled) ANSI STANDARD SPR (Honeywell 2000 System Print Tape) COBOL (Honeywell 2000 COBOL Tape with 120-byte labels)

Command	Left Part	Right Parts	Default	Interpretation
VOLUME (cont.)	CODE =	code,	EBCDIC	The left part "CODE =" specifies that the input tape data is to be translated prior to output. The right part "code" can be any one of the following options: ASCII BCD BCL (Burroughs BCL) CBCD (CDC BCD) EBCDIC PEBCDIC (wrap-around, printable EBCDIC) H2BCD H6BCD (same as BCD) IBMBCD USER (single user-defined code translation)[t] id (user-defined code translation)[t]
	LCODE =	code,	EBCDIC	The left part "LCODE = " specifies that the input tape labels are to be translated. The right part "code" can be any one of the following options: ASCII BCD BCL (Burroughs BCL) CBCD (CDC BCD) EBCDIC PEBCDIC (wrap-around, printable EBCDIC) H2BCD H6BCD (same as BCD) IBMBCD USER (single user-defined code translation)[t] id (user-defined code translation)[t]
	EOV =	({PAUSE / NOPAUSE}, {EOF / NOEOF}),	(NOPAUSE, NOEOF)	The left part "EOV =" specifies action to be taken by the control program when the end-of-volume point is encountered on the input tape.

t The standard codes are shown in Appendix C. Creation of a USER translation table referenced by either the keyword USER or an identifier is defined using the CODE statement. See the subsequent section of this chapter "CODE Statement".

Command	Left Part	Right Parts	Default	Interpretation
VOLUME (cont.)	EOV = (cont.)			If "PAUSE" is specified, a message will be displayed when EOV is encountered. A CONTINUE response by the operator will cause the tape to be rewound and the normal volume change sequence to be performed. If NOPAUSE is specified, the rewind will be issued as soon as the EOV label is processed. If "EOF" is specified, the end-of-volume label will be treated as an end-of-file label. When this occurs, the first part of the volume spanning page is output as the last page of the job. (Note: The second part of the spanned page is printed as the first job page when the next volume is started, with possible page format irregularity.) If "NOEOF" is specified, normal end-of-volume processing results.
	PLABEL =	{YES / NO} ,	NO	The left part "PLABEL =" is used to specify whether or not the tape labels on a labeled input tape are to be printed. If "YES" is specified, all tape labels (except those encountered during a volume change) are printed on an output page as they are encountered. The output page is delivered to the sample print tray. If "NO" is specified, no tape label printing results.
	OSCHN =	value,	9	The left part "OSCHN =" is meaningful only if the statement HOST = OSWTR has been previously coded. The right part "value" specifies the vertical format unit (VFU) channel which is used to signal the end of a report generated by the IBM OS Writer. When a skip to the specified channel occurs (as determined by the printer carriage control field within a logical record), AND an overprint operation immediately follows the skip-to-channel operation, the IBM OS Writer banner page is considered found.

Command	Left Part	Right Parts	Default	Interpretation
VOLUME (cont.)	OSTLP =	value,	0	The left part "OSTLP" is meaningful only if the statment HOST = OSWTR has been coded. The right part "value" specifies the number of trailer banner pages which are to follow the end of a report created by the IBM OS Writer.
	OSHDP =	value,	0	The left part "OSHDP =" is meaningful only if the statement HOST = OSWTR has been coded. The right part "value" is the number of header pages which precede the report.
	BMULT = RMULT =	value, value,	1 1	The "BMULT =" and "RMULT =" statements provide for the specification of multiplication factors which are applied to the maximum block (BMULT) and record (RMULT) length values extracted from the header tape label. The right part "value" is an integer in the range $1 \leq \text{value} \leq 15$.
	RMODE =	$\left\{ \begin{matrix} S \\ M \end{matrix} \right\}$;	M	The left part "RMODE =" is used to control processing of multi- or single-report processing. The right part "S" sets up the print job to be run in single-report mode. The right part "M" sets up the print job to be run in multi-report mode. The operator can override the "RMODE" value on the START command. See "Starting a Job" in Chapter 7.

Points to Note

1. The input statement LABEL = "label" should not be coded in cases where it is not appropriate. For example, if the statement HOST = GRASP is coded, no LABEL statement is required, and, if one is coded, it is ignored.
2. If the HOST left part and its associated right parts are inconsistent with the LABEL left part and its associated right parts, an error message is output during job descriptor library processing, and a valid label right part is automatically substituted. The HOST right part which is specified is assumed to be correct; PDL substitutes a valid label. Table 4-1 summarizes the label specifications that are valid for each host type. The shaded areas of the table show the label that is substituted by PDL when an invalid host/label pair is specified.
3. A tape to be dumped is treated as unlabeled (the statement HOST = DUMP or HOST = OCTDUMP is coded).

Table 4-1. Valid Host Computer/Label Specification

HOST TYPE	LABEL SPECIFICATION				
	Unlabeled	ANSI	STANDARD	SYSTEM PRINT	COBOL
IBMOS	X	X	X		
IBMDOS	X	X	X		
GRASP, POWER, POWERVS	LABEL SPECIFICATION IS IGNORED				
OSWTR	X	X	X		
US70	X	X	X		
XEROX	X	X			
B2500 Burroughs Medium Systems	X	X	X		
B2700 Burroughs Medium Systems	X	X	X		
B3500 Burroughs Medium Systems	X	X	X		
B3700 Burroughs Medium Systems	X	X	X		
B4700 Burroughs Medium Systems	X	X	X		
B6700 Burroughs Large System	X	X			
H2000 Honeywell 200/2000 Series			X	X	X
H6000 Honeywell 600/6000 Series			X		
DUMP	LABEL SPECIFICATION IS IGNORED				
OCTDUMP	LABEL SPECIFICATION IS IGNORED				
UNIVAC	X				
ANSI		X			

The shaded areas of this table show the label that is substituted by PDL when an invalid host/label pair is specified.

CODE Statement

This statement must be coded whenever the CODE/LCODE = USER or CODE/LCODE = id left/right part(s) appear in the VOLUME statement. The CODE statement by itself causes no action to occur except to define a user code translation table.

Command	Left Part	Right Parts	Default	Interpretation
[ac]: CODE	DEFAULT =	{ ASCII, BCD, BCL, CBCD, EBCDIC, PEBCDIC, H2BCD, H6BCD, IBMBCD, value },	EBCDIC	"ac" is a 1 to 6 character statement identifier that is referenced by the CODE/LCODE = id portion of the VOLUME command. One character must be alphabetic; the other characters alphabetic or numeric. The "DEFAULT" left part enables the user to specify a base code from which code assignment exceptions can be readily made. The base code is specified from the right part selections of the statement; the exceptions are specified via the "ASSIGN =" statement, described below. The right part "value" is a string of characters which is a one byte hexadecimal, octal, or character constant.
	ASSIGN =	{ (input, output) | (input, (output, ..., output)) };		The right part "input" defines the input code; the right part "output" defines the corresponding output code.

Points to Note

1. Multiple user-defined code translation tables are allowed; only one may be without a statement identifier. The corresponding CODE/LCODE left part on the VOLUME command references each user table via a statement identifier. The right part USER is used to reference the user-defined code translation table in which no statement identifier is coded.

2. The "DEFAULT =" statement must be coded prior to any "ASSIGN =" statements. A "DEFAULT =" statement following any defined corresponding input/output codes causes these correspondence to be replaced by the DEFAULT assignment.

3. An example of statement coding used to modify a base code set is shown below.

   A user's job input tape is recorded in EBCDIC. On output, however, codes 5B, 5C and 5D (characters $ *), respectively) are to be assigned to the character blank (X'40). The coded statements to effect this input/output modification would be as follows:

   CODE DEFAULT = EBCDIC, ASSIGN = (X'5B', X'40'), ASSIGN = (X'5C', X'40'), ASSIGN = (X'5D', X'40');

   or, alternatively,

   CODE DEFAULT = EBCDIC, ASSIGN = (X'5B', (X'40', X'40', X'40'));

   Thus, in the latter case, consecutive input codes need not be specified to accomplish code modification.

BLOCK Statement

The BLOCK statement is used to describe the physical structure of tape blocks.

Command	Left Part	Right Parts	Default	Interpretation
BLOCK	LENGTH =	value,	1330	The right part "value" specifies the length, in bytes, of the longest physical block in the file. "value" is an integer in the range 12 ≤ value ≤ 4096. The tape label contents may override a coded "LENGTH =" command.
	LTHFLD =	size,	0	The right part "size" specifies the length, in bytes, of the field containing the block length right part "value" specified above. "size" is an integer in the range 0 ≤ size ≤ 5. If "size" is set to zero, the block length field is not considered to be part of the block, and the length of a block, on tape, is the actual block length.
	OFFSET =	value,	0	The right part "value" specifies the length, in bytes, of the block length field offset. This offset is the number of bytes from the first byte of a block to the block length field. "value" is an integer in the range 0 ≤ value ≤ (LENGTH-LTHFLD) -1.
	FORMAT =	type,	BIN	The right part "type" specifies the recorded mode of the block length field. "type" must be one of the following parameters: BIN (Binary) DEC (Decimal) PACK (Packed with no sign) PKSG (Packed with sign)
	ADJUST =	value,	0	The right part "value" specifies the block adjustment length. This length is a constant integer which is to be added to or subtracted from the value in the block length field of every tape block. The resulting value is the true block length. The character plus (+) or minus (-) may be used to specify a positive or negative adjustment.

Command	Left Part	Right Parts	Default	Interpretation
BLOCK (cont.)	PREAMBLE =	length,	0	The right part "length" specifies the block preamble length. This length is the byte offset from the first byte of a tape block to the first byte of the first logical record (the operating system portion of the record in the block). "length" is an integer in the range 0 ≤ length ≤ LENGTH.
	POSTAMBLE =	length,	0	The right part "length" specifies the length, in bytes, of any extraneous data at the end of all tape blocks. In effect, "length" is an offset from the end of a block backwards to the end of the last logical record. "length" is an integer in the range 0 ≤ length ≤ LENGTH.
	CONSTANT =	sc,	none	The "CONSTANT" statement is used when tape blocks are to be delimited by block delimiter constants. This indicates that the constant and the data following the constant are ignored until the end of the block is reached. The right part "sc" specifies a string, hexadecimal, or octal character constant as described in Chapter 3. The length of the constant must be from one to four bytes.
	ZERO =	{ YES / NO } ,	NO	The right part "YES" specifies that the end of a tape block is indicated by a value of zero in the record length field (before applying the record length adjustment; see the "ADJUST =" statement described previously). Data which follows the record will be ignored up through the end of the block. The right part "NO" indicates that the end of a tape block is <u>not</u> indicated by a value of zero in the record length field.
	LMULT =	value;	1	The right part "value" specifies a multiplication factor to be applied to the block length. The value specified will be multiplied by the value in the length field (see "LENGTH =" statement previously described) to compute the number of bytes in the block. "value" is an integer in the range 1 ≤ value ≤ 15.

Points to Note

1. The BLOCK LENGTH left/right parts may be overridden by ANSI, IBM OS/Standard, or Honeywell 2000 COBOL labels which specify block length.

2. The values for BLOCK LTHFLD, OFFSET, FORMAT, and PREAMBLE left/right parts may be overridden if RECORD STRUCTURE is changed as the result of ANSI, IBM OS/Standard, or Honeywell 2000 COBOL label processing.

3. The LENGTH on a 4X3 packed format tape is the number of 6-bit bytes or characters in the tape block. For Honeywell 6000 ASCII tapes, the LENGTH is still the number of 6-bit characters in the block.

4. The length of the block delimiter constant should not be coded as the block postamble. Both lengths will be subtracted from the end of the block.

5. The search for the block delimiter constant will start after the block preamble and proceed forward to the first appearance of the constant.

RECORD Statement

The RECORD statement describes the characteristics of the logical tape records.

Command	Left Part	Righ Parts	Default	Interpretation
RECORD	STRUCTURE =	type,	FB	The "STRUCTURE" statement describes the general record structure for a file. The right part "type" may be any of the following structures: F Fixed length. FB Fixed length blocked. V Variable length. VB Variable length blocked. U Undefined length. UB Undefined length blocked. The tape label contents may override the STRUCTURE left/right parts.
	LENGTH =	value,	133	The right part "value" specifies the length, in bytes, of the longest logical record in the file. "value" is an integer in the range 1 ≤ value ≤ BLOCK LENGTH. The tape label contents may override the LENGTH statement.
	LTHFLD =	size,	0	The right part "size" specifies, in bytes, the record length field length. "size" is an integer in the range 0 ≤ size ≤ 5. If size is set equal to zero, then record lengths are not contained in the records, and the record length is the maximum length ("LENGTH =") for each record.
	OFFSET =	value,	0	The right part "value" specifies the record length field offset. This offset is the byte offset from the first byte of the record to the record length field. "value" is an integer in the range 0 ≤ value ≤ (LENGTH-LTHFLD) -1.
	FORMAT =	type,	BIN	The right part "type" specifies the format of the record length field to be one of the following: BIN (Binary) DEC (Decimal) PACK (Packed with no sign) PKSG (Packed with sign)

Command	Left Part	Righ Parts	Default	Interpretation
RECORD (cont.)	ADJUST =	value,	0	The right part "value" specifies the record adjustment length. "value" is a constant integer which is to be added to or subtracted from the value in the length field of every record (see "LENGTH =" statement described previously). The first character of the "value" right part may be plus (+) or minus (-).
	PREAMBLE =	length,	0	The right part "length" specifies the record preamble length. This length is the byte offset from the first byte of the record to the first byte of the user's portion of the record. "length" is an integer in the range 0 ≤ length ≤ LENGTH.
	POSTAMBLE =	length,	0	The right part "length" specifies the length, in bytes, of any extraneous data in a postamble at the end of a user's record. "length" is an integer in the range 0 ≤ length ≤ LENGTH.
	CONSTANT =	sc,	none	The "CONSTANT" statement is used when tape records are to be delimited by record delimiter constants. The record delimiter constant string signals the end of the record, but it is not included in the print line. The right part "sc" specifies a string, hexadecimal, or octal character constant as described in Chapter 3. The length of the constant must be from one to four bytes.
	LMULT =	value;	1	The right part "value" specifies a multiplication factor to be applied to the record length. The value specified will be multiplied by the value in the length field (see "LENGTH =" statement previously described) to compute the number of bytes in the record. "value" is an integer in the range 1 ≤ value ≤ 15.

Points to Note

1. The RECORD LENGTH left/right parts may be overridden by ANSI, IBM OS/Standard, or Honeywell 2000 COBOL labels which specify record length.

2. The values for RECORD LTHFLD, OFFSET, FORMAT, and PREAMBLE left/right parts may be overridden if RECORD STRUCTURE is changed as the result of ANSI, IBM OS/Standard, or Honeywell 2000 COBOL label processing.

3. RECORD CONSTANT may be enabled as the result of RECORD STRUCTURE being changed to "U" in label processing; however, no definition is assumed for the constant string. The default must be zero, or it must be defined in the job descriptor entry.

4. The LENGTH on a 4X3 packed format tape is the number of 6-bit bytes or characters in the record.

Special Processing Statements

Introduction

The special processing statements enable the user to specify logical functions which are performed on either a record or block basis. The following functions can be specified:

1. Define an end-of-report condition.
2. Select or delete the processing of blocks.
3. Select or delete the processing of records.
4. Define a page which will be offset in the bin.
5. Control the suppression of printing for sets of records within a report.

In order to fully define a special processing statement, the user must specify one or two fields in the record or block to be tested. In general, a logical processing statement has the following format:

Function TEST = (criteria);

The logical processing statement tests the value of the specified "expression" and directs the flow of processing based on the result of the test.

The "expression" portion of the statement defines a test to be performed on either one or two specified fields and their associated constants for a true or false value. The fields in the record or block are compared with their associated set of constants using either an equal (EQ) or not equal (NE) operator.

The basic element used to describe a test for a logical function is the CRITERIA statement. Each CRITERIA statement describes a field in either a record or block and the specific test to be performed. For example, the following CRITERIA statement describes a test for a subfield equal to a specific constant table.

CRITERIA CONSTANT = (offset, length, EQ, id_1);

The left part CONSTANT specifies that the content of a tape record or block, located "offset" bytes from the start of the record or block, with length "length" in bytes, is to be compared to the table constant value where id_1 is the identifier of a table containing constants. (See the final section of this chapter, "Table Statement".) When the subfield matches the constant set, the CRITERIA statement is true.

To complete the description of the entire test for a logical function, the TEST left part requires a right part that specifies either one or two CRITERIA statements. If only one test is to be performed to determine the value of a particular logical processing function, the form of the TEST left/right parts is as follows:

TEST = (id_1);

where "id_1" is the identifier for the particular CRITERIA statement. The parentheses in this format are optional.

If two CRITERIA statements are needed to determine the true/false value for a logical processing function, they may be either ANDed together or ORed together. The formats of the TEST left/right parts may be as follows:

TEST = (id_1, AND, id_2); or TEST = (id_1, OR, id_2);

where "id_1" and "id_2" are the identifiers for two CRITERIA statements. The parentheses in this format are required.

The particular logical processing function to be performed when the test expression is true is specified by the command. For example, to specify that a record should be selected if a particular field is set, the user should code the following three statements:

```
T1:   TABLE      CONSTANT = (sc);
C1:   CRITERIA   CONSTANT = (offset, length, EQ, T1);
      RSELECT    TEST = (C1);
```

CRITERIA Statement

There are two formats for the CRITERIA statements: constant mode and change mode. In the constant mode, the user must specify the location, length, and contents of a fixed field within a user's record or block. Every user's record or block is examined at the specified location (expression function "offset") to determine if the constant is present or not present (the identifier "id" defines the table containing the constant). If so, the CRITERIA statement is true; if not, the statement is false. A constant mode CRITERIA statement can be coded as follows:

id_1: CRITERIA CONSTANT = (offset, length, $\left\{\begin{matrix}EQ\\NE\end{matrix}\right\}$, id_2);

where id_2 is the name (or identifier) of a value TABLE.

In the change mode, the user must specify the length and location of a control field in each record or block. When the content of the control field of one record (or block) differs from the content of the control field of the previous record (or block), the CRITERIA statement is true.

Change-mode CRITERIA statement can be coded as follows:

id_1: CRITERIA CHANGE = (offset, length, NE, LAST);

Each CRITERIA statement may be either constant mode or change mode but not both.

Identifier/Command	Left/Right Parts	Interpretation
CRITERIA	CONSTANT = (offset, length, $\left\{\begin{matrix}EQ\\NE\end{matrix}\right\}$, id);	In block processing, the expression "offset" is the offset, in bytes, from the start of the physical tape block to a field within the tape block which is to be compared to a table or string constant, or, in the case of record processing, from the start of the user's portion of the record to the field in the record which is to be compared. The expression "length" is the length, in bytes, of the test field.

Identifier/Command	Left/Right Parts	Interpretation
CRITERIA (cont.)		EQ and NE are keywords which specify the operation of "equal to" or "not equal to". The expression "id" is the identifier of a TABLE statement.
	CHANGE = (offset, length, NE, LAST);	The expression "offset" is the offset, in bytes, from the start of the user's portion of the record to the control field within the record, or from the start of the physical tape block to the control field within the block. The expression "length" is the length, in bytes, of the control field. The keyword NE indicates "not equal to". The keyword "LAST" indicates that the control field of the current record (or block) is being compared to the control field of the previous (last encountered) record (or block).

Test Expressions

The activation of testing is done by coding the keyword TEST as the left part of any logical processing command described below. The corresponding right part specifies the criteria for the test. Either one or two CRITERIA statements can be specified. If one CRITERIA statement is specified, the form of the left/right part is as follows:

$$TEST = id_1;$$

If the CRITERIA id_1 is true, the test is true; if the CRITERIA id_1 is false, the test is false. The CRITERIA id_1 may be in either change-mode or constant-mode format.

If two CRITERIA statements are specified for a test, the testing of the two is linked by either the AND logical operator or the OR logical operator. If AND is coded, then both CRITERIA statements must be true in order for the TEST to be successful. If OR is coded, then if either CRITERIA statement is satisfied, the TEST is successful. The format of the left/right part is as follows:

$$TEST = (id_1, \left\{ \begin{matrix} AND \\ OR \end{matrix} \right\}, id_2)$$

The second and third parameters are optional, but if either is specified, the other must also be present. The CRITERIA tables may specify either change-mode or constant-mode functions; there are no restrictions on their usage or combination.

There is one special case when the record or block is too short to include the field being tested. If the test specifies a constant-mode function, the CRITERIA will fail. If the test specifies a change-mode function, the CRITERIA will fail as no change has occurred, but the value for LAST will be unchanged for comparison with the next record.

Logical Processing Commands

There are eight separate logical processing commands. The user can specify all eight but each individual command can be specified only once per job descriptor entry. If any one command is specified more than once, the last occurrence will be used without notification of any error. The available commands are listed below and will be described separately in the following sections:

BDELETE	Block Deletion
BSELECT	Block Selection
RDELETE	Record Deletion
RSELECT	Record Selection
RSTACK	Stacked Reports - Report Separation on a Record
ROFFSET	Report Offsetting on a Record
RSUSPEND	Suspend Printing on a Record
RRESUME	Resume Printing on a Record

Block Deletion and Selection

Interspersed blocks within one report or file may be either selected for, or deleted from, printing by use of the BDELETE and BSELECT statements. These statements can also be used to selectively delete from printing specialized blocks that are not originally placed on the data tape for printing, i.e., control blocks, unsupported labels, etc.

Command	Left Part	Right Parts	Default	Interpretation
BSELECT	TEST =	$\left\{ \begin{matrix} id_1 \\ (id_1, \left\{ \begin{matrix} AND \\ OR \end{matrix} \right\}, id_2) \end{matrix} \right\};$	none	Defines test expression for selecting blocks for printing. id_1 and id_2 are identifiers for either the change-mode or constant-mode CRITERIA statements. If only id_1 is present, then the test is satisfied if the block satisfies the criteria in id_1. If id_1 and id_2 are both present and the keyword AND is coded, the test is satisfied only if the block satisfies the criteria in both id_1 and id_2. If the keyword OR is coded, the test is satisfied if the block satisfies the criteria in either id_1 or id_2. If the test is satisfied, the block is selected for printing.
BDELETE	TEST =	$\left\{ \begin{matrix} id_1 \\ (id_1, \left\{ \begin{matrix} AND \\ OR \end{matrix} \right\}, id_2) \end{matrix} \right\};$	none	id_1, id_2, AND, and OR as above. If the test is satisfied, the block is deleted from the printed output.

Points to Note

1. Offsets to the subfields of the blocks are the offsets, in bytes, relative to zero from the start of the block to the beginning of the subfield.

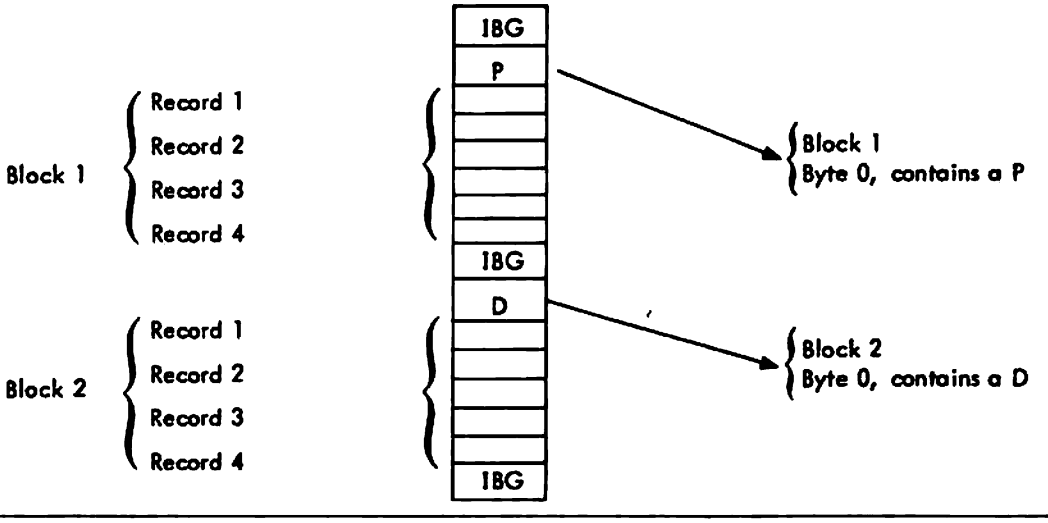
2. Block selection and deletion is performed prior to the extraction of the records from the block. If a block does not match the same format as the normal blocks, it can be deleted and will not cause a processing error. For example, a control block in a fixed blocked file may cause a processing error unless it is deleted first.

3. If a block is deleted from, or not selected for, printing, none of the records contained within the block will be processed at all, and will not be available for any other logical processing functions.

4. Figure 4-7 contains an example of the use of the BSELECT statement to process interspersed reports on a block basis.

Coded Statements

```
T1: TABLE     CONSTANT = ('P');
C1: CRITERIA  CONSTANT = (0, 1, EQ, T1);
    BSELECT   TEST = (C1);
```



Key

IBG Interblock Gap

The contents of the first byte of each tape block (offset = 0, length = 1) is examined for the character constant 'P'. When a 'P' is detected, the entire block is selected (BSELECT) for printing. When the first byte of the block does not contain a 'P', the block is bypassed and not printed. In the above example, only block 1 is printed.

Figure 4-7. Processing Interspersed Reports on a Block Basis, Example

Record Deletion and Selection

Interspersed records within one report or file may be either deleted from, or selected for, printing by use of the RDELETE and RSELECT statements. These statements can also be used to selectively delete from printing specialized records that are not originally placed on the data tape for printing, i.e., control records, offset records, etc.

Command	Left Part	Right Parts	Default	Interpretation
RSELECT	TEST =	$\begin{Bmatrix} id_1 \\ (id_1, \begin{Bmatrix} AND \\ OR \end{Bmatrix}, id_2) \end{Bmatrix}$;	none	Defines test expression for selection of records for printing. id_1 and id_2 are identifiers for either the change-mode or constant-mode CRITERIA statements. If only id_1 is present, then the test is satisfied if the record satisfies the criteria in id_1. If id_1 and id_2 are both present and the keyword AND is coded, the test is satisfied only if the record satisfies the criteria in both id_1 and id_2. If the keyword OR is coded, the test is satisfied if the record satisfies the criteria in either id_1 or id_2. If the test is satisfied, the record is selected for printing.
RDELETE	TEST =	$\begin{Bmatrix} id_1 \\ (id_1, \begin{Bmatrix} AND \\ OR \end{Bmatrix}, id_2) \end{Bmatrix}$;	none	id_1, id_2, AND, and OR as above. If the test is satisfied, the record is deleted from the printed output.

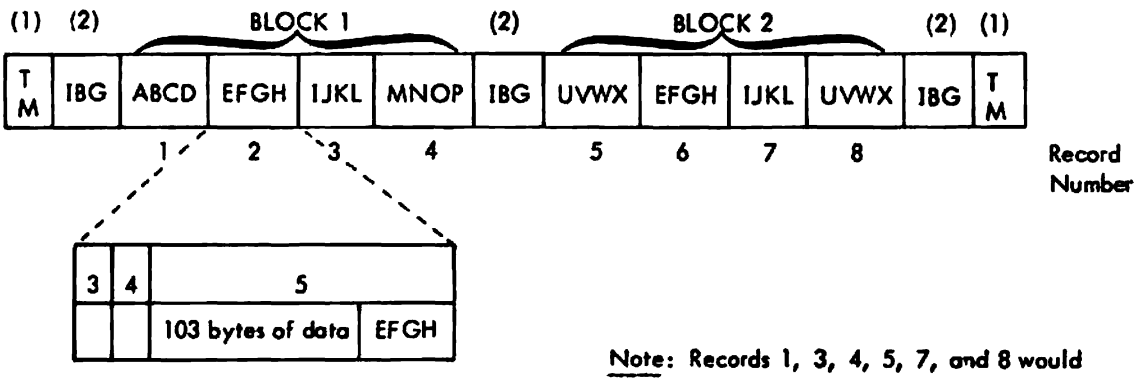
Points to Note

1. Offsets to the subfields of the records are the offsets, in bytes, relative to zero from the start of the user's portion of the record to the beginning of the subfield.

2. Figure 4-8 contains an example of the use of the RDELETE statement to process interspersed reports on a record basis.

Coded Statements

```
T1:  TABLE     CONSTANT = ('EFGH');
C1:  CRITERIA  CONSTANT = (104,4,EQ,T1);
     RDELETE   TEST = C1;
```



Note: Records 1, 3, 4, 5, 7, and 8 would be printed in the report.

Key

(1) TM = Tape Mark
(2) IBG = Interblock Gap
(3) Record Length Field
(4) Carriage Control Byte
(5) Print Line

If the contents of a deletion field located 104 bytes from the start of the user portion of the record are equal to the constant 'EFGH', then the record is not printed.

Figure 4-8. Processing Interspersed Reports on a Record Basis, Example

Print Suppression

The print suppression logical processing function permits the user to delete from printing groups of records that are distinguishable at the start and end, but whose intermediate records may not be unique or distinguishable. Print suppression is invoked by the use of two separate commands - RSUSPEND and RRESUME. The tests for each command are independent and must be separately described. Each of the commands can specify the full range of tests as described previously for the other logical processing commands.

When specifying either the RSUSPEND or RRESUME commands, the user can also specify whether the printing will begin the suspension or resumption of printing on the current or next record. This is controlled by the left/right parts BEGIN = $\left\{\begin{matrix}\text{CURRENT}\\\text{NEXT}\end{matrix}\right\}$. This additional control provides the necessary flexibility to cope with the variability of requirements for print suppression. The BEGIN left/right parts are independently specifiable in both RSUSPEND and RRESUME commands.

Upon encountering a record which satisfies the test criteria specified on the RSUSPEND command, printing will be suspended. If BEGIN = CURRENT is coded on the RSUSPEND command, then this record will not be printed. If BEGIN = NEXT is coded, then the record satisfying the TEST is printed and records will be discarded beginning with the following record.

Printing will be resumed when a record satisfying the TEST in the RRESUME command is encountered. If BEGIN = CURRENT is coded in the RRESUME command, the record satisfying the TEST will be printed. If BEGIN = NEXT is coded, printing will resume with the next record.

Command	Left Part	Right Part	Default	Interpretation
RSUSPEND	TEST =	$\left\{\begin{matrix}id_1\\(id_1, \left\{\begin{matrix}\text{AND}\\\text{OR}\end{matrix}\right\}, id_2)\end{matrix}\right\}$,	none	Defines test expression for the beginning of print suppression. id_1 and id_2 are identifiers for either the change-mode or constant-mode CRITERIA statements. If only id_1 is present, then the test is satisfied if the record satisfies the criteria in id_1. If id_1 and id_2 are both present and the keyword AND is coded, the test is satisfied only if the record satisfies the criteria in both id_1 and id_2. If the keyword OR is coded, the test is satisfied if the record satisfies the criteria in either id_1 or id_2. If the test is satisfied, the record will be used to suspend printing.
	BEGIN =	$\left\{\begin{matrix}\text{CURRENT}\\\text{NEXT}\end{matrix}\right\}$;	NEXT	If BEGIN = CURRENT is coded, the record satisfying the test expression will not be printed. If BEGIN = NEXT is coded, the record satisfying the test expression will be printed and printing will be suppressed beginning with the next record.

Command	Left Part	Right Part	Default	Interpretation
RRESUME	TEST =	$\left\{ \begin{array}{l} id_1 \\ (id_1, \left\{ \begin{array}{l} AND \\ OR \end{array} \right\}, id_2) \end{array} \right\}$,	none	Defines test expression for the resumption of printing following print suppression. id_1 and id_2 are identifiers for either the change-mode or constant-mode CRITERIA statements. If only id_1 is present, then the test is satisfied if the record satisfies the criteria in id_1. If id_1 and id_2 are both present and the keyword AND is coded, the test is satisfied only if the record satisfies the criteria in both id_1 and id_2. If the keyword OR is coded, the test is satisfied if the record satisfies the criteria in either id_1 or id_2. If the test is satisfied, the record will be used to resume printing following print suppression.
	BEGIN =	$\left\{ \begin{array}{l} CURRENT \\ NEXT \end{array} \right\}$;	NEXT	If BEGIN = CURRENT, the record satisfying the test expression will be printed. If BEGIN = NEXT is coded, the record satisfying the test will not be printed and printing will begin with the next record.

Points to Note

1. The user should ensure that if an RSUSPEND command is coded, then an RRESUME command is also present for the job. A warning will be issued if one but not both commands are invoked for a job. However, the JDE will be compiled as programmed.

2. If a data record satisfying the test expression in the RSUSPEND is encountered, printing will be suspended. If no record satisfying the test expression in the RRESUME command is encountered (or no RRESUME command is present for the job), there will be no output generated for records that occur after the point of suspension.

3. A record satisfying the RSTACK test will still be found and will terminate the report, even if the printing of records is suspended at the time.

4. Record selection/deletion is performed prior to suspend/resume processing. If a record satisfying either the suspend or resume test criteria was previously not selected for, or deleted from, printing, it will not cause either the suspend or resume function.

5. The records just prior to print suspension and after resumption should have compatible carriage control characters. No additional carriage control characters will be inserted by the system during the print suppression.

6. Figure 4-10 contains an example of the use of the RSUSPEND and RRESUME statements to delete the printing of Job Control statements from a job.

Sample Input

Records to be deleted from printing

```
//JOB . FOR001A0 C 219                                    DATE 09/23/78
//OPTION CAYAL                                                00000020
  PHASE FOR001D0,                                           09/23/78
//ASSGN SYSRLB,3340,TEMP,VOL=PVTLIB,SHR                     06/16/78
  INCLUDE IMPROOT2                                          06/16/78
  INCLUDE IMPCBBM                                           06/16/78
  INCLUDE UTL063A0                                          09/02/78
//EXEC    FCOBOL,SIZE=64K                                   09/01/78
```

Printed output

```
CBL SUPMAP
CBL SXREF
CBL LIB
CBL OPT
CBL STATE                                                   06/24/78
CBL NOOPT                                                   06/24/78
CBL FLOW                                                    06/24/78
00001  000120 ID DIVISION.
00002  000130 PROGRAM-ID. FOR001B0.
00003  000140 AUTHOR. D. DYLAN
00004  000150 DATE-WRITTEN. MAY 13TH 1978.
00005  000160 REMARKS:
00006  000170     THIS PROGRAM WILL UPDATE ITEM AND FORECAST
00007  000180     RECORDS OFF OF THE FORECAST MASTER AND OUT-
00008  000190     PUT A NEW TAPE. THE USER MAY REQUEST MAIN-
00009  000200     TENANCE ON ALL FIELDS, OR MAY JUST WANT TO
00010  000210     CHANGE ONE FIELD.
```

Coded Statements

```
T1:TABLE          CONSTANT = ('//JOB ','//EXEC ');
T2:TABLE          CONSTANT = ('EOJ ');
T3:TABLE          CONSTANT = ('//EXEC ');

C1:CRITERIA       CONSTANT = (1,7,EQ,T1);
C2:CRITERIA       CONSTANT = (1,4,EQ,T2);
C3:CRITERIA       CONSTANT = (1,7,EQ,T3);

RSUSPEND          TEST = (C1,OR,C2),
                  BEGIN = CURRENT;
RRESUME           TEST = (C3,OR,C2),
                  BEGIN = NEXT;
```

Figure 4-10. Print Suppression Used to Delete Job Control Card Statements from the Printed Output

Stacked Reports

The stacked reports feature enables the printing system user to define a series of reports in a single file. This is accomplished by specifying an "end-of-report" condition in the coded logical processing statement RSTACK. "End-of-report" is that point in processing a report when all of the pages of a copy of a report have been formatted to disk and processing has not begun on the next report.

Reports are "stacked" in a file if more than one report is included in a single file and separated from each other logically and not physically with tapemarks and/or operating system labels. In processing stacked reports, the system checks each record for the logical end-of-report specification in the TEST expression of the RSTACK statement.

The RSTACK statement uses a TEST = left/right part as described in the previous section "Test Expressions". One or two subfields of each record can be tested using either the constant-mode or change-mode criteria tables.

There are two modes of stacked reports which are supported by the system. In the delimiter mode, the record satisfying the TEST criteria is not part of the report following the delimiter but simply serves to separate or "delimit" one report from another. In the non-delimiter mode, reports are stacked one on top of each other without any special records separating the reports. The two modes are specified by the user by coding the left/right part DELIMITER = YES/NO.

If DELIMITER = YES is coded, the user may actually separate each report with multiple successive records, each of which satisfies the TEST expression on the RSTACK statement. In this case, all consecutive delimiters are treated as a delimiter packet. In delimiter mode, the user has the option to print the delimiter or the delimiter packet and to select the output destination of this delimiter page - BIN, TRAY, or BOTH. The delimiter page, when printed, is output as part of the subsequent report. All of the printing options selected in the Print Descriptor Language remain in effect during report delimiter printing except carriage control, which is ignored; the carriage control is replaced by a 'Print and Space 1 Line' control.

When end-of-report in encountered, the first delimiter of the next report will be displayed on the keyboard/display screen if report mode is single. If report mode is multi the delimiter will not be dispalyed. Delimiter mode does not effect whether or not the delimiter will be displayed to the operator.

To resume input, the operator must key-in a START command to begin the next single-report job. The delimiter page itself will be printed when OUTPUT processing begins printing the report.

A special case exists when printing the first report of a tape and RSTACK is in effect. The first data record will be displayed on the screen while input is stopped. If after inspecting this record the operator is satisfied with the job parameters he must enter CONTINUE I to resume input. Otherwise he must abort (see "Aborting a Job" in Chapter 7) the job-id and then to start another job must enter a new START command with appropriate parameters. In either case this record is treated as a delimiter.

When the START command is entered to begin printing the next report, a non-delimiter report (DELIMITER = NO) will be formatted in its entirety including multiple copies (caused by copy sensitive processing) before the next end-of-report condition is encountered.

Delimiter on Accounting Log

The user can select the option of including part of the first record of a report on the Accounting Log which can be printed for that report. This option is selected by coding ACCTINFO = (offset, length) on the RSTACK statement. This option will normally be used to print part of the first delimiter.

Command	Left Part	Right Parts	Default	Interpretation
RSTACK	TEST =	$\left\{ \begin{matrix} id_1 \\ (id_1, \left\{ \begin{matrix} AND \\ OR \end{matrix} \right\}, id_2) \end{matrix} \right\}$,	none	id_1 and id_2 are identifiers for either the change-mode or constant-mode CRITERIA statements. If only id_1 is present, then the test is satisfied if the record satisfies the criteria in id_1. If id_1 and id_2 are both present and the keyword AND is coded, the test is satisfied only if the record satisfies the criteria in both id_1 and id_2. If the keyword OR is coded, the test is satisfied if the record satisfies the criteria in either id_1 or id_2. If the test is satisfied, the record will specify an "end-of-report" condition.
	DELIMITER =	$\left\{ \begin{matrix} YES \\ NO \end{matrix} \right\}$,	NO	If DELIMTIER = YES is coded, all consecutive records satisfying the TEST criteria separate one report from another but are not part of either report. If DELIMITER = NO is coded, this single record separates one report from another and is actually part of the subsequent report.
	PRINT =	$\left\{ \begin{matrix} TRAY \\ BIN \\ BOTH \\ NONE \end{matrix} \right\}$	NONE	If DELIMITER = YES is coded, the user may specify if the report delimiters are to be printed, and if so, the location to where the printed delimiters are to be output: NONE indicates that report delimiters are not to be printed. BIN indicates that report delimiters are to be printed, and the output delivered to the output stacker bin. TRAY indicates that report delimiters are to be printed, and the output delivered to the sample print tray.

Command	Left Part	Right Parts	Default	Interpretation
RSTACK (cont.)				BOTH indicates that report delimiters are to be printed, and the output delivered to both the sample print tray and output stacker bin. If DELIMITER = NO is coded, no page will be printed.
	ACCTINFO =	(offset, length);	(0,0)	Specifies that the subfield, beginning at offset "offset" from the beginning of the user portion of the record, and for a length of "length" in the FIRST (or only) delimiter record (or the first data record if DELIMITER = NO) is to be saved for printing on accounting log at the end of the report. "length" is limited to a maximum of 128 bytes. The default is (0,0), which indicates that no accounting information is to be extracted and printed.

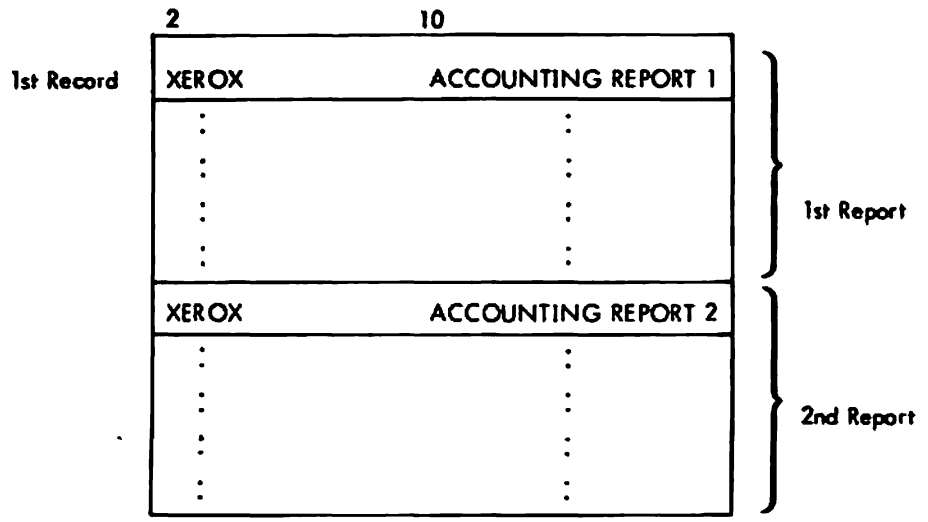
Points to Note

1. If the TEST expression on the RSTACK statement consists solely of a change-mode CRITERIA statement, the DELIMITER = NO must be coded.

2. An RSTACK statement containing a TEST expression specifying a constant-mode CRITERIA statement and DELIMITER = NO can be used to detect a heading of a report as a report boundary.

3. A record which is an RSTACK delimiter cannot be deleted from, or not selected for, printing by the RSELECT/RDELETE logical processing. If the record satisfies the RSTACK test criteria but is not a delimiter, it can be deleted from, or not selected for, printing but will still cause report separation.

4. Figure 4-11 includes an example of a stacked report.



Coded Statements:

```
T1: TABLE     CONSTANT = ('XEROX');

C1: CRITERIA  CONSTANT = (1,5,EQ,T1);

    RSTACK    TEST = C1,
              DELIMITER = YES,
              ACCTINFO = (10,19);
```

Figure 4-11. Example of Stacked Report Processing

TABLE Statement

The TABLE statement is used to build a table of constants, for use by the special logical processing statements.

Identifier/ Command	Left Part	Right Parts	Default	Interpretation
ac:TABLE	CONSTANT =	(sc,...sc);		The table identifier (referenced in the CRITERIA statement as "id") is "ac". The right part "(sc,...sc)" is one or more hexadecimal, octal, or character string constants, as described in Chapter 3.

Points to Note

1. Each constant included in a TABLE statement will be examined by the system to see if it is equal in value to the constant specified in the "expression constant" portion of the CRITERIA statement.

2. Each constant in a TABLE statement must be the same length in bytes.

3. The number of bytes for all constants in the table (after conversion if any constants are hexadecimal or octal) is limited to 255 bytes.

4. The TABLE statement must precede its reference in a CRITERIA statement.

5. OUTPUT PROCESSING FUNCTIONS

Introduction

The PDL programmer defines the output report format for a job or series of jobs in the job descriptor library. The OUTPUT, CME, LINE, PDE, and VFU statements enable the user to specify the following report characteristics:

1. Number of copies and collate mode.
2. Copy modification (or spot carbon) of output.
3. Forms and Fonts to be used.
4. Control of report offsetting.
5. Print line length and carriage control conventions.
6. Vertical format channel (output channel to output line correspondence).
7. Top-and bottom-of-form values.

The job accounting command ACCT, enables the installation to specify that an accounting summary is to be output at the end of processing for each report. The ABNORMAL command allows the user to select optional features that enhance output integerity. The remaining sections of this chapter discuss the output statements and the syntax to be used in coding these statements.

Output Control Features

The OUTPUT statement enables the programmer to specify:

1. The number of output copies which are to be printed.
2. Whether they are to be collated or uncollated.
3. If the copy modification (or spot carbon) feature is to be utilized.
4. Control of report offsetting.

Command	Left Part	Right Parts	Defaults	Interpretation
OUTPUT	COPIES =	number,	1	This right part "number" specifies the number of copies which are to be output. The "copies" option parameters on the START command allows this right part to be overridden by the operator when initiating a print job. The range for "number" is 1 to 32,767.
	COLLATE =	{ YES / NO } ,	YES	If YES is specified, the output pages are collated. If NO is specified, the output is not collated.
	OFFSET =	{ ALL / FIRST / NONE } ,	ALL	The default (ALL) results in an offset of each copy of each file. The right part "NONE" specifies that there is to be no offset at any time. The right part "FIRST" specifies that an offset is to occur only on the first copy of a file. The OFFSET control of "FIRST" or "ALL" copies may be modified by the ROFFSET command (see "Report Offsetting on a Record" in Chapter 4). An example of job offset control is shown in Figure 5-1.
	MODIFY =	{ cme-id / (cme-id, init [,copies]) / NONE },	NONE	The right part "cme-id" is the statement identifier of the CME statement. The CME statement must precede a reference to its identifier by the MODIFY left/right pair (or a CME control file may be on disk cataloged in the CME directory - see "Cataloged CMEs" in this chapter). The right part "init" specifies the initial ply (pass) to which the associated CME is to be applied. The right part "copies" specifies the number of plies (passes) on which to apply the CME. If "copies" is not specified then the CME will apply to all copies beginning with the copy number specified by the right part "init". The right part "NONE" specifies that no CME is to be used. Data will be processed without modification.

Command	Left Part	Right Parts	Defaults	Interpretation
OUTPUT (cont.)	MODIFY = (cont.)			Different CMEs may be associated with different copies of a report by the use of several CME specifications within a single MODIFY command or by the use of multiple MODIFY left parts on the same OUTPUT command. CME copy ranges may not overlap. Only one CME may be specified per report ply (pass). See the MODIFY DJDE command described in Chapter 6 for a discussion of how CMEs may be changed on a page-by-page basis within a report ply.
	COVER =	{ FRONT (FRONT,SEP) BACK BOTH (BOTH,SEP) NONE },	NONE	The left part "COVER" allows specification of the number and type of cover pages to be picked from the auxiliary feed tray. The right parts are defined as follows: "FRONT" specifies that a cover page is to be picked at the front of each report and will be the first page of the report. "SEP" specifies that each front cover will not have report data printed on it. "BACK" specifies that a cover page is to be picked at the end of each report. No report data is printed on back covers. "BOTH" specifies that both front and back cover pages are to be picked and the front of each report will be the first page of the report. "NONE" specifies that no cover pages are to be picked.

Command	Left Part	Right Parts	Defaults	Interpretation
OUTPUT (cont.)	FORMAT =	pde-id,	FMT1	Page Descriptor Entries (PDEs) which are used in the formatting of the printed output are referenced via the left part "FORMAT". The right part "pde-id" references a PDE which must be defined previously in a Job Descriptor Library or may be a reference to a PDE file separately cataloged in the PDE library on disk. A set of standard "pde-id"s are defined in Table 5-1 (i.e., FMT1, FMT2, etc.). Details on the PDE command are discussed in section "Page Descriptors" of this chapter. Creating and compiling PDE files with the PDL processor is discussed in "PDE and CME Compilation" in Chapter 3.
	FORMS =	{form-id (form-id, init [,copies])}, NONE	NONE	The left part "FORMS" allows for the specification of forms to be associated with the report copies. Different forms may be associated with different copies of a report by the use of multiple FORMS left parts on the same OUTPUT command. The right part "form-id" specifies a 1-6 character file name which exists on disk. This file is created by compiling Forms Description Language commands with the FDL system task (see Chapter 10 for further details on FDL). The right part "init" specifies the beginning ply (pass) number to which a specified form applies. The right part "copies" specifies the number of plies (passes) to which a specified form applies. If "copies" is not defined the last (or only) form specified will apply to all copies beginning with copy number "init". If the form is not the last one specified, then "copies" defaults to 1.

Command	Left Part	Right Parts	Defaults	Interpretation
OUTPUT (cont.)	FORMS = (cont.)			If both "init" and "copies" are not specified then the form will apply to all copies of the report.
	NUMBER =	{(pnum,lnum,cnum) / NO},	NO	The left part "NUMBER" specifies whether or not page numbering is to be performed. The first font specified in the "FONTS =" left/right part (of the PDE command) is used to print the page numbers (see "Page Descriptors" in this chapter). The right parts are defined as follows: "pnum" specifies the starting number for page numbering. "lnum" specifies the line number on which the page number is to be placed. "cnum" specifies the ending column number for page digit sequence. "NO" specifies that no page numbering is to be performed.

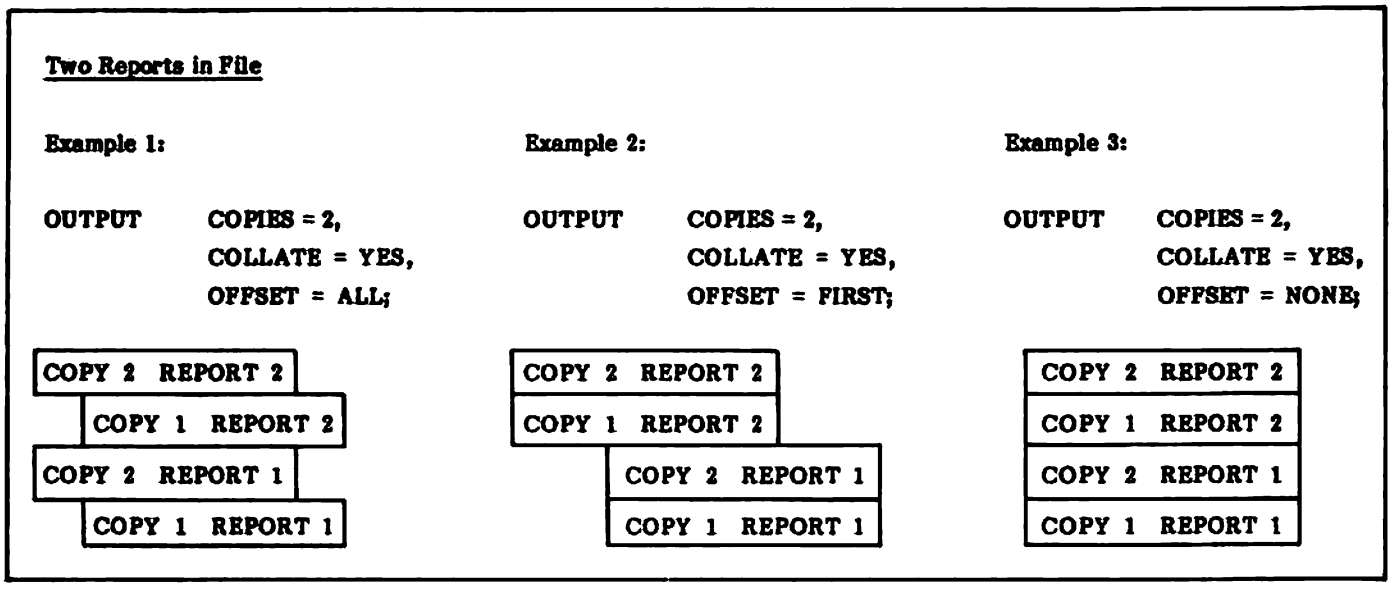


Figure 5-1. Results of Job Offset Control Options

Table 5-1. OSS Standard Formats (PDEs) Available on System Tape

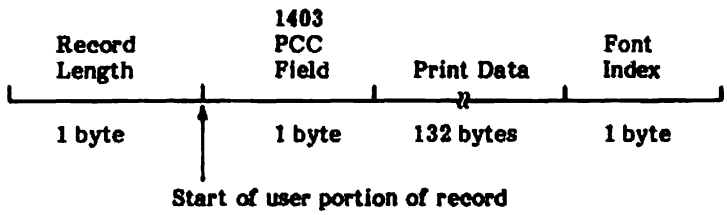
PDE ID	Number of Lines	Number of Columns	LPI	CPI	Approximate Point Size	Mode	BEGIN Values	Default FONT ID
FMT1	66	132	8.1	13.6	9	L	(.18,.66)	L0112B
FMT2	66	150	8.1	15	9	L	(.18,.50)	L0212A
FMT3	88	132	10.7	13.6	7	L	(.14,.66)	L0312A
FMT4	88	150	10.7	15	7	L	(.14,.50)	L0412A
FMT5	49	100	6	10	12	L	(.17,.50)	L0512A
FMT6	80	100	8.1	13.6	9	P	(.57,.58)	P0612A
FMT7	60	90	6	12	12	P	(.50,.50)	P07TYA
FMT8	60	75	6	10	12	P	(.50,.50)	P0812A
FMT9	80	200	10.1	20.0	7	L	(.50,.50)	L0912A
FMT10	132	132	12.5	17.6	6	P	(.22,.51)	P1012A
FMT11	132	150	12.5	20.0	6	P	(.22,.50)	P1112A

The names in the "PDE ID" column above are standard print formats which may be refered to in the FORMAT = pde-id left/right part of the OUTPUT command. If a print format is used which has "Number of Lines" greater than 66 then the VFU command must be used to set the TOF and BOF parameters.

Print Line Structure

The LINE statement describes the location and format of the print line data on the input tape. For example, the print data offset specifies the number of bytes between the start of the user portion of the record and the first character of that record which is to be printed. Likewise, the print length specifies the number of characters in the longest print line in the file.

Example:




```
LINE DATA = (1, 132);
```

The LINE statement also defines the width of the left-hand margin of the print page, and the carriage control convention that is to be applied to the print data file (ANSI, XEROX, etc.). The installation may also define its own carriage control convention if desired. In the preceding example, the following statements could be coded:

```
LINE  PCC = (0, NOTRAN), PCCTYPE = IBM1403;
```

The LINE statement allows the user to specify in a user data record on tape a specific font to be used. The user specifies a font by including an index value (which indicates which font in the FONTS left/right part of a PDE command) into the data record. This byte position of the font index is defined by the FONTINDEX left/right part of the LINE command. In the above example, the statement would be coded as:

```
LINE  FONTINDEX = 133;
```

Command	Left Part	Right Parts	Default	Interpretation
LINE	MARGIN =	{ value | (value { b | , } { IN | CM | POS }) },	(1, POS)	The left part "MARGIN" provides for specification of the left margin on a physical page. "value" is a value of the form "nnn.mm" (a decimal number with up to 2 digits to the right of the decimal point) which is the distance for the left margin and can be specified in inches (IN), or centimeters (CM). "value" must be specified as an integer ("nnn") for character positions (POS).
	DATA =	(pdo, length),	(1, 132)	The right part "pdo" (print data offset) is the number of bytes between the start of the user portion of the logical record and the first character of the record which is to be printed.

Command	Left Part	Right Parts	Default	Interpretation
LINE (cont.)	DATA = (cont.)			The right part "length" specifies the maximum length of printable data within each logical record. The user has the ability to print anywhere on the surface of an 8.5 by 11.0 inch paper and to specify the top and left margins to be used for output.
	PCC =	(offset, { NOTRAN / TRAN }),	(0,NOTRAN)	The right part "offset" specifies the byte offset within each logical record to the printer carriage control (PCC) field. The right part { NOTRAN / TRAN } specifies whether or not the printer carriage control byte is to undergo code translation along with the remainder of the record. TRAN indicates that the byte is to be translated; NOTRAN prohibits translation.
	PCCTYPE =	{ ANSI / B2500 / B2700 / B3500 / B3700 / B4700 / B6700 / H6000 / H2000 / IBM1401 / IBM1403 / UNIVAC / US70 / XEROX / id / USER / NONE },	ANSI	The right part specifies the carriage control convention which is to be utilized in printing a job. Creation of a user-defined PCC table referenced by either an identifier "id" or the keyword USER is defined using the PCC statement. See "Printer Carriage Control" section of this chapter. The PCCTYPE left/right part determines the INITIAL parameter (defined under the PCC command) as follows: If PCCTYPE = ANSI, then INITIAL = BOF. If PCCTYPE = USER or id, then the INITIAL parameter is obtained from the corresponding PCC command. If PCCTYPE is anything else, then INITIAL = TOF.

Command	Left Part	Right Parts	Default	Interpretation
LINE (cont.)	FONTINDEX =	{ offset / NONE } ,	NONE	The left part "FONTINDEX" allows the user to specify in a user record a specific font to be used. The right part "offset" specifies the byte position, relative to zero, in the user data record which contains an index into the set of fonts as defined in the PDE command. The font index in the data record is a number in the range 1 to n, where n is the number of fonts specified in the PDE command. The right part "NONE" specifies that there is no font index.
	OVERPRINT =	({ PRINT / IGNORE / MERGE } , { DISP / NODISP }),	(PRINT, NODISP)	The right part { PRINT / IGNORE / MERGE } specifies the manner in which overprint lines will be handled. "PRINT" specifies that all overprint lines are to be printed normally. "MERGE" is the same as "PRINT". If "IGNORE" is coded, all overprint lines are ignored. "DISP" and "NODISP" are Xerox 1200 Computer Printing System options which are preserved here for compatibility purposes. Neither is functional. The number of overprint lines will always be printed on the accounting page.
	VFU =	vfu-id;		The right part "vfu-id" is the statement identifier of the VFU table which must precede this reference to it. (See "Defining Print Line Positions", below). The VFU table defines print line positions corresponding to skip to channel commands for the job or jobs which are to be processed.

Defining Print Line Positions

The VFU statement is used to assign output line numbers to printer carriage control channels. These line to channel assignments perform the same function as the printer carriage control tape on a conventional line printer. The statement is also used to assign line numbers to the "top-of-form" and the "bottom-of-form". Top-of-form indicates the number of lines from the top of an output page to the first print line on the page. Likewise bottom-of-form indicates the number of lines from the top of an output page to the last print line on the page.

Top and bottom of form are used for pre-job page alignment and for page overflow processing. For all PCCTYPEs except ANSI and user defined PCCs, the pre-job page alignment is to top-of-form in the expectation that the first carriage control command of the job will be print and space one line, or something similar. Selection of ANSI causes alignment to bottom-of-form to handle the skip to channel 1 and print command, which usually begins a job of that carriage control type. User defined PCCs may set alignment at either TOF or BOF.

Page overflow occurs when spacing to the next line causes the bottom-of-form line number to be exceeded. Page transition occurs, and line spacing is continued from the top-of-form line number. Honeywell 2000 carriage control and Xerox carriage control are exceptions to this processing (see 9700 Tape Formats manual).

Command	Left Part	Right Parts	Default	Interpretation
ac: VFU				"ac" is a 1 to 6 character statement identifier that is referenced by the VFU = id portion of the LINE command. One character must be alphabetic; the others may be alphabetic or numeric.
	ASSIGN =	{(channo, lineno), (channo, (lineno,...,lineno))},	none	The right part "channo" is the number of the channel which is being assigned. It is an integer in the range 0 ≤ channo ≤ 15. The user may end the VFU statement with a semicolon and start another VFU statement without an id field to continue specification for the same channel or a different channel. Example: V1: VFU ASSIGN = (1,(7,14,28,35)); and V1: VFU ASSIGN = (1,(7,14)); VFU ASSIGN = (1,(28,35)); are equivalent in application. The right part "lineno" is the number of the output print line being assigned to a particular channel.

Command	Left Part	Right Parts	Default	Interpretation
VFU (cont.)	TOF =	value,	1	The right part "value" specifies the number of lines from the top of the output page to the first print line on the page.
	BOF =	value,	66	The right part "value" specifies the number of lines from the top of the output page to the last print line on the page.

Points to Note

1. The VFU statement must precede the print line structure statements "LINE VFU =" described in the previous section of this chapter.

2. Top-of-form and bottom-of-form specifications are independent of channel assignments.

3. Top-of-form should be less than or equal to the smallest channel value assignments.

4. Bottom-of-form should be greater than or equal to the largest channel value assignments.

5. There are no default assignments for any channel, including channels 1 and 12.

6. Any unspecified channel assignemnt causes a print and space 1 line operation. Under some vendor formats, the default may be space 1 line and print, according to what is consistent with the carriage control set.

7. Multiple line numbers may be assigned to the same channel number. This simulates the vertical tabbing feature of an impact line printer where a skip to channel command causes transition to the next punched hole in the specified channel of the paper tape. This tape, which controls the printer, facilitates spacing a fixed number of lines down the print page. There may be multiple punches in any vertical format channel on the impact printer's tape. As shown in the example in Figure 5-2, a skip to channel command in the 9700 causes selection of the next line number in the ASSIGN list (for that channel) larger than the current line number, with page transition and alignment to the first line number in the list if none is larger.

8. An example of a coded VFU statement is shown in Figure 5-2.

```
V1: VFU     ASSIGN = (1,5),
            ASSIGN = (2,(10,15,20,25,30,35,40,45,50)),
            ASSIGN = (12,55),
            TOF = 5, BOF = 55;
```

Interpretation

1. Top-of-form is assigned to line number 5; bottom-of-form is assigned to line number 55.

2. Three channels (1, 2, and 12) have been assigned. Channel 2 has been assigned to nine line numbers.

3. Assume the printing system is printing a report, and the current line number is eleven:

   a. If a "skip to channel one and print"[t] command is issued, a page transition will occur. Printing begins on line 5 (assigned to channel one), which is top-of-form on the new page.

   b. If a "skip to channel two and print" command is issued when the current line number is 11, the next line to be printed is line 15 of the current page. Lines 10, 15, 20, and so on, are also assigned to channel two, but since the current line number is 11, the next consecutive line number assigned to channel two greater than 11 is line 15.

   c. If a "skip to channel twelve and print" command is issued, the next line to be printed is line 55 of the current page. Line 55 is assigned to channel twelve.

Figure 5-2. VFU Statement, Coding Example

[t] Printer carriage control operations are defined by the user via the printer carriage control statement (PCC) described in the subsequent section. The standard carriage control tables are described under the corresponding vendor format description in the 9700 Tape Format manual.

Printer Carriage Control

The PCC statement enables the user to assign one-byte printer carriage control (PCC) codes and define their action. Line spacing, skip to channel, and printing actions are all defined via this command.

Command	Left Part	Right Parts	Default	Interpretation
[ac:] PCC	INITIAL =	{ BOF / TOF } ,	TOF	"ac" is a 1 to 6 character statement identifier that is referenced by the PCCTYPE = id .portion of the LINE command. One character must be alphabetic; the others may be alphanumeric. The right part { BOF / TOF } identifies the initial reference point for printing a page. If "TOF" is specified, the control program will perform the first spacing, skipping, or printing action from top-of-form. Likewise, if "BOF" is specified, the control program will perform the first spacing, skipping, or printing action from bottom-of-form.
	DEFAULT =	{ ANSI / B2500 / B2700 / B3500 / B3700 / B4700 / B6700 / H2000 / H6000 / IBM1401 / IBM1403 / US70 / UNIVAC / XEROX / ccln / NONE } ,	PSP1 (print and space one line)	The left part "DEFAULT" allows the installation to select a set of printer carriage control codes. A specific table may be selected or a "ccln" can be defined. The right part "ccln" specifies the action to be performed when a code has not been specifically assigned. The assignment codes for various actions are described under "ASSIGN =" (right part "ccln" below).

Command	Left Part	Right Parts	Default	Interpretation
PCC (cont.)	ASSIGN =	{(byte, ccln) (byte, (ccln,...,ccln))},		The right part "byte" is the printer carriage control byte being defined. Its value is in the range 0 ≤ byte ≤ 225 (X'00' to X'FF'). The right part "ccln" specifies the action that should be taken when the printer carriage control byte defined in "byte" above is encountered. The right part "ccln" may be any of the following definitions: ([{TOF, OVR, IGN,}] [{SPm SKn}] [{P N}] [{SPm SKn}]) If TOF is specified, and "byte" causes bottom-of-form (BOF) to occur, the printing system is to go to top-of-form (TOF) on the next page and stop spacing. If OVR is specified, and "byte" causes bottom-of-form (BOF) to occur, the printing system is to go to top-of-form (TOF) on the next page and continue spacing. If IGN is specified, and "byte" causes bottom-of-form (BOF) to be encountered, bottom-of-form is to be ignored and spacing is to continue through the end of the physical page when page transition to top-of-form occurs and spacing is continued. OVR is the default and need not be specified. The fields [{SPm SKn}] [{P N}] [{SPm SKn}] Field 1 Field 2 Field 3 specify the action that is to be taken when "byte" is encountered. Each of the three fields is optional; however, at least one field must be specified.)

Command	Left Part	Right Parts	Default	Interpretation
PCC (cont.)	ASSIGN = (cont.)			Field 1: SPm Space m lines before printing ($0 \leq m \leq 15$). SKn Skip to channel n before printing ($0 \leq n \leq 15$). Field 2: P Print the output data at the line number computed after Field 1 is processed. N No printing is to occur for this record. If Field 2 is not specified, no printing is to take place for this record. Field 3: SPm Space m lines after printing ($0 \leq m \leq 15$). SKn Skip to channel n after printing ($0 \leq n \leq 15$).
	MASK =	value,	X'FF'	The right part "value" specifies an eight-bit value to be ANDed with the printer carriage control byte being defined after translation, if any, to mask off bits from the code which are not relevant to the operation being specified by the code.
	ADVTAPE =	{YES / NO} ,	YES	The right part is used to specify whether the printing system is to advance to a new page when two successive channel skip commands are issued with no intervening print. For example, on most printers, the following actions: PSK1 Print and skip to channel 1 SK1N Skip to channel 1, do not print would cause a blank page to be output. However, on a 1403 printer, these actions would not cause a blank page to be output.

Command	Left Part	Right Parts	Default	Interpretation
PCC (cont.)	ADVTAPE = (cont.)			"YES" specifies that multiple skips will be honored. "NO" specifies that multiple skips will result in only one skip action being taken.

Points to Note

1. Multiple user-defined PCC tables are allowed; only one may be without a statement identifier. The corresponding PCCTYPE left part on the LINE command references each table via a statement identifier. The right part USER is used to reference the user-defined PCC table in which no statement identifier coded.

2. The "DEFAULT =" left-right parts must precede any "ASSIGN =" left-right parts. Any preceding "ASSIGN =" left-right parts will be replaced by the DEFAULT assignment.

3. The user may end a PCC statement with a semicolon and start another PCC statement to continue specification of the carriage control codes. Multiple PCC statements may be used within a single PCC table definition as long as there are no intervening non-PCC statements.

4. Consecutive "byte" right parts need not be specified. Thus, the statements

   ASSIGN = (X'60', SP1), ASSIGN = (X'61', SP2), ASSIGN = (X'62', SP3);

   can be coded in the single statement ASSIGN = (X'60', (SP1, SP2, SP3));

5. Figure 5-3 contains an example of a coded PCC statement.

```
PCCEX:  PCC     INITIAL = TOF, DEFAULT = PSP1, MASK = X'FF', ADVTAPE = YES,
                ASSIGN = (X'00'  ,(P, PSP1, PSP2, PSP3,PSP4,PSP5, PSP6,PSP7,
                                   PSP8,PSP9,PSP10,PSP11,PSP12,PSP13,PSP14,
                                   PSP15));
        PCC     ASSIGN = (X'40'  ,(P,SP1P, SP2P,SP3P,SP4P SP5P,SP6P,SP7P,
                                   SP8P,SP9P,SP10P,SP11P,SP12P,SP13P,SP14P,
                                   SP15P));
        PCC     ASSIGN = (X'80'  ,(PSK0,PSK1,PSK2,PSK3,PSK4,PSK5,PSK6,
                                   PSK7,PSK8,PSK9,PSK10,PSK11,PSK12,PSK13,
                                   PSK14,PSK15));
        PCC     ASSIGN = (X'C0'  ,(SK0P,SK1P,SK2P,SK3P,SK4P,SK5P,SK6P,
                                   SK7P,SK8P,SK9P,SK10P,SK11P,SK12P));
```

Figure 5-3. Coded PCC Statement, Example

Copy Modification Feature

INTRODUCTION

The Copy Modification feature (also referred to as spot carbon) offers the ability to modify 9700 output on a per copy basis. It allows the user to replace certain parts of report output on selected copies with specified static data, or to specify the placement of font change requests within variable data.

This feature is controlled through the use of the Copy Modification Entry (CME) command. CME defines a rectangular space upon the printed page within which printed data will be replaced with a substitution string. More than one CME may be applied to a job. CMEs may be coded as part of the job descriptor library or created as a separate file such that it may be referenced by one or more job descriptor libraries. This is described more fully in "Cataloged CMEs" of this chapter.

The MODIFY left part on the OUTPUT command (and the DJDE command MODIFY) relates the CME to the particular copies to be changed. The MODIFY left/right parts are described in a previous section of this chapter.

An example of CME is shown in Figure 5-4.

Command	Left Part	Right Parts	Default	Interpretation
ac: CME				"ac" is a 1 to 6 character statement identifier that is referenced by the MODIFY = cme-id portion of the OUTPUT command (or the MODIFY DJDE command).
	LINE =	$\left\{ \begin{matrix} n \\ (n,m) \\ (n,-) \end{matrix} \right\}$,	none	The right part "n" is the initial line of the Copy Modification rectangle. The right part "m" is the number of lines within the Copy Modification rectangle. If not specified, the information will apply only to the line indicated by "n". To indicate that the information is to apply to all lines on a page beginning with the line indicated by "n", a dash character ("-") may be used in place of "m".
	POS =	p,	1	The right part "p" specifies the initial character position of the rectangle in the print line. It must not exceed the position number of the last print position on a print line.

Command	Left Part	Right Parts	Default	Interpretation
CME (cont.)	CONSTANT =	sc,	none	The right part "sc" identifies the characters that will be printed within the rectangle. "sc" is any string constant. The width of the Copy Modification rectangle is determined by the number of characters specified by "sc". More than one "sc" is allowed.
	FONT =	value;	none	The right part "value" specifies the index into the font list on the "FONTS =" right part of the PDE command. (Note: a PDE command is selected by the "FORMAT =" right part of the OUTPUT command or the DJDE FORMAT command). "value" may range from 1 to n, where n is the number of different fonts specified by the FONTS option. A value of 1 specifies the first font in the FONTS option. A font specification applies to input variable data as well as static CME data. If a line number and character position but no insertion text are specified, the font change specified will apply to input variable data at the position specified.

Points to Note

1. The CME statement must precede the output processing statement which references it, or it must be represented in a control file cataloged in the CME directory.

2. There is no default for the CONSTANT = sc portion of the CME command. It must be specified.

3. In order for the character string specified to be printed on a line, the normal data must appear on that line; that is, if a line is skipped, either by line spacing or channel skipping, then no character subsitution will occur.

4. Copy Modification (spot carbon) is not implemented if a report is printed in the uncollated mode.

5. Within a text string, as specified with the CONSTANT = sc, the character "#" may be used as a lower-case toggle. When a text string is encountered, it is assumed that characters are to be inserted into the print line as they appear in the text string (in upper case normally). If a "#" is encountered the lower-case mode is invoked and all letters after the "#" will be considered lower-case until another "#" is encountered. The sequence "##" is used to indicate that the character "#" is to be inserted and is not to be treated as toggling lower-case-restrict mode.

6. Multiple lines may be specified and multiple columns may be specified for each line. Multiple line specifications must be given in ascending (top of page to bottom of page) order. Multiple column specifications for a line must be given in ascending (left to right) order. There also may be multiple text specifications following a column specification. These are combined to form a single text string. Additionally, font specifications may be specified at any point. The last font specified remains in effect until another font is specified.

EXAMPLE

The CME statement:

C1: CME LINE = (49,18), POS = 100, CONSTANT = (10)'*';

which when referenced by an OUTPUT MODIFY = C1 command would result in a page that looks like:

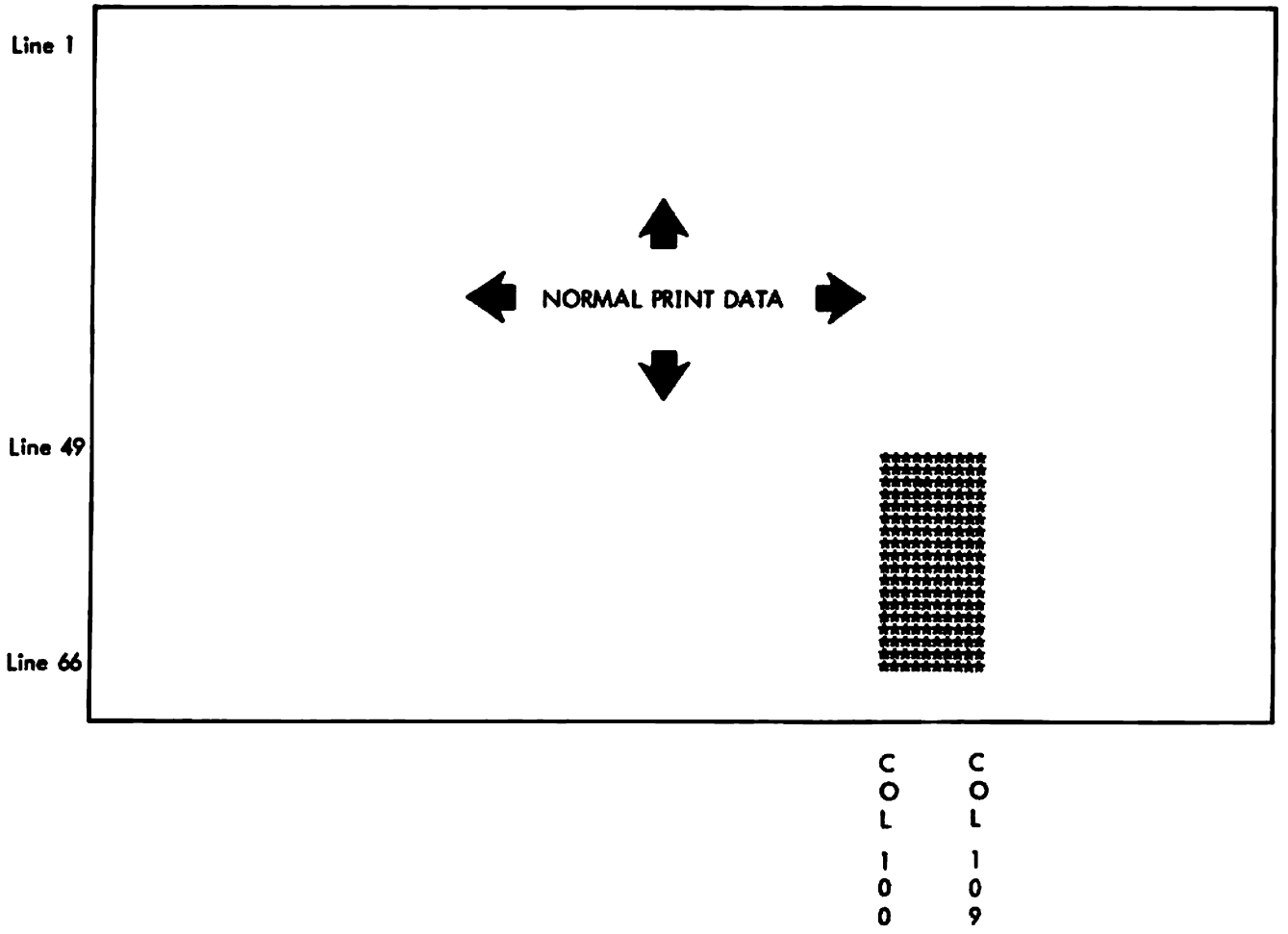


Figure 5-4. Example of a CME

Short Form CME Specifications

CME specifications may be given in short form to minimize the specification length. In short form, only the first character of a keyword need be given, equals signs are omitted, and commas are not inserted except where necessary to avoid ambiguity. The following is an example of CME specifications shown in standard and abbreviated form.

Standard Form:

```
CME12: CME   LINE =47, POSITION = 1, FONT = 5,
             LINE = 48, POSITION = 1, FONT = 1,
             LINE = 49, POSITION = 10, CONSTANT = 'ABCD';
```

Short Form:

```
CME12: CME   L47, P1, F5, L48, P1, F1, L49, P10, C'ABCD';
or
CME12: CME   L47P1F5L48P1F1L49P10'ABCD';
```

Cataloged CMEs

CME commands need not be part of a user's job descriptor library. They may be created as a separate disk file and used as if they were part of the job descriptor library that references them. This is done by creating a file of CME commands to the referenced in a job descriptor library and using the PDL processor to compile them. PDL will create a control file on disk, cataloged in the CME directory.

When the CME is referenced (by OUTPUT MODIFY) in a job descriptor library, and the CME does not exist within the library the system will search the CME directory for the named CME, which will be loaded into memory for use in processing the report.

The DJDE command MODIFY can also be used to dynamically associate a cataloged CME file with report processing. Further details are contained in Chapter 6. For the "MODIFY = cme-id" DJDE command the CME statement must be cataloged on disk.

Page Descriptors

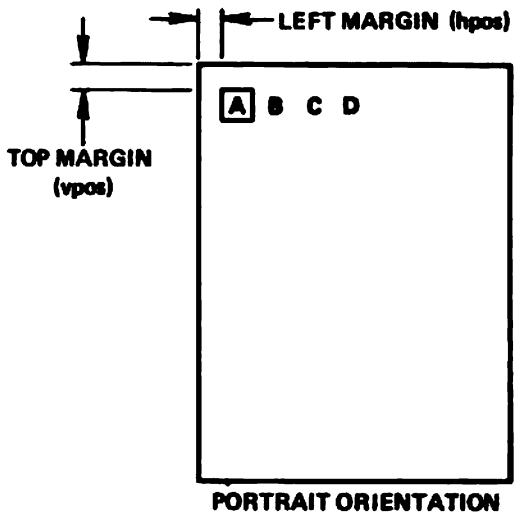
The PDE command specifies the Page Descriptor Entry (PDE) to be used. The PDE describes formatting information for each page of the report, including page orientation (landscape or portrait), location of the beginning print line for each logical page and the fonts to be used.

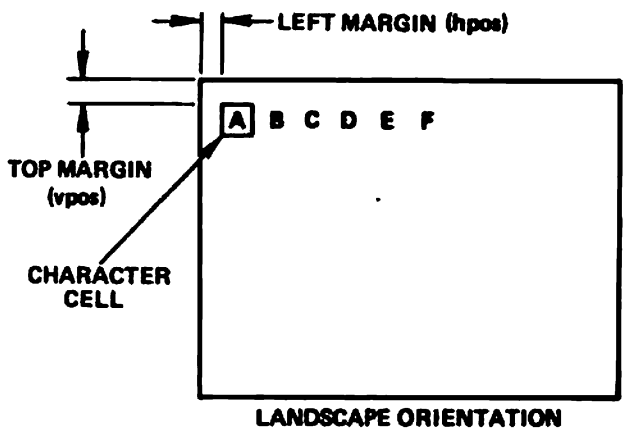
PDEs may be coded as part of the job descriptor library or created as a separate file such that it may be referenced by one or more job descriptor libraries. PDEs are called out on the "FORMAT =" left/right part of the OUTPUT command (or a DJDE FORMAT command). Typical PDE specifications are provided on the OSS system tape as described in Table 5-1. Figures 5-5, 5-6 and 5-7 illustrate the considerations in the formatting of a page with the PDE command.

PDEs created as a separate disk file may be used as if they were part of the job descriptor library that references them. This is done by creating a file of PDE commands (in the JSL directory) and using the PDL processor to compile them. PDL will create a control file on disk, cataloged in the PDE directory. When the PDE is referenced, the system will search the PDE directory for the named PDE, and if found, will load it into memory for use in processing a report. See "PDE and CME Compilation" in Chapter 3 for further details.

Command	Left Part	Right Parts	Default	Interpretation
ac: PDE	PMODE =	{ PORTRAIT / LANDSCAPE } ,	LANDSCAPE	The left part "PMODE" specifies the printing mode for each physical sheet. "LANDSCAPE" indicates that printing is to be parallel to the long edge of paper. "PORTRAIT" indicates that printing is to be parallel to the narrow edge of paper. Figure 5-5 illustrates these two print modes.
	FONTS =	{ (f1[[, f2]...fn]) / ((f1, s1)[[,(f2, s2)]....]) } ,	none	FONTS specifies the fonts to be used in printing variable input and CME data. Each "fi" (i = 1, n) specifies a 1-6 character identifier corresponding to a font cataloged on the system disk. Each "si" value specified an optional override line-spacing value to be associated with the font. Each "si" spacing value is a decimal value specifying lines per inch. If an override line spacing value is specified, then lines printed using the font will cause the indicated line spacing to be performed after the line using the font. If different fonts are specified on the same print line, then the line spacing value specified for the font of the last character in the line will be used to determine the position of the next print line.

Command	Left Part	Right Parts	Default	Interpretation
PDE (cont.)	FONTS = (cont.)			At least one font must be specified in a PDE command. See the section "Copy Modification Feature" in this chapter for further information regarding change of fonts within a print line.
	BEGIN =	(vpos [CM / IN], hpos [CM / IN]),	(0, 0)	The "BEGIN" left part specifies the location of the starting print line for each logical page on a physical page. "vpos" specifies the vertical position of the first character of the first print line on a logical page. IN specifies inches; CM specifies centimeters. The default type measurement is inches. "vpos" may be specified as a decimal number with up to three digits to the right of decimal point (e.g., 0.563 IN, 2.35 CM, and 4.3 are all legal specifications). "hpos" specifies the horizontal position of the first character of the first print line on a logical page. IN and CM have the same meaning as for "vpos". Multiple BEGIN options may be specified to accommodate multiple logical pages on a single physical page. In specifying the location of the beginning of a print line on a logical page, measurement is performed by viewing the physical page in the mode in which it is to be printed, e.g., landscape or portrait.





Note: Measurements refer to the location of the character cell that contains the first character to be printed.

Figure 5-5. Vertical and Horizontal Positions in Landscape and Portrait Modes

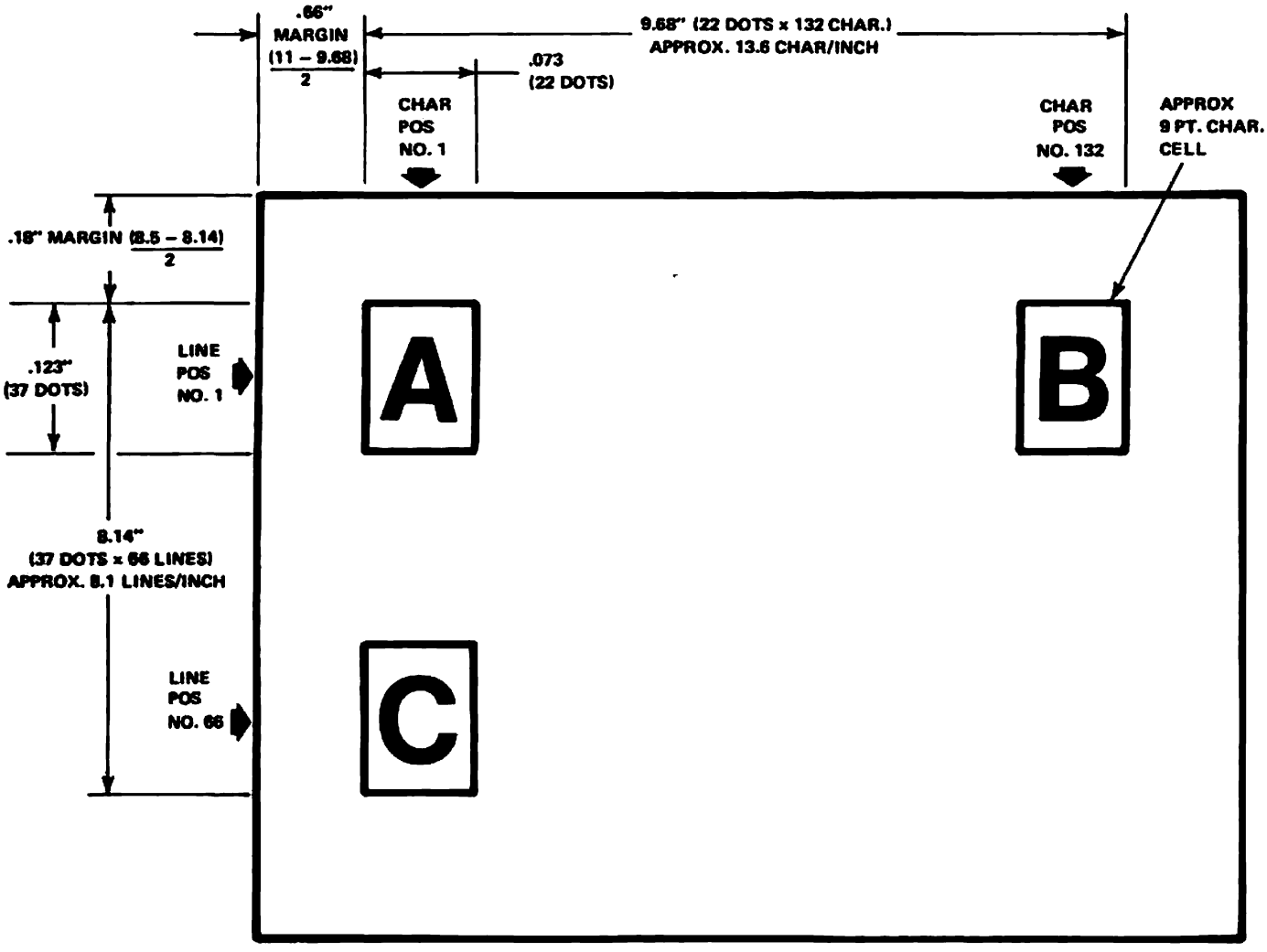


Figure 5-6. FMT1: Equavalent Impact Printer Format 6 Lines/Inches: Landscape Orientation

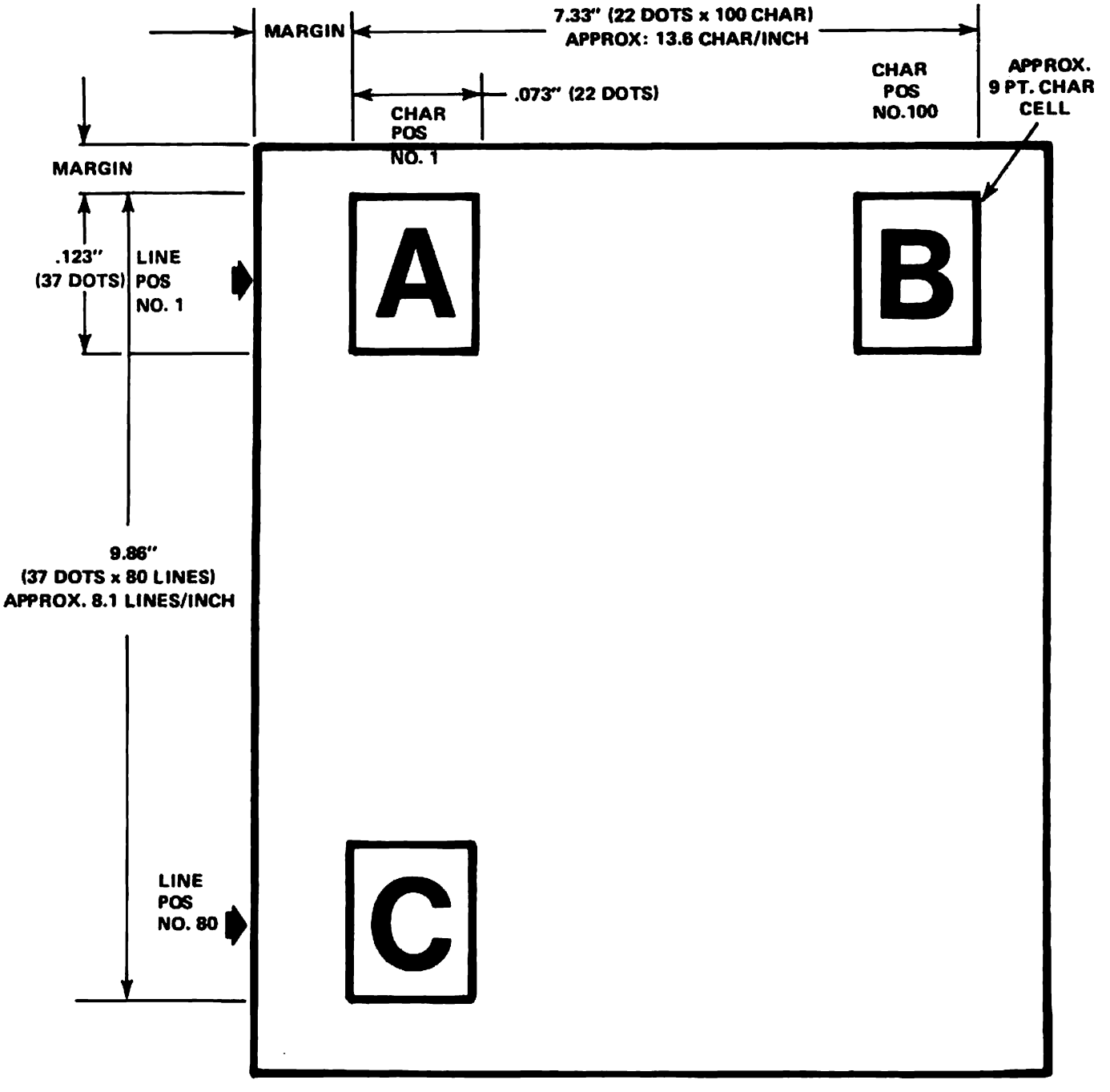


Figure 5-7. FMT6: Equivalent Impact Printer Format, 6 Lines/Inch: Portrait Orientation

Operator Message Commands

Two commands are provided to permit the user to inform the operator of special conditions: MESSAGE and ROUTE. Examples of these commands are contained in Figures 5-8 and 5-9.

Command	Left Part	Right Parts	Default	Interpretation
MESSAGE				The MESSAGE command causes a message to be displayed to the operator via the keyboard/display unit during job processing or job printing. Multiple ITEXT and OTEXT messages may be specified in the same or in different MESSAGE commands. If multiple messages are specified for the same pass number, all such messages will be displayed as a group before processing or printing is commenced.
	ITEXT =	{sc / (sc ,passnum)},	none	The left part "ITEXT specifies a message to be displayed during input processing. "sc" specifies the text message to be output (1-31 characters). "passnum" optionally specifies the pass to which the message text applies The message will be output to the operator just before processing of the indicated copy ply (pass) is commenced. If no pass number is specified, the indicated message will be output at the beginning of the first pass.
	OTEXT =	{sc / (sc [,passnum][,WAIT)]',}	none	The left part "OTEXT" specifies a message to be displayed during job printing. "sc" specifies the text message to be output (1-31 characters). "passnum" optionally specifies the pass to which the message text applies. The indicated message will be output to the operator prior to beginning printing of the specified report ply. If no pass number is specified, the message will be output once at the beginning of printing the entire report.

Command	Left Part	Right Parts	Default	Interpretation
MESSAGE (cont.)	OTEXT = (cont.)			WAIT optionally specifies that after the text is displayed, printing is to be suspended until the operator has responded with a CONTINUE command.
ROUTE				The ROUTE command is used to print a message on a separate sheet of paper preceding a report ply or an entire report. This command is useful for printing routing information so that different report copies may be routed to desired destinations.
	RTEXT =	{ sc (sc ,passnum [,line][,col]) },	none	"sc" specifies the message to be printed (1-132 characters). It will be printed with the first font specified in the "FONTS =" left/right part of the PDE command. "passnum" optionally specifies the pass (copy ply) to which the text applies. If not specified, the message will be printed at the beginning of the entire report. "line" optionally specifies the line number on which the first line of a block of RTEXT message is to be printed. The default is line 1 for the first text string of the pass. Otherwise the default is the next line of the page. "col" optionally specifies the column number at which the first character of a block of RTEXT message is to be printed. The default is column 1 for the first text string of the pass or if "line" is specified. Otherwise, it is the same as the previous column number.
	RFORM =	form-id;	none	The left part "RFORM" specifies a form to be printed on the separate sheet preceding the report ply or entire report. "form-id" is the name of a file cataloged in the FDL directory. It is created by compiling FDL source statements with the FDL processor. Further information on forms and the FDL processor are contained in Chapter 10.

The text: $\begin{Bmatrix}\text{USER 1} & \text{USER 2}\\ \text{BLDG 1} & \text{BLDG 2}\end{Bmatrix}$ is to be printed (in center of page) preceding the respective copies of a two copy landscape report. The page is 132 columns by 66 lines. The following ROUTE command will accomplish this:

```
ROUTE   RTEXT = ('USER 1', 1, 33, 64),
        RTEXT = ('BLDG 1' 1, 34, 64),
        RTEXT = ('USER 2', 2, 33, 64),
        RTEXT = ('BLDG 2', 2, 34, 64);
```

Figure 5-8. ROUTE Command Example

The following MESSAGE command will inform the operator that blue paper is required for copy 2 of a 4 copy report. Printing will be suspended at the appropriate point so that the operator can load the paper.

```
OUTPUT    COPIES = 4;
MESSAGE   ITEXT = ('COPY 2 WILL NEED BLUE PAPER'),
          OTEXT = ('LOAD BLUE PAPER', 2, WAIT),
          OTEXT = ('LOAD WHITE PAPER', 3, WAIT);
```

Figure 5-9. MESSAGE Command Example

Job Accounting

The ACCT statement enables the user to request that an accounting summary be output at the end of the printing for each report. This summary consists of a single page of information containing job set-up information and counts of processing events. An example of an accounting summary is shown in Figre 5-10.

Command	Left Part	Right Parts	Default	Interpretation
ACCT				The ACCT command permits an accounting summary to be printed at the end of each printed report for accoutning purposes.
	USER =	{ NONE BIN TRAY (BIN, TRAY) (TRAY, BIN) } ,	BIN	The right part "NONE" specifies that no accounting summary is to be output. Otherwise, the right part specifies where the one-page accounting summary is to be output: BIN (Summary is output to selected output bin only) TRAY (Summary is output to sample print tray only) {(BIN,TRAY) (TRAY,BIN)} (Summary is output to both the selected output bin and the sample print tray)
	DEPT =	sc ;	none	The left part "DEPT" associates a department code with the JDL for accounting purposes. The right part "sc" is a string constant of up to 31 characters representing a department code under which the accounting information will be maintained. Only alphanumerics (A thru Z, 0 thru 9) and the character ":" are allowed for the department code. If no department code is given, accounting information will be maintained on a JDL name basis. The system level command ACCOUNT must also be used in conjunction with "DEPT = sc" to add a department name. See "Accounting File Maintenance" in Chapter 7 for further details.

```
DATE:                  15 MAY 78  AT  12:16:20
DEPARTMENT:            SYSTST: JDL
JOB ID:                REPORT NO. 1
FILE ID:
INPUT PROCESSING TIME:          00:00:14.057
OUTPUT PROCESSING TIME:         00:00:24.550
JOB COMPLETION CODE:
     PAGES PRINTED:             90
     SAMPLE PAGES:              1
     PAPER PATH HOLES:          1
     LINES PRINTED:             450
     TAPE MOUNTS:               1
     BLOCKS READ:               16
     BLOCKS SKIPPED:            0
     RECORDS READ:              450
     DJDE RECORDS READ:         0
     NUMBER OF COPIES:          1
     OVERPRINTS:                54
     COLLATE:                   YES
     SF/MF:                     SINGLE
JDL/DJE USED:          TASKS$/DMPHEX

INITIAL FONT LIST:              LO112B
INITIAL FORM LIST:              -NONE
INITIAL CME LIST:               -NONE
```

Figure 5-10. Sample Accounting Page

Abnormal Condition Handling

The ABNORMAL statement enables the user to implement optional features which enhance output integrity. The coding procedures, features, and interpretation for the ABNORMAL statement are described in the subsequent table.

Command	Left Part	Right Parts	Default	Interpretation
ABNORMAL	RES =	sev-level,	9999	The right part "sev-level" specifies a severity level for error messages. Any error message of this severity level or greater will result in the job being aborted. Appendix A provides a list of all the system error messages and their severity level. The severity level is the 4 digit value after the characters "OS".
	BLKSP =	{ YES / NO } ,	YES	The right part "YES" specifies that the operator will be allowed to perform tape block spacing. When the right part "NO" is specified, block spacing operations will not be permitted. Attempts by the operator to do so will be treated as invalid.
	SECURITY =	{ YES / NO } ,	NO	The left part "SECURITY" specifies a blanket security flag. The right part "YES" specifies that the following restrictions on operator procedures are enforced during the processing and printing of a job: 1) No block spacing is permitted. Same as BLKSP = NO described above. 2) No command file DJDE parameters may override job parameters of a job descriptor entry. 3) No page spacing is permitted during the printing of a job. 4) No sample print is allowed.

Command	Left Part	Right Parts	Default	Interpretation
ABNORMAL (cont.)	REP =	{YES / NO} ;	NO	The left part "REP" specified whether a sheet will be placed in the bin whenever an unusual condition is encountered during the processing of a report. The right part "YES" specifies that a page will be delivered to the bin describing the problem or event encountered during job processing. It will be offset from the remainder of the output in the bin for easy identification. The right part "NO" specifies that no page will be inserted in the output stream.

6. DYNAMIC JOB DESCRIPTOR ENTRIES

Introduction

Dynamic Job Descriptor Entries (DJDEs) are used to dynamically modify the printing environment established by a JDE. DJDE commands modify previously established JDE entries via an operator initiated command file or as part of the input data tape. Dynamic job descriptor entry processing enables certain JDE parameters to be changed on a report-to-report or page-to-page basis. Some of the benefits derived from changing these job parameters are as follows:

- The printing system does not stop between reports nor is operator intervention required.
- Forms may be changed on a page-to-page basis.
- Many variations on VFU channel, margin, and top- and bottom-of-form assignments may be applied to reports as they are created instead of being stored in the 9700 via job descriptor entries. Thus, the number of JDEs required for job processing is reduced.
- Unusual processing requirements may be satisfied through DJDE adjustments to the print line length and starting print position.
- The required number of copies is automatically generated, with routing or distribution notification sent to the operator.

DJDE Applications

A very useful application of the DJDE is when the printing system is processing in the multi-file (or multi-report) mode. In this mode, the operator starts up a print tape on the 9700 and typically returns to the device only when minor operational activity is required. The options which may be considered when updating applications to include a DJDE are as follows:

Delimiter Mode Stacked Reports

- In the report body
- Adjacent to, and after, the delimiter records
- Within the delimiter records
- A combination of the previous three options

Change Mode Stacked Reports

- Within the body of the stacked report, where the DJDE records contain the same change field contents as the report to which the DJDE applies.

A File Without Stacked Reports

- Within the file

DJDE Commands

Table 6-1 contains a summary of DJDE commands. These commands are a subset of the previously discussed PDL commands. The details of coding these commands and their syntax are discussed in the following sections.

Table 6-1. DJDE Command Summary

DJDE Command	Report Oriented	Page Oriented	Description
JDL or SYSTEM	X		Invokes the named JDL at the next report boundary.
JDE or JOB	X		Invokes a JDE within the specified JDL at the next report boundary.
COPIES	X	X	If COLLATE = YES, specifies the number of copies to be printed starting at the next report boundary. If COLLATE = NO, specifies the number of copies starting at the next page boundary.
FORMS		X	Specifies the form to be merged on printed pages.
COLLATE	X		Invokes collated or uncollated mode at the next report boundary.
MODIFY		X	Specifies the Copy Modification Entries (CME) to be used for variable data replacement and/or font switching operations on input data.
FORMAT		X	Specifies the Page Descriptor Entry (PDE) to be used to set up formatting control.
NUMBER	X		Specifies page numbering control for the next report.
MARGIN		X	Specifies left printing margin within a logical page.
DATA		X	Specifies location and length of printable data within a user's input record.
OVERPRINT		X	Specifies overprint control for input data.
ASSIGN		X	Specifies an assignment of a print channel to a page line position.
TOF		X	Specifies the line number corresponding to top-of-form.
BOF		X	Specifies the line number corresponding to bottom-of-form.
ITEXT	X		Specifies a message to be displayed to the operator during input processing. This becomes effective at the next report boundary.
OTEXT	X		Specifies a message to be displayed to the operator during job printing and, optionally, suspends printing. This becomes effective at the next report boundary.
RTEXT	X		Specifies routing information to be printed on a separate printed sheet prior to a report copy or an entire report. This becomes effective at the next report boundary.
RFORM	X		Specifies a form to be printed on a separate sheet prior to a report copy or an entire report. Becomes effective on the next report boundary.
FONTINDEX	X		Specifies whether or not data records contain a font index.
C			Permits inclusion of commentary in DJDE records.
END			Signifies the end of information within a DJDE.

Report and Page Oriented DJDEs

As listed in Table 6-1 there are two types of DJDE commands: report-oriented and page oriented. Report oriented commands are associated with the report (or report-ply) as a whole and are placed at the beginning of a report prior to the first data record or in an end-of-report delimiter record (to set up the environment for a subsequent report). Page oriented commands effect changes to specific pages within a report and can change these pages differently in different copies. Such commands may be placed within the report itself and would take effect at the next page boundary. They may also appear at report boundaries to effect changes to all pages in a report (or report-ply).

Tape and Disk DJDEs

There are two ways to use DJDEs to modify JDE parameters. First, via a user created DJDE file on disk which will modify the JDE when a print job is initiated. The other way is to create DJDE records on the print tape which will modify the JDE values as the tape is being processed. DJDEs on a disk file are used only at job initiation, whereas DJDEs on the user's tape can modify JDE values dynamically on a report or page basis.

The following discussion will be concerned with the DJDEs that are part of the input data tape. DJDEs initiated from disk files are discussed in section "Command File DJDEs" of this chapter.

DJDE Commands on Tape

DJDEs to be used as part of the input data tape consists of two parts. The first part is the Job Descriptor Library specification via the Print Description Language IDEN command. This specification notifies the system that DJDEs may be contained on a job input tape. Additionally, the specification describes the search criteria for identifying the DJDEs.

The second part is the actual DJDE record (or records) which are created as part of the job input tape reports. Each DJDE report record contains an identification field which matches the search criteria specified in the Job Descriptor Library. Additionally, each report record contains a series of left/right parts which describe the actual job descriptor entry changes to be applied to the report.

The subsequent two sections of this chapter describe the two parts of the DJDE in detail.

JDL Specification: IDEN Statement

As stated previously, the Print Description Language IDEN statement is used to notify the system that a DJDE record (or records) may be contained on the job input tape. Further, the statement describes the characteristics of the DJDE record prefix so the system can identify the record. The statement syntax and interpretation are described in Table 6-2. An example of a coded IDEN statement is shown in Figure 6-1.

Table 6-2. IDEN Syntax and Interpretation

Command/ Identifier	Left Part	Right Parts	Default	Interpretation
IDEN	PREFIX =	sc,	none	The right part "sc" is the DJDE prefix. "sc" is a byte string of up to 255 characters represented as a hexadecimal, octal, or character constant. "sc" must be defined in all DJDE report records.
	SKIP =	value,	1	The right part "value" specifies the number of bytes (beginning at 0) from the beginning of the user portion of the record to the beginning of the DJDE. In other words, it is the offset to the starting column of the DJDE left/right parts.
	OFFSET =	value,	0	The right part "value" specifies the number of bytes (beginning at 0) from the beginning of the user portion of the record to the beginning of the prefix string constant of the DJDE record.
	OPRINFO =	{ YES / NO };	NO	The right part "YES" specifies that a sheet which contains the contents of the DJDE is to be printed and delivered to the bin. The right part "NO" specifies that no sheet will be printed and delivered to the bin.

The IDEN statement coded below describes the DJDE report record shown in Figure 6-2.

```
IDEN PREFIX = '$TEST', SKIP = 9,
     OFFSET = 3, OPRINFO = YES;
```

Figure 6-1. Sample IDEN Statement Coding

DJDE Report Record Specification

DJDEs are created by the user as part of the job tape report. They are looked for by the system only if a DJDE identifier has been specified by an IDEN command within the JDE used to process the job. The DJDE data consists of one or more fixed format records each of which may be up to the record length specified for the job input data. For each record, the prefix (the identification field) and the DJDE left/right parts must begin in the same location. The DJDE records are terminated by an "END" statement. All specified DJDE information will be applied at the next page or report boundary after the "END" is encountered. There may be multiple DJDE command sequences in a job. Each set modifies only the specific parameters mentioned within the DJDE. The DJDE left/right part options are defined in Table 6-2.

The following points should be considered when preparing to introduce DJDE records into job tape reports. Examples of DJDE records are illustrated in Figures 6-2 and 6-3.

- DJDE records need not be consecutive, as none of the parameters are applied until "END"; is encountered. However, it is highly recommended that the DJDE records be consecutive, especially when used with delimiter records.

- Report oriented DJDE commands must appear within the report to which they apply but prior to the first data record; otherwise the commands will not be executed.

- The DJDE report record may contain more than one left/right part except for the case of a comment, which must be the only DJDE information in that record. Each left/right part within a record is separated from the next left/right part by a comma.

- The end of all DJDE left/right parts in a record is coded by either of the following:

  ,; (comma, semi-colon)
  ;b (semi-colon, blank)

- The end of the right part of a DJDE record which is split and continued on the next DJDE record is shown by:

  ,; (comma, semi-colon)

- The prefix in the DJDE record may appear after the DJDE left/right part(s) as long as it is consistent in location for all DJDE records.

- If the file containing the DJDE is variable-blocked, the program which blocks the file may strip off trailing blanks. Thus, if the comment record contains no actual comments, the blank following the "C" may be eliminated. If the "C" is the last character of the record, the command record will be accepted. However, any character other than a blank following the "C" will cause the record to be processed as a legitimate DJDE record, and not as a comment. See "C" command in Table 6-3.

- If a DJDE is always created because of coding procedures, but in fact there is no need to set any of the DJDE left/right parts for a particular report, a null DJDE may be created with only an "END;" statement and no other commands specified.

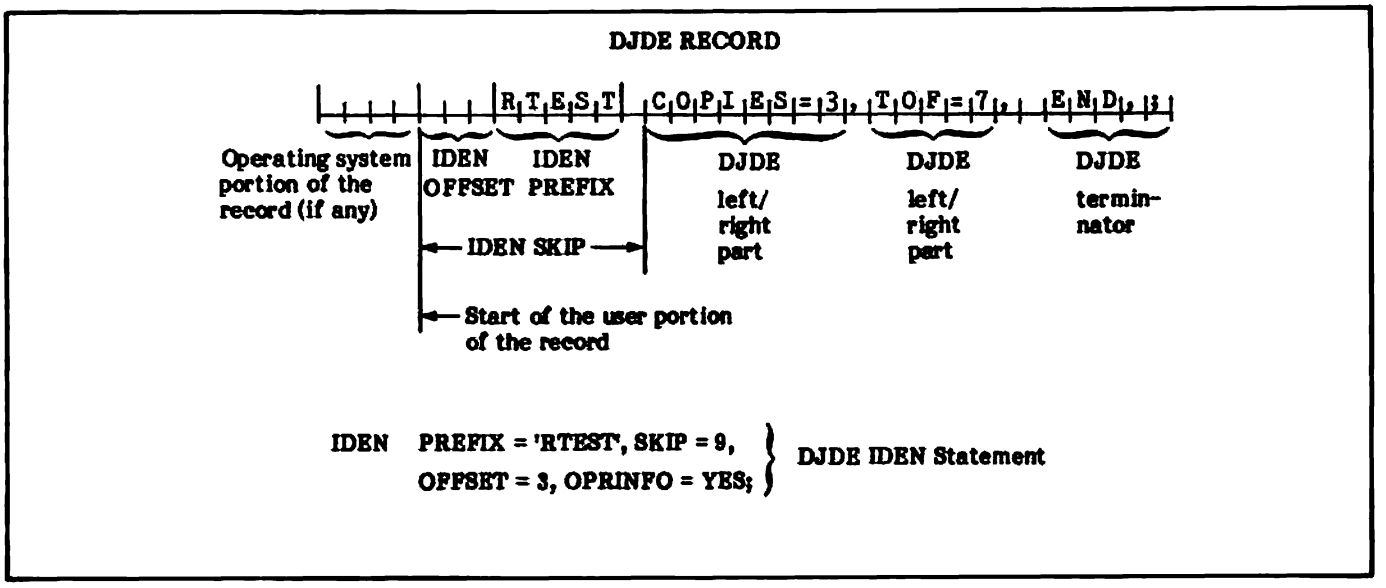


Figure 6-2. Single DJDE Record And IDEN Statement

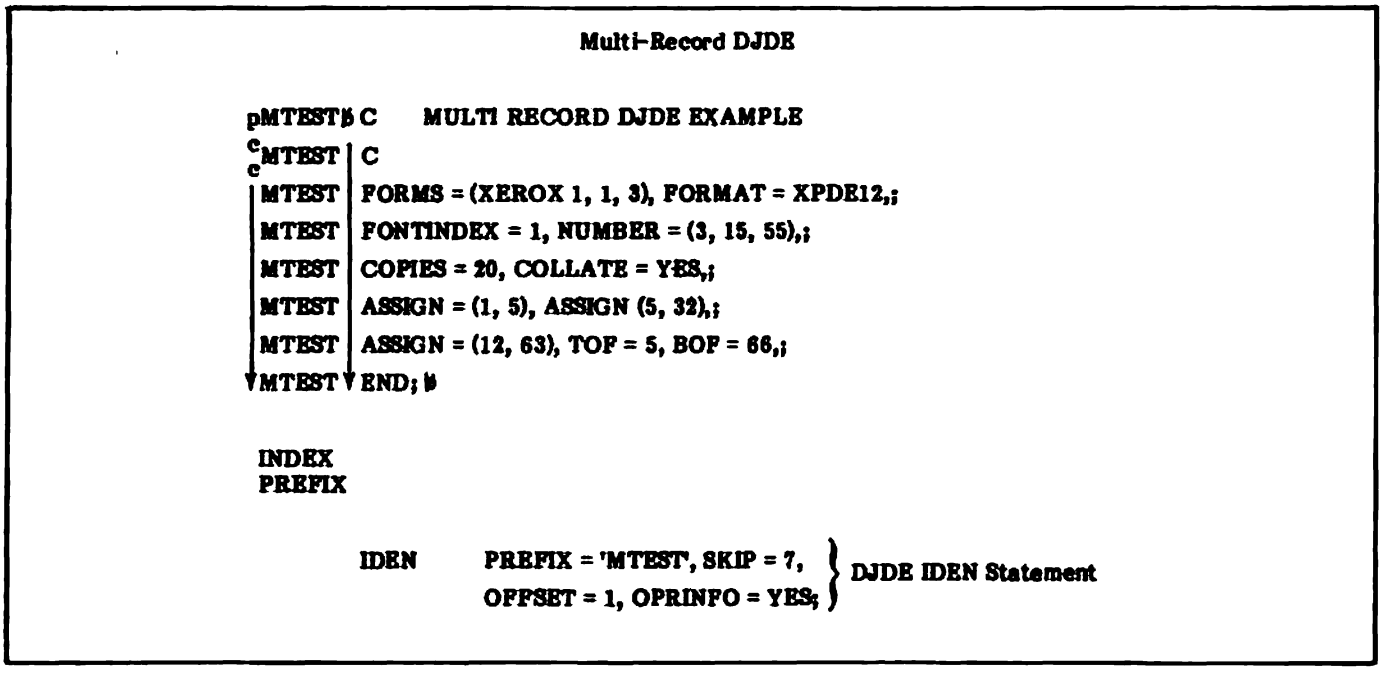


Figure 6-3. Multiple Record DJDE

DJDE Output Report Changes

The changes to the Job Descriptor Entry, specified in the DJDE, are incorporated when "END;" is encountered. The changes begin on the next page following the last DJDE record, except for report related parameters. These parameters take effect at the next report boundary.

The specification OPRINFO=YES in the job descriptor entry ensures that the DJDE records are printed and sent to the bin at the next page transition after an "END" DJDE record has been processed. Comments in the DJDE may be used for operator notification or output routing instructions in conjunction with the OPRINFO option.

DJDE records may be printed and delivered to the bin or sample print tray regardless of the OPRINFO option, provided the following conditions are present:

- The DJDE contains an error (records printed and are delivered to the bin).
- The DJDE occurs in stacked report delimiter records for which either the TRAY or BOTH option was specified.

Command File DJDEs

A DJDE command file is a disk file of DJDE commands (described in Table 6-3) which are incorporated into a user selected JDE when a print job is initiated. The command file is initiated by the operator as a parameter on the START command (START *filename) and will only modify printing envionment parameters at job startup time. The dynamic modification of JDE parameters on a report or page basis from an input tape is described in previous sections of this chapter.

DJDE Command File Creation

The DJDE file can be created via the EDITOR (described in Chapter 8) on the 9700 or written to tape on a host system and then entered into the 9700 file system. If the DJDE records are to be created on tape they must be in a fixed, unblocked format with no carriage control and a maximum record length of 133 characters. Only the first 72 characters of a record will be used for command information.

DJDE command files must be cataloged in the CMD file directory before it can be initiated. A file created using the EDITOR can be put into the CMD file directory by specifying the CMD "file-type" on the SAVE command. A file on tape can be copied into the CMD directory by specifying CMD as the outfile "type" (see "Copying a File" in Chapter 7).

Each record of a DJDE command file will contain a series of left/right parts which describe the Job Descriptor Entry changes to be applied to the print job. Each left/right part within a record is separated by commas and each record is terminated by ",;". These left/right parts are defined in Table 6-3, and are the same as those available for DJDEs on the input tape. The command file DJDEs do not require an IDEN statement in the JDE (nor an IDEN field on the command record) as does the input tape DJDE.

Examples of DJDE command files are illustrated in Figures 6-4 and 6-5. Figure 6-4 illustrates a simple DJDE command file. Each record of the file begins in the first byte and records are terminated by a ",;". To use this file it must be cataloged in the CMD file directory under a user specified name.

Figure 6-5 illustrates a feature of the 9700 DJDE commands that allows the JDL and JDE to be referenced within the file along with the specification of other DJDE commands. It is possible with this type of command file to have all the information within it necessary to start a job. When the operator initiates the print job he will only have to name this file on the START command. No other parameters need be entered. In this example the Job Descriptor Library "HON26" would exist on disk cataloged in the JDL file directory.

```
C
C     SAMPLE DJDE COMMAND FILE
C
COPIES = 5,COLLATE = YES,;
FORMS = FRMX,FORMAT = PDEX,;
FONTINDEX = 1,;
DATA = (2,132),;
END;
```

Figure 6-4. Sample DJDE Command File

```
C     DJDE COMMANDS WITH REFERENCE
C     TO JDL FILE ON DISK
C
JDL = HON26, JDE = H2,;
FORMS = CHART1,;
COPIES = 2,;
END;
```

Figure 6-5. JDL/JDE Specified in DJDE Command File

Initiating a DJDE

When a print job is to be run, the DJDE file is activated by the operator via the START command on the keyboard display. The syntax and explanation of the START command parameters are discussed in Chapter 7 under "Starting A Job". The command file is referenced on the START command by the "*filename" option which is the first parameter that can be specified. The asterisk ("*") preceding the file name denotes it as a DJDE command file.

Example 1:

```
START *MODSYS,JDE10,JDLDOS          (Operator Key-in)
```

where

MODSYS is a file of DJDE commands in the CMD file directory as in Figure 6-4.

JDE10 is the identifier of the Job Descriptor Entry which is part of the Job Descriptor Library "JDLDOS".

JDLDOS is the file name of the Job Descriptor Library in the JDL file directory.

Example 2:

```
START *HONWEL       (Operator Key-in)
```

where

HONWEL is a file name (in the CMD directory) which consists of DJDE commands.

No other parameters are necessary because the file "HONWEL" contains DJDE commands which request a specific JDL and JDE as in Figure 6-5.

Table 6-3. DJDE Left-Right Parts

Left Part	Right Parts	Description
{JDL / SYSTEM} =	jdl-id,	The right part "jdl-id" specifies the name of the Job Description Library to be invoked at the next report boundary. It must exist on disk in the JDL directory.
{JDE / JOB} =	jde-id,	The right part "jde-id" specifies the Job Descriptor Entry to be used within the selected Job Descriptor Library at the next report boundary.
COPIES =	value,	Specifies the number of copies of a report to produce. In collated mode this will take effect at the next report boundary, in uncollated mode at the next page boundary. The right part "value" is the same as defined in OUTPUT COPIES command of Chapter 5.
FORMS =	{form id / (form-id,init [,copies]) / NONE},	Specifies the form to be merged on the printed pages. It takes effect on the next page boundary. The form specified will be invoked beginning with the first page of the specified starting copy number. Multiple FORMS commands may be used to associate different forms with different copy plies. FORMS right parts are the same as defined in OUTPUT FORMS command of Chapter 5.
COLLATE =	{YES / NO} ,	Specifies whether collated or uncollated mode is to be used. It takes effect at the next report boundary. The right part "YES" specifies that report copies are to be printed in collated mode. "NO" specifies that report copies are to be printed in uncollated mode.
MODIFY =	{cme-id / (cme-id,init [,copies]) / NONE},	Specifies the Copy Modification (CME) to be used in report processing. It takes effect at the next page boundary. "cme-id" refers to a file which is separately cataloged in the CME library. See section "Copy Modification Features" in Chapter 5 for further details. The right parts of the DJDE MODIFY command are defined in the OUTPUT MODIFY description of Chapter 5.
FORMAT =	pde-id	Specifies the Page Descriptor Entry (PDE) to be used for formatting control. It takes effect on the next page boundary. "pde-id" refers to a separately cataloged file in the PDE library on disk. Details of the PDE command and PDE files are discussed in Chapter 5.

Table 6-3. DJDE Left-Right Parts (Continued)

Left Part	Right Parts	Description
NUMBER =	{ (pnum,lnum,cnum) / NO } ,	This command specifies page numbering control. It takes effect on the next report boundary. The right parts are the same as those for OUTPUT NUMBER command as defined in Chapter 5.
MARGIN =	{ value / (value, { IN / CM / POS }) } ,	This command specifies the left printing margin within each logical page. It takes effect on the next page boundary. The right part "value" is the same as defined in the LINE MARGIN command of Chapter 5.
DATA =	(pdo, length),	Specifies the location and length of printable data within an input record. It takes effect on the next page boundary. The right part parameters are defined in the LINE DATA command in Chapter 5.
OVERPRINT =	({ PRINT / MERGE / IGNORE } , { DISP / NODISP }) ,	Command for overprint control. It takes effect on the next page boundary. The right part parameters are the same as for the LINE OVERPRINT command in Chapter 5.
ASSIGN =	{ (channo,lineno) / (channo,(lineno,...,lineno)) } ,	This command specifies an assignment of a VFU channel number to a page line number. It is possible to have multiple ASSIGN commands within a DJDE. They take effect on the next page boundary. The right part parameters are defined the same as for the VFU ASSIGN command in Chapter 5.
TOF =	value,	This command specifies the "top-of-form" line number, It takes effect on the next page boundary. Command is defined same as VFU TOF in Chapter 5.
BOF =	value,	This command specifies the "bottom-of-form" line number, It takes effect on the next page boundary. Command is defined same as VFU BOF in Chapter 5.
FONTINDEX =	{ offset / NONE } ,	This command allows the user to specify in a user record a specific font to be used. It takes effect on the next report boundary. The right part parameters are same as LINE FONTINDEX command in Chapter 5.
ITEXT =	{ sc / (sc, passnum) } ,	The ITEXT, OTEXT, RTEXT, and RFORM commands have the same syntax and parameter definitions as defined in Chapter 5. These commands become effective at the next report boundary.
OTEXT =	{ sc / (sc [,passnum] [,WAIT]) } ,	
RTEXT =	{ sc / (sc ,passnum [,line [,col]]) } ,	
RFORM =	form-id,	

Table 6-3. DJDE Left-Right Parts (Continued)

Left Part	Right Parts	Description
C␣	sc	The "C" command allows for comment text to be included in the DJDE record. The "C" must be followed by a space and not by an equal sign. All text up to a semicolon or end-of-record will be treated as commentary.
END␣		This command is used to indicate the end of DJDE information. When END command is encountered, the system will apply all DJDE information specified to the current printing environment at the next report/page boundary. "END" must terminate with a semicolon/blank character pair.

7. SYSTEM OPERATIONS

Introduction

Control is exercised over the 9700 system through the use of the keyboard/display unit. It is the primary means of communication between the operator and the Operating System Software. Through the keyboard/display, the operator can control the loading of the operating system; initiate jobs; build forms; modify source files; request sample prints of data, forms, fonts, and logos; catalog forms, fonts, logos, JDLs, PDEs and CMEs; and obtain accounting summaries and other system outputs. While print jobs are running, messages are provided to the operator concerning the status of the jobs, and if necessary, of unexpected conditions.

A summary of commands available to the operator is shown in Table 7-1. Details of these commands are defined in the following sections of this chapter and in the 9700 Operators Guide (Xerox Publication No. 600P81096).

Table 7-1. System Command Summary

Command	Function	Ref. Page
ABORT	Abort print job or discontinue system activities	7-13
ACCOUNT	Accounting file maintenance	7-19
ALIGN	Align laser image with paper	7-5
CONTINUE	Resume input/output activities	7-12
COPY	Copy files from disk/tape to disk	7-25
DELETE	Delete files from system disk	7-27
FILE	List file directories on keyboard/display	7-27
FEED	Select active paper tray	7-22
FONTS	Set maximum number of active fonts per print job	7-28
FORMS	Set maximum number of active forms per print job	7-29
JOBS	Display job status information	7-15
LIST	Hard copy listing of file directories	7-27
LOGON	Create access privilages	7-6
MOVE	Position tape by blocks or files	7-21
REPORT	Report of system activity and accounting	7-19
RESET	Forces all system activities to cease	7-14
REWIND	Rewind tape	7-21
SAMPLE	Sample and test pattern print	7-17
SELECT	Select active output bin	7-23
SPACE	Position tape by logical reports or pages	7-20
START	Initiate print job	7-8
STOP	Suspend input/output activities	7-11
task initiation	Initiate a task	7-18
UNLOAD	Lower inactive paper bin	7-23

Key-in Conventions

The following conventions are to be observed when communicating with the 9700 via the keyboard/display unit.

- The system is always ready to accept input
- All commands may be abbreviated to the first three characters
- Commands are terminated by striking RETURN key except those entered via the function keys (described below)
- Error and information messages from the system will be preceded by "OS" and followed by a four digit code (the severity level). The complete set of system messages is contained in Appendix A. PDL and FDL messages to the keyboard/display are contained in Appendix B.
- A message acknowledging an operator's request will be displayed as positive feedback to the operator. These messages are underlined in examples to differentiate them from user keyins
- The error message OS2710 INVALID COMMAND RE-ENTER is displayed when an erroneous command or keyword within a command is entered
- There are six special keys on the keyboard/display unit called function keys. They need only be pressed to invoke a specified action (i.e., RETURN key is not pressed). These function keys are:

(CONTINUE)	Using this function key is like entering the CONTINUE command. (See "Continuation After Stopping Printing")
(DELETE)	Pushing this key while holding down the control key (CTRL) will erase the entire current line from the screen.
(RUBOUT)	This key is used to erase characters. Each time it is pushed, it erases the previous character on the current line and backspaces one position.
(SAMPLE COPY)	Pushing this function key produces a sample page in the Sample Print Tray during printing, just like the SAMPLE command does. (See "Sample Print")
(STOP)	Pushing this key is the same as entering the STOP command. (See "Stopping the System")
(STATUS)	This key can be used instead of the JOBS command to get jobs status. (See "Status Information")

System Start-Up

The operator must turn the main POWER ON switch on the right side of the Control Module to power up the 9700. The POWER and WARM lights will light and the system will begin its warm up cycle (10 15 minutes). When the warm up cycle is completed the WARM light will go out and the READY light will come on. On the keyboard/display the following message will be displayed:

READY
$

Once the prompt symbol $ has been displayed, the operator can boot the system from disk by typing in the appropriate command on the keyboard. The boot command has the following form:

B

After the boot command has been entered, diagnostics will be run to test the memory, the processors, and tape and disk controllers. Any errors discovered will be reported. If they are beyond the capability of the operator to repair, a Xerox field engineer must be called. If there are no errors, the system is booted and it responds with the following messages:

XEROX 9700
ELECTRONIC PRINTING SYSTEM
VERSION X REVISION Y
ENTER DATE (MM/DD/YY): (Operator enters date)
ENTER TIME (HH:MM/SS): (Operator enters time)
(System will print time and date)
ARE DATE AND TIME CORRECT AS DISPLAYED (Y/N)?

If N is entered the above time and date requests will be made again. If Y is entered, the system will respond with the message:

OS1000 READY FOR COMMANDS

The system is now ready to accept operator commands.

Setting Physical Page Alignment

ALIGN

The ALIGN command is used to align the laser image with the paper. Every 9700 machine will require slightly different alignment values, but once determined, these values remain relatively constant for a given machine. Alignment values entered will be preserved by the system until changed. The command form is:

ALIGN [scans][,dots]

where

scans — is a positive decimal integer in scan line units indicating an offset perpendicular to the scan line direction. This offset is measured from the start of imaging to the first scan line coinciding with the leading edge of the paper.

dots — is a positive decimal integer in dot units indicating an offset parallel to the scan line direction. This offset is measured from the start of a scan line to the first dot position coinciding with the edge of the paper.

If no operands are specified, then the current alignment values are displayed.

Examples:

```
OS1000 READY FOR COMMANDS
ALIGN   125, 152
ALIGNMENT IS 125 SCAN LINES AND 152 DOTS
ALIGN
ALIGNMENT IS 125 SCAN LINES AND 152 DOTS
ALIGN   ,160
ALIGNMENT IS 125 SCAN LINES AND 160 DOTS
```

File Access Protection

LOGON

The LOGON command provides an installation with a way to authorize/restrict access to files of a given type (i.e., those cataloged in a certain file directory) for a particular action (record editing, deleting and copying of files). This command allows three user "classes" which provide varying degrees of access to files as illustrated in Figure 7-2. Passwords are required with the use of the two "higher" levels of access.

The command form is:

$$\text{LOGON}\ \left[\left\{\begin{matrix}1\\2\text{,password}\\3\text{,password}\end{matrix}\right\}\right]$$

where

1,2, or 3 specify the user classification (1 = lowest, 3 = highest). Table 7-1 illustrates how this classification is related to the functions that can be performed.

password is a 1-15 character string that may consist of any characters (including leading and trailing blanks) on the keyboard. Passwords are provided with a 9700 system which may be modified at each installation.

When on class level 2 or 3, the password may be displayed by typing LOGON with no parameters. After displaying the password, the user also has the opportunity to change it. The initial password for level 2 after building a new system is null (i.e., logon on the system with LOGON 2,(RETURN)).

Table 7-2. File Access Classes

FUNCTION	RECORD EDITING			FILE DELETE			FILE COPY		
TYPE↓ CLASS→	1	2	3	1	2	3	1	2	3
CMD	N	Y	Y	N	Y	Y	Y	Y	Y
CME	N	N	N	N	Y	Y	Y	Y	Y
FNT	N	N	N	N	Y	Y	Y	Y	Y
FRM	N	N	N	N	Y	Y	Y	Y	Y
FSL	N	Y	Y	N	Y	Y	Y	Y	Y
JDL	N	N	N	N	Y	Y	Y	Y	Y
JSL	N	Y	Y	N	Y	Y	Y	Y	Y
LGO	N	N	N	N	Y	Y	Y	Y	Y
PDE	N	N	N	N	Y	Y	Y	Y	Y
SYS	N	N	Y	N	N	Y	N	N	Y
TMP	N	Y	Y	N	Y	Y	N	Y	Y
TSK	N	N	N	N	N	N	N	N	Y

TYPE — is the name of the file directory in which files are cataloged. Each file directory will contain a specific type of file (i.e., type CME contains files with DJDE commands. The directory names are defined in "Disk File Manipulation" of this chapter.

{N / Y} — means NO or YES as to whether a particular function (record editing, deleting, copying) may be performed.

RECORD EDITING — means that records in a file may (or may not) be edited. If access is no (N) then a GET command will not be accepted nor any record editing commands.

FILE DELETE (or COPY) — means a file may (or may not) be deleted (or copied) either from an EDITOR command or with the OSS level DELETE (COPY) command.

Job Control

The operating system provides commands which enable the operator to control the flow of print jobs through the printing system. These commands are entered through the keyboard/display unit.

Starting a Job

START

The START command is used to initiate the printing of a user's print tape. The command form is:

$$\text{START } [*\text{filename},]\left[[\text{jde}]\left[,[\text{jdl}]\left[,\begin{Bmatrix}\text{S}\\\text{M}\end{Bmatrix}\ [,\text{copies}\ [,\text{REPORTS:r1,r2}\ ,\ldots]]\right]\right]\right]$$

Each parameter of the START command is positional and must be separated by commas. With the exception of the *filename option, a comma must be entered to replace a parameter that is not specified. Options specified on the START command will override those specified in the job descriptor library and options specified in the DJDE command file will override those specified in the job descriptor library. Further details on override parameters are contained in "Hierarchy of Replacement" in Chapter 3. Examples of START command usage are provided after the parameters are defined.

Options on the START command are as follows:

filename — is a 1-6 character identifier for a file on disk consisting of DJDE commands. It must be cataloged in the CMD directory. These DJDE files are discussed in Chapter 6 under "DJDE Command Files". Note that filename is preceded by an "*".

jde — is a 1-6 character identifier for the Job Descriptor Entry to be used in processing the job. If the "jde" option is not specified then a default one will be selected from the Job Descriptor Library specified as the "jdl" option. The default "jde" name is DFLT (see "Default JDE" in Chapter 3).

jdl — is a 1-6 character identifier of a disk file which contains the Job Descriptor Library for the print job. It must be cataloged in the JDL file directory. If the "jdl" option is not specified then the default one is used. The default "jdl" is a file named DFAULT which is user created and cataloged in the JDL directory.

$\begin{Bmatrix}\text{S}\\\text{M}\end{Bmatrix}$ — are parameters used to control the processing mode of a job. The two modes are: Multi-report (M) and Single-report (S). The default mode is Multi-report. Single-report mode halts the system after the processing of each report to allow the operator to select the job set-up parameters for the next report. Multi-report mode allows all reports in all files to be processed continuously. Input processing will automatically sequence from report to report, file to file, volume to volume until all reports have been processed.

copies — specifies the number of copies of a report to be printed. It is an override of the "copies" value specified in the job descriptor entry (also overrides a value for "copies" if specified in a DJDE command file).

REPORTS:r1,r2...

The keyword REPORTS allows the user running in Multi-report mode to specify the sequence and set of reports to be processed. For "r1,r2..." the user specifies numeric values or ranges of values, representing the order of reports to be printed. A range is specified as n -m , where n and m are the first and last reports in the range to be processed, respectively. For example, entering "REPORTS:6,1-3,5,4" would cause the sixth report to be printed first, followed by the first through third, followed by the fifth and then the fourth. The REPORTS option is also useful if only one report is to be printed of a multiple report tape. This would save the step of spacing (see SPACE command) over reports not needed. A maximum of 15 values (or ranges of values) is allowed.

Example 1:

```
START J12,H2SYS
```

This command starts a print job using the H2SYS job descriptor library and the job descriptor entry, J12. It will run in multi report mode (by default) and print the number of copies as specified in the J12 job descriptor entry. The job descriptor library, H2SYS, must reside in object form in the JDL directory. After the START command is initiated, several messages will be displayed on the keyboard/display to inform the operator of the print jobs progress. In most instances, only one more keyin will be required of the operator. An example of the interaction is as follows:

```
OS1000 READY FOR COMMANDS
START J12,H2SYS
OS1010 STARTING JOB 00003
OS2010 MOUNT INPUT TAPE; "CONTINUE I" WHEN READY
CONTINUE I
OS0010 RESUMING INPUT
OS0020 RESUMING OUTPUT
OS1020 JOB 00003 HAS COMPLETED INPUT PHASE
OS1030 JOB 00003 HAS COMPLETED PRINTING
OS1000 READY FOR COMMANDS
```

Example 2:

```
START J12,H2SYS,,5
```

This command is the same as in Example 1 with the exception that five copies are requested. The value of 5 entered for copies will override the value specified in the J12 job descriptor entry. Note that a comma replaces the unspecified mode option.

Example 3:

```
START
```

No options are specified so all defaults take effect. The default for the job descriptor library will be the user file named DFAULT which must be in object form and cataloged in the JDL directory. The default job descriptor entry is made up of the SYSTEM level commands specified in the job descriptor library DFAULT.

Example 4:

```
START *UNDJDE,,UNJOB,S,4
```

In this example, the DJDE command file UNDJDE (cataloged in CMD directory) is initiated along with the job descriptor library UNJOB (cataloged in the JDL directory). The job descriptor entry to be used is the one defined by the SYSTEM level commands in the UNJOB job descriptor library (DFLT). The job will be run in single report mode and 4 copies will be printed.

Example 5:

```
START *USDJDE
```

In this example only the DJDE command file USDJDE is initiated. This file (cataloged in the CMD directory) will contain DJDE records including a specification for the job descriptor library and job descriptor entry (if not specified, the default jdl/jde would be used).

Stopping the System

STOP

The STOP command is used to suspend tape input and/or printer output activities. Since at any given time, the system may be processing input for one job and output for another job, input and output may be controlled separately. Stopping input suspends tape processing but allows printing of the page backlog to continue. Stopping output suspends printing after cycling out the paper path but allows processing of tape input to continue. The command form is:

STOP{I / O} or (STOP) function key

where

I requests that input activities be suspended until resumed by CONTINUE command.

O requests that output activities be suspended until resumed by a CONTINUE command.

If no operand is specified or the (STOP) function key is used, both input and output activities are suspended.

Examples:

```
STOP I
OS0510 INPUT STOPPED

STOP
OS0510 INPUT STOPPED
OS0500 OUTPUT STOPPED
```

Continuation after Stopping Printing

CONTINUE

The CONTINUE command is used to resume tape input and/or printer output activities suspended by the STOP command or soft-stop mid job errors. As with the STOP command, input and output activities may be controlled separately. Continuing input allows tape input to be spooled to disk even if printer output is stopped. Continuing output allows an existing page to be printed even if input is stopped. The command form is:

CONTINUE{I / O} or (CONTINUE) function key

where

I requests that input activities be resumed

O requests that output activities be resumed

If no operand is specified or the (CONTINUE) function key is used, both input and output activities are resumed.

Examples:

```
CONTINUE I
OS0020 RESUMING INPUT

CONTINUE
OS0020 RESUMING INPUT
OS0010 RESUMING OUTPUT
```

Continuation After MOVE or SPACE

After a MOVE or SPACE command has been issued the operator has the option (after the operation completes) of entering a CONTINUE command (as described above) or one of the form:

```
CONTINUE   jde, jdl
```

where jde and jdl are defined the same as on the START command.

This form of the CONTINUE command is to be used when the jdl and jde parameters to be used for the next report to be processed from tape are different than those issued on the last START command.

Aborting a Job

ABORT

The ABORT command removes a print job from the system or discontinues certain system activities. A job abort is handled in this manner: if the job is in the input phase, tape processing ceases with the next tape read; if the job is queued for printing, it is removed from the queue; and if the job is printing, it is truncated at the abort point and the system continues with any jobs remaining in the print queue. Other system activities may also be aborted. These include the language processors (PDL and FDL), tasks evoked by name, and commands as noted in the individual command descriptions. The command form is:

ABORT [job-id]

where

job-id is the job identification assigned to a job in response to the START command

If no job-id is specified, any non-print task will be aborted.

Examples:

```
ABORT 5
OS0900 JOB 5 ABORTED

ABORT
OS0950 TASK ABORTED
```

Resetting the System

RESET

The RESET command forces all system processing activity to cease. All print jobs are removed from the job queue and any pages in the printer paper path are cycled to the stacker. Accounting information on disk is updated to reflect any processing or partial printing done. If the input device is tape, it is left positioned at the point at which it was when the RESET command was given. The system remains in an idle state until the next command. The command form is:

RESET

Example:

```
RESET
OS0990 RESETTING THE SYSTEM
```

Status Information

JOBS

The JOBS command is used to display system status and job status information. System status shows the state of the system (running, idle, stopped), and job status shows information about all jobs known to the system. The command form is:

JOBS or (STATUS) function key

In response to this command, the system outputs a formatted display:

system-status

JOB-ID	JDE-USED	STATUS	COPIES	PAGES	MODE
id	jde	s	c	p	m

where

system-status is one of the system states defined below:

RUNNING	if either input or output processing is active
IDLE	if the print job queue is empty
STOPPED	if both input and output activities have been suspended

id is the job identification supplied by the system in response to a START command

jde is the JDE name used to start the job

s is the job status which may be:

QUEUED	if the job is waiting to be processed
INPUT	if the job is in the input phase and has not begun to print
OQUEUED	if the job is waiting to print
PRINTING	if the job is being printed
ABORT	if the job is to be aborted

c is the total copies to be printed if job status is INPUT. If job status is QUEUED, c is either the copies value specified by operator on START command or blank for default value specified by the JDE. Otherwise, c is the remaining copies to be printed (if mode is COLLATE) or total number of copies to be printed (if mode is NON-COLLATE).

p if printing and mode is COLLATE, zero if on first pass; otherwise, the remaining page backlog on the system disk for current copy; or if printing and mode is NON COLLATE, the remaining page backlog on the system disk for the current page (if mode is NON-COLLATE).

m is either C (COLLATE mode) or NC (NON-COLLATE mode)

Example:

In the following example, five complete copies and 120 pages of the current copy remain to be printined of the job with "id" of 23. Job 24 is in the input phase, so 15 represents the total number of copies to be printed. The 9700 does not know the number of pages or the mode until the job has completed input. The system status is RUNNING, since there are jobs in the print queue and Input and Output have not been stopped.

```
RUNNING

JOB ID      JDE-USED   STATUS     COPIES     PAGES     MODE
00023       JOB10      PRINTING   5          120       C
00024       JOB6       INPUT      15
END OF JOBS STATUS
```

Sample Print

SAMPLE

The operator may request a sample print either during job printing or when the system is in the idle state. In the former case, the next page to be printed will be printed twice, with one copy being delivered to the output bin normally and the other being delivered to the sample print tray. In the case where the system is in the idle state at the time the request is given, the operator must indicate the item to be sample-printed (i.e., the operator may request a sample print of a font, form, or a logo).

Single Sample Print

To request a sample print during job printing, the operator enters the command:

SAMPLE or (SAMPLE COPY) function key

To request a sample print of a font, form, or logo, the operator enters a request of the form:

SAMPLE file-name. file-type

where

file-name specifies the name of a file to be sample-printed

file-type specifies the type of the file to be sample-printed. If only ".type" is entered, then all files of that type will be sample printed. "type" may be any of the following:

- FRM if the file is a form cataloged in the FRM library
- FNT if the file is a font cataloged in the FNT library
- LGO if the file is a logo cataloged in the LGO library

Continuous Sample Print

To request the continuous printing (up to 32,767 copies) the operator enters the command:

SAMPLE file-name.TST

where

file-name indicates the name of a form file cataloged in the FRM library

TST specifies the file is to be printed continuously

The specified form will be printed continuously until printing is halted by giving the STOP command or the ABORT command.

Starting a Task

Certain tasks supplied with the system are invoked by task name. These tasks can be run only when the system is in the idle state, i.e., when the print job queue is empty. Currently included in this category are the EDITOR, FDL, PDL and OSDS. The command form is:

```
taskname   parm(1),parm(2),...
```

where

taskname	is a 1-6 character identifier for the task
parm(i)	are task specific parameters supplied with the command

Tasks invoked by name may be aborted via the ABORT command. The task name need not be given since no other jobs or tasks will be active.

System Activity Statistics

REPORT

The operator may obtain a display or printed report of system activity and machine usage information by the following command entered via the keyboard/display unit:

```
REPORT ACTIVITY [,CLEAR]
```

If "CLEAR" is specified, system activity information is deleted from the activity log as the report is generated. Before processing of the report begins, the operator is asked whether report information is to be displayed with the following message:

```
OS2150 DISPLAY (Y/N)?
```

If "Y" is entered, the report is displayed on the keyboard/display unit. Any other response causes the report to be printed with output delivered to the sample print tray.

Customer Usage Statistics

REPORT

The customer usage statistics summary report is obtained by entering the following command via the keyboard display unit.

```
REPORT USER [,CLEAR]
```

The report is output to the bin. If "CLEAR" is specified, the accounting log is cleared as the report is generated.

Accounting File Maintenance

ACCOUNT

The Print Descriptor Language command ACCT (see "Job Accounting" in Chapter 5) allows the user to set-up department codes (DEPT = sc) under which accounting information will be maintained. In addition to the PDL command, the OSS level command ACCOUNT must also be invoked by the user. The format of the command is:

```
ACCOUNT  { ADD,department    }
         { DELETE,department }
         { LIST              }
```

where

ADD — specifies a department name is to be added to the list of departments under which accounting information will be maintained.

department — is a string of alphanumerics (A thru Z, 0 thru 9 and the character ":") specifying the name of the department.

DELETE — specifies a department name is to be deleted from the list of department names. A department name cannot be deleted if data for that department exists in the file.

LIST — lists all the department names in the current list on the keyboard/display.

Logical Report Spacing

SPACE

If single-report mode has been selected on the START command, the operator may position a tape by logical report units. After each report has been processed, the system will inform the operator with the message.

```
OS1020 JOB job-id HAS COMPLETED INPUT PHASE
OS1000 READY FOR COMMANDS
```

The operator may then enter another START command to process the next report on the tape or position to another report on the tape by issuing a SPACE command. The SPACE command has the following form:

```
SPACE  n  REPORTS
```

where

n — is a decimal integer in the range -100 to +32767. A positive number indicates forward report spacing, and a negative, backward spacing. The tape cannot be positioned beyond physical BOT and EOT limits. The default is the forward space over one report

REPORTS — indicates positioning by logical report units.

Page Spacing

During the printing of a report, the SPACE command may be used to position forward and backward by pages within the current report. Prior to issuance of the spacing function, printing must be stopped with the STOP or STOP O command. The SPACE command is then issued and printing is resumed with the CONTINUE command. The command form is:

```
SPACE  n  PAGES
```

where

n — is a decimal integer in the range -100 to +32767. A positive n results in forward spacing over n pages unless end-of-report terminates the spacing. A negative n results in backward spacing over n pages unless beginning-of-report terminates the spacing. Backward spacing is limited to 100 pages for system disk space considerations. The default for n is 1.

PAGES — indicates positioning by page units.

Example:

```
STOP O
OS0500 OUTPUT STOPPED
SPACE 200 PAGES
OS0610 PAGE SPACING FORWARD
CONTINUE O
OS0020 RESUMING OUTPUT
```

Block/File Positioning

MOVE

The MOVE command is used to physically position a magnetic tape a specified number of files or blocks either forward or backward. If the tape unit is actively being manipulated by a system task when this command is issued, that task controls whether and how the tape is positioned as specified subject to physical BOT and EOT limits. The command form is:

```
MOVE  n {FILES  }
        {BLOCKS }
```

where

n — is a decimal integer in the range -32768 to +32767. A positive number indicates forward movement and negative backward movement. The default is 1.

FILES — indicates positioning by files. A positive "n" moves the tape forward over "n" tape marks unless end of data terminates the tape movement. A negative "n" moves the tape backward over "n" tape marks unless BOT terminates the tape movement. File positioning is the default.

BLOCKS — indicates positioning by blocks. A positive "n" moves the tape forward over "n" interblock Tape Gaps unless EOT or a tape mark terminates tape movement. A negative "n" moves the tape backward over "n" blocks unless BOT or a tape mark terminates tape movement.

Rewinding a Tape

REWIND

The REWIND command is issued by the operator to rewind the current tape on the tape drive. It should be issued after the user has completed processing reports with a tape and is going to mount another. REWIND must be used if a print job is being processed in single report mode ("S" option on the START command) and end-of-data on the tape has not been reached. The command form is:

```
REWIND
```

Printer Paper Control

Paper Tray Selection

FEED

The operator may select or allow the system to automatically select the active paper tray with the FEED command. Manual tray selection allows the selected tray to be emptied and the printer cycled down. Automatic tray selection allows the main tray to be fed until a "paper low" condition is sensed; then, an automatic switch to the aux tray by the operating system allows printing to continue while the main tray is refilled. The command form is:

FEED { MAIN / AUX / AUTO }

where

MAIN or AUX — selects the MAIN or AUX tray, respectively. If the selected tray is not ready, the printer is cycled down and an appropriate message is issued. While operating, the printer is cycled-down whenever the selected tray is emptied.

AUTO — selects the automatic tray selection mode. The operating system will treat the MAIN tray preferentially. The MAIN tray will be fed initially until emptied, then the AUX tray will be selected until either the MAIN tray becomes ready again or the AUX tray is emptied.

After system boot, the MAIN option is implicitly in effect. If no operand is specified and manual tray selection is in effect, the default is selection of the currently inactive tray. If no operand is specified and AUTO is in effect, then the command is ignored. This command form offers a convenient means of switching trays in manual mode. However, if the inactive tray is not ready, the active tray will remain selected.

Bin Selection

SELECT

The operator may select, or allow the system to automatically select, the active output bin with the SELECT command. Manual bin selection allows the selected bin to be filled; when full, the operating system will stop the printer. Automatic bin selection allows the active bin to be filled; when full, the operating system will automatically switch to the alternate bin thus allowing printing to continue. For both selection modes, approximately seven pages will be delivered to the previously active bin after a new bin selection is made. Thereafter, pages will be delivered to the new bin if it is ready. The command form is:

$$\text{SELECT} \quad \left\{ \begin{array}{l} 1[,\text{SAMPLE}] \\ 2[,\text{SAMPLE}] \\ \text{AUTO} \end{array} \right\}$$

where

1 or 2 — selects bin 1 or 2, respectively. If the selected bin is not ready, the printer is cycled down and an appropriate message is issued. Whenever the selected bin is filled, the printer will be cycled down.

SAMPLE — selects the specified bin as the active bin to be filled and assigns the remaining bin as a logical sample print copy bin. If the selected bin is not ready the printer is cycled down and the message "OS2310 BIN n NOT READY" is issued.

AUTO — selects alternate bin automatically when active bin becomes full. Selects "active" bin if ready or automatically switches bins at initiation of printing.

If SELECT is entered with no operand the alternate bin is selected but does not change mode (i.e., if in AUTO it remains in AUTO, and if in manual it remains in manual).

Bin Lowering

UNLOAD

The operator may lower an inactive bin with the UNLOAD command. Full bins are lowered automatically by the operating system if they are not actively receiving paper. If the operator wants to unload a partially-filled, active bin, he may do so by selecting (with the SELECT command) the opposite bin and issuing the UNLOAD command. When all paper has been removed from a bin and the bin tray has been returned to its "in" position, the system automatically raises the bin to the ready position. The command form is:

$$\text{UNLOAD} \quad \left\{ \begin{array}{l} 1 \\ 2 \end{array} \right\}$$

where

1 or 2 — causes bin 1 or 2, respectively, to be lowered if inactive. If no operand is specified, the default action is to unload the inactive bin.

Disk File Manipulation

The system maintains a catalog of disk resident files. Each of these files is classified by a type within the catalog. Examples of various file-types are job descriptor library source, forms source, fonts, and logos. Commands are provided to list the catalog by file type, delete files from the catalog, and copy files into the catalog. The term "file-id" used in the syntax of the commands refers to the form:

file-name.file-type

where

file-name is a 1-6 character file name

file-type is one of the file directories under which file-name is cataloged

The following file-type names are used in the command definitions to follow:

CMD	-	DJDE command file	LOG	-	Logo data
CME	-	Copy modification object language	PDE	-	Page descriptor object language
FNT	-	Font data	SYS	-	System control
FRM	-	Form object language	TMP	-	Temporary file
FSL	-	Form source language	TSK	-	System task image
JDL	-	Job Descriptor object language	TST	-	Sample print test
JSL	-	PDL job source language			

Copying a File

COPY

The operator may use the COPY command to create new files on the system disk. A file to be copied may be input from tape or disk. Tape files are identified to the system by name and type or by the current position of the magnetic tape. After a copy operation is complete the tape is not rewound. A disk file is always identified by name and type. The command forms are:

Disk File to Disk File

COPY input-file-id output-file-id

where

file-id specifies the input or output file which is of the form "file-name.file-type" as defined previously

Example:

```
COPY FORMA.FSL  FORMX.FSL
CREATING FILE FORMX.FSL
```

Unlabeled Tape File to Disk File

COPY TAPE output-file-id

where

TAPE is a keyword specifying the input file will come from tape

file-id specifies the output file which is of the form "file-name.file-type" as defined previously

For this command, the input tape must be unlabeled, unblocked, EBCDIC, 9 track and 1600 bpi. The maximum record length is 133 character although only the first 72 characters of each record will be entered into the disk file. The largest file that can be created is 5000 records.

Example:

```
COPY TAPE UNSYS. JSL
```

Will copy a file from an unblocked, unlabeled tape to disk. The file will be named UNSYS and cataloged in the JSL file directory.

9700 Labeled Tape to Disk

```
COPY TAPE  { LABEL output-file-id }
           { NEXT                 }
           { ALL                  }
```

where

TAPE is a keyword specifying the input file will come from tape

LABEL specifies that the tape is a 9700 labeled tape

file-id specifies the output file which is of the form "file-name.file-type" as defined previously

NEXT specifies that the next file on the tape is to be copied to disk

ALL specifies all the files on tape are to be copied to disk

For this command the input tape must-be created by XEROX specifically for input to the 9700 (e.g., font and logo tapes).

Example:

```
COPY TAPE LABEL OCRA.FNT
SEARCHING FOR FILE OCRA.FNT
CREATING FILE OCRA.FNT
```

The above command will find and copy the file OCRA from the tape into the file named OCRA. It will be cataloged in the FNT file directory.

If another file existed on this tape (after the one just copied) then the command

```
COPY TAPE NEXT
```

will copy the next file from tape and store it in a "file-name.file-type" as specified in the tape label information.

The command

```
COPY TAPE ALL
```

will copy all the files from the 9700 labeled tape, store them in the appropriate "file-name" and catalog them in their "file-type".

Deleting a File

DELETE

The operator may delete a file from the system using the DELETE command. Deletion of a file causes its name to be removed from the system catalog and its file space on disk to be made available for reuse. The command form is:

DELETE file-id[,file-id],...]

where

file-id is of the form "file-name.file-type" as defined previously

Displaying a Catalog

FILE

The operator may display the disk file catalog on the keyboard/display using the FILE command. Specific file types within the catalog or the entire catalog may be displayed on the screen. The command form is:

FILE [file-type [,file-type],..]

where

file-type may be any one of the system's directories previously defined (i.e., CMD, CME, FNT, FRM, FSL, JDL, JSL, LGO, PDE, SYS, TMP, TSK and TST). If no file-type is specified, the default is to display all the files of each catalog.

Listing a Catalog

LIST

The operator may list the disk file catalog using the LIST command. Specific file types within the catalog or the entire catalog may be listed and delivered to the sample print tray. The command form is:

LIST [file-type [,file-type],...]

where

file-type is one of the file directories under which files are cataloged (i.e., CMD, CME, FNT, FRM, FSL, JDL, JSL, LGO, PDE, SYS, TMP, TSK, and TST).

Example:

LIST JSL, FSL, FRM	The system will print to the sample print tray all the files in the specified file directories (i.e., JSL, FSL, and FRM).
LIST	The system will list all the files in all the file directories

Establishing Print Job Characteristics

To optimize the allocation of controller memory, an installation may specify two parameters which characterize the jobs to be printed. These parameters define the maximum number of fonts and forms the system is to attempt to keep resident in memory during a print job.

Fonts

FONTS

The maximum number of active fonts is specified by the FONTS command. The command form is:

FONTS [number]

where

number — is a positive decimal integer greater than zero which specifies the maximum number of active fonts per print job. No single page may invoke more fonts than specified. This number must include all fonts used on the forms as well as variable data. The forms font, if a form is used, must be included in the derivation of "number". A larger "number" may be specified than will ever be used on a single page, thus allowing font changes on a page to page basis without throughput degradation if all the fonts can be loaded into memory. If "number" is omitted, the system responds with the value currently in effect.

The FONTS command remains in effect until the FONTS command is reissued.

Example:

OS1000 READY FOR COMMANDS
FONTS
OS1310 NUMBER OF ACTIVE FONTS IS 5

It is possible to use, during the course of a print job, more than "number" of fonts as long as no more than "number" of fonts is called for on a single page. If, during a print job, the current FONTS value is exceeded while processing a page (i.e., the total of fonts required for form printing plus those required for variable data printing), that job is aborted and the following message is displayed:

OS2880 MAXIMUM NUMBER OF FONTS ALLOWED EXCEEDED. ENTER FONTS COMMAND

The maximum number of active forms is specified by the FORMS command. The command form is:

FORMS [number]

where

number is a positive decimal non-zero integer that specifies the maximum number of active forms per print job. Specifying a "number" larger than 1 allows form changes on a page to page basis without throughput degradation once the forms have been loaded into memory. If "number" is omitted, the system responds with the value currently in effect.

The FORMS command remains in effect until the FORMS command is reissued.

Examples:

```
OS1000 READY FOR COMMANDS
FORMS
OS1300 NUMBER OF ACTIVE FORMS IS 1

OS1000 READY FOR COMMANDS
FORMS 3
OS1300 NUMBER OF ACTIVE FORMS IS 3
```

8. EDITOR

Introduction

The EDITOR is a 9700 system task which provides file editing facilities as part of normal operator communication and control. These facilitites are available to the user for the creation, modification and maintenance of files. The EDITOR has commands for creating and modifying source files (JDL, FDL, CME, PDE, DJDE source statements) as well as commands for the maintenance of any type of disk files.

The EDITOR allows the user to directly access each line (or record) of a source file. The user enters single line EDITOR commands, via the keyboard/display, which provide the following facilities:

- Creating a sequenced source file.
- Inserting, reordering, and replacing lines or groups of lines of text.
- Selective printing to the keyboard/display or sample print tray.
- Reordering and renumbering of records within a file.
- Merging part of one file into another.
- Context editing operation that allow matching, moving and substituting character strings within a specified range of text lines.
- File maintenance (allowing the user to copy and delete files of any type).

Each source file has line numbers associated with its data records. These line numbers are supplied by the EDITOR during file creation. Files entered via magnetic tape will have line numbers added to them at the time they are entered into the system. When a source file is being modified, the commands entered by the user are applied to a "working file", not to the permanent file on disk. At the time a request is made to edit an existing file, the specified file is copied into working storage. All editing operations are performed upon the working file and it is saved permanently on the disk only when specifically requested by the user.

Examples of commonly used EDITOR commands are illustrated in Figures 8-1, 8-2 and 8-3.

File Directories

A user may edit source files which are cataloged in the following "file-type" directories:

File Type	Contents
CMD	DJDE Command File source
FSL	Form source language
JSL	PDL job source language

When files are initially created they must be "saved" in one of the above directories. Non-source files are cataloged in the following directories:

File Name	Contents
CME	Copy modification object
FNT	Font data
FRM	Form object
JDL	Job descriptor object
LGO	Logo data
PDE	Page descriptor object
TMP	Temporary file
TSK	System task image
TST	Sample print test

Files within these directories may be accessed only by a subset of the EDITOR commands. These commands allow listing of the file type directories (LIST and FILE), deleting (DELETE) files or copying (COPY) files. LIST. FILE, DELETE and COPY are also available as system level commands (see "Disk File Manipulation" in Chapter 7).

Calling the Editor

To create a new source file the user requests the EDITOR facilities by typing EDIT on the 9700 keyboard/display. If a file already exists on disk and is to be modified, it can be edited by typing in:

EDI[T] [file-name.file-type](RETURN)

where "file-name" is a 1-6 character file name which exists on disk and "file-type" is one of the file directories (CMD, FSL, JSL) under which the "file-name" is cataloged.

Note that only the first three characters of the command need be entered. This is true of all EDITOR commands. The (RETURN) indicates the pressing of the RETURN key on the keyboard/display. All commands and data lines are ended by depressing the RETURN key.

After the user types in the request for the EDITOR processor, the EDITOR prompts the user for a command by typing "EDIT >" which indicates that another command may be entered.

Editor Commands

EDITOR commands fall into the following three categories:

- File commands: Commands that apply to an entire file. These commands may be given at any time.
- Record commands: Commands that act upon the record or a group of records within a file. These commands may be given only after a file has been selected for editing.
- Intra-record commands: Commands that make changes within an individual record. These commands generally manipulate character strings and may be given only after a specific set of records has been selected.

These commands and categories are summarized in Table 8-1. The syntax of each command is explained in the following section along with examples of usage.

Table 8-1. EDITOR Command Summary

FILE COMMANDS	FUNCTION
CLEAR	Clears the entire contents of the working file
COPY	Copies files from tape/disk to disk
CONVERT	Sets mode to convert
DELETE	Deletes a file from the system
EDIT	Begins an editing session
END	Ends an editing session
FILE	Lists disk file catalog on the operator's display
GET	Gets an existing file and copies it into working storage for editing
KEYS	Requests that the beginning and ending line numbers of the working file be displayed
LIST	Lists disk file catalog on the printer
MERGE	Merges an existing file into current working storage
NOCONVERT	Sets mode to no convert
SAVE	Saves the contents of working storage as a permanent file
RECORD COMMANDS	**FUNCTION**
DISPLAY	Displays source lines on the operator's console
DUPLICATE	Duplicates lines from the file into another area of the file
FIND	Finds and displays lines containing a specified text string
INSERT	Inserts lines using a specified line number into the file
MODIFY	Specifies a range of lines to be affected by intra-record commands
MOVE	Moves lines from one place to another in the file
PRINT	Produces hard copy of the working file to the sample print tray
REMOVE	Removes a specified group of lines from the file
RENUMBER	Renumbers the working file
REPLACE	Specifies that lines are to be removed and others inserted in their place
STEP	Implicitly specifies the modify range to be the next record
INTRA-RECORD COMMANDS	**FUNCTION**
D	Delete string
F	Insert following string
P	Insert prior to string
S	Substitute string

Editor Conventions

The syntax of EDITOR commands are defined in detail in the following section. The commands are in alphabetic order grouped according to their type (File, Record, Intra-record). Conventions used in defining the syntax are as follows:

- When typing in commands, only the first three characters need be entered. For example, typing in REN or RENUMBER will be interpreted as the same command by the EDITOR.

- All commands are terminated by pressing the RETURN key on the keyboard.

- The RUBOUT function key can be used to erase characters. Each time it is pressed, it erases the previous character on the current line and backspaces one position.

- Examples shown in this chapter will differentiate user typed commands from system supplied messages by underlining those printed by the system.

- The term "file-id" refers to the name of a file and the file directory in which it is cataloged. Thus, when specifying "file-id", the user must type "file-name.file-type". For example, SIGMA.JSL would be the file-id of a file named SIGMA which is cataloged in the JSL file directory.

- When "n-m" is used in the command syntax the following usage is implied:

  n-m = starting and ending line number for command
  n = only line number for this command
  n- = starting line number to end of file
  -m = beginning of file to ending line number

- The maximum line number for a file is 32,765.

File Commands

CLEAR

This command is used to erase all data lines from the working file. It has no parameter option and is used when a new file is to be created.

Example:

```
EDIT > CLEAR              Clear working storage
EDIT > INSERT 10,10       Issue next command
```

If a file exists in working storage (and has not yet been SAVEed) when the CLEAR command is entered, the EDITOR will respond with the following message:

```
THE FILE HAS NOT BEEN SAVED
DO YOU WANT TO SAVE IT (YES/NO)?
```

If the user types in NO the CLEAR command takes effect and the message

```
WORK FILE CLEARED
```

is printed on the console. If YES is typed the CLEAR command is ignored and the message COMMAND IGNORED is printed. The user may then save the file by typing in the SAVE command.

CONVERT

This command allows the user to edit files containing lower case characters using the upper case only keyboard/display. In order to replace lower case characters in a file, the lower case characters must be bracketed by the # symbol. The # symbol is used by EDITOR to indicate the start and the end of conversion to lower case characters. If the file does not actually contain lower case characters, the CONVERT command need not be used. After using the CONVERT command and text modification requiring it, the CONVERT mode should be "reset" with the NOCONVERT command (described below).

Example:

The CONVERT command would be needed if the text line "This is the end" (which was entered into a disk file from tape) needs to be altered. This text line could not be created (displayed) in lower case on the 9700 keyboard/display character because it is only possible to enter (display) upper case characters from (on) it.

The following command sequence illustrates the use of the CONVERT command to change the word "This" to "That".

```
  :
EDIT > CONVERT
EDIT > MOD 10
THIS IS THE END
EDIT > S/T#HIS/T#HAT/
THAT IS THE END
EDIT > NOCONVERT
```

The keyboard/display prints only in upper case but internally to 9700 the sentence is initially represented as "This is the end" and then as "That is the end".

After the use of CONVERT command is completed, the user should "reset" this mode by using NOCONVERT (described below).

COPY

This command allows the user to create new files on the disk. A file to be copied may be input from tape or disk. The parameters of this command are defined in "Copying a File" in Chapter 7.

DELETE

This command allows the user to delete a file from the system. The command form is:

DELETE file-id[,file-id] ,...]

where "file-id" is of the form "file-name.file-type" as defined previously.

Example:

```
DELETE UNIVAC.JSL,FORMX.FSL
UNIVAC.JSL DELETED
FORMX.FSL DELETED
```

EDIT

This command specifies that an editing session is to begin. It has the form:

EDIT [file-id]

If the "file-id" is specified then the requested file is brought into working storage.

Example:

```
OS1000 READY FOR COMMANDS
EDIT  SYSPDL.JSL
EDIT >
```

Additional commands then may be entered.

END

This command terminates an editing session. Control is returned to the system and no more EDITOR commands will be accepted until the EDITOR is requested again. The message OS1000 READY FOR COMMANDS will be displayed on the console after the END takes effect.

If the user types in END and a file (one just created or updated) has not been saved (SAVE command) the following message will be displayed on the console:

```
THE FILE HAS NOT BEEN SAVED
DO YOU WANT TO SAVE IT (YES/NO)?
```

If a YES is typed, the EDITOR ignores the previous END command and the message "COMMAND IGNORED" is displayed. The user may then type in the SAVE command. If the user types in NO, the EDITOR ends the editing session and retains the contents of working storage for a future session. CAUTION: There is only one workfile, thus any intervening use of the EDITOR is likely to destroy its contents.

FILE

This command allows the user to display on the keyboard/display the file in each disk file directory. The command form is:

```
FILE [file-type [,file-type],...]
```

where "file-type" may be any one of the system's directories previously defined (i.e., CMD, CME, FNT, FRM, FSL, JDL, JSL, LGO, PDE, SYS, TMP, TSK). If none is specified, the default is to list all the files of each catalog.

GET

This command specifies that an existing file is to be brought into working storage. It has the form:

```
GET  file-id
```

Example:

```
OS1000 READY FOR COMMANDS
EDIT
EDIT > GET  SYS2.JSL
```

If the GET command is typed and a file exists in working storage that has not been saved (SAVE command) then the message:

THE FILE HAS NOT BEEN SAVED
DO YOU WANT TO SAVE IT (YES/NO)?

is printed on the console. A YES response causes the EDITOR to ignore the GET command. The user can then issue the SAVE command. If the response is NO, the GET command is performed and the previous contents of working storage is lost.

KEYS

This command is used to request that the beginning and ending line numbers of the working file be displayed. This command has no optional parameters.

Example:

```
EDIT > GET SYS2.JSL
EDIT > KEYS
BEGINNING LINE NUMBER 000010
ENDING LINE NUMBER 000050
```

LIST

This command allows the user to list on the printer the files in each disk file directory. It has the form:

LIST [[file-type [,file-type], ...]]

where "file-type" is defined as one of the file directories under which files are cataloged (i.e., CMD, CME, FNT, FSL, JDL, JSL, LGO, PDE, SYS, TMP, TSK, and TST).

Examples:

```
EDIT > LIST JSL, FSL, FRM
----
  :
  :
----                    The system will list on the printer all the files in the specified file directories
----                    (i.e., JSL, FSL, and FRM).
----
```

```
EDIT > LIST             The system will list all the files in all the file directories.
```

MERGE

The purpose of this command is to add the contents of a disk file to working storage. The current records in working storage will not be destroyed when the records of the new file are brought in. The MERGE command has the form:

MERGE file-id [,n [,s]]

where n is the sequence number to be assigned to the first record of the file, and s is the sequence increment.

NOCONVERT

This command is a "reset" of the CONVERT command described previously. By "resetting" the CONVERT command no upper-to-lower case conversion will take place. This is the default mode of the EDITOR. The NOCONVERT command has no optional parameters.

SAVE

This command specifies that the contents of the working file are to be saved in permanent file storage. It has the form:

SAVE [file-id]

If no "file-id" is specified, the contents of the working file is saved under the name currently associated with the working file (as set by a previous GET command, a previous SAVE command, or as set when EDIT is called). Otherwise the operator is prompted for a "file-id" under which to save the contents of the working file.

DISPLAY

This command is used to obtain a listing of all or selected lines of the current working file on the display screen.

This command has the form:

```
DISPLAY    n-m
```

where n-m are as defined previously in section, "EDITOR Conventions".

Examples:

DISPLAY	Display all lines
DISPLAY 10-20	Display lines 10 through 20
DIS 5 -	Display lines 5 through the end of the file
DIS - 100	Display lines from the beginning of file through line 100

If the number of lines to be displayed does not exceed the screen display capacity, all requested lines are displayed on the screen. If the number of lines to be dispalyed exceeds the display screen capacity, the user will be prompted with the message:

ENTER RETURN TO CONTINUE

to indicate that all requested lines have not been displayed. To continue displaying another set of lines, the user depresses the RETURN key. If any other response is given, the display is halted.

DUPLICATE

This command specifies that a line or group of lines will be duplicated at another point in the file. It has the form:

```
DUPLICATE   n [- m] ,p [,s]
```

The specified lines are moved to a new location starting with line number p. The new line numbers will be incsemented by s. the default value for s is 1. The command is identical to the MOVE command except that original lines are not deleted from the file.

FIND

This command specifies that all lines in the specified range containing the given string are to be displayed. It has the form:

FIND [n [- m]]/string/

Examples:

FIND /JDE/	Display each line that has the characters "JDE" in it.
FIN 5-100/JOB/	Display each line from 5 to 100 that has the characters "JOB"
FIN 5/SYSTEM/	Display line 5 if it contains the word "SYSTEM"
FIN 5-/SYSTEM/	Display each line from 5 to end of file if it contains the word "SYSTEM"
FIN -100/JOB/	Display each line from 1 to 100 that has the characters "JOB"

INSERT

This command specifies that a line or group of lines are to be inserted into the file. It has the form:

INSERT n [,s]

Lines are inserted starting at n and incremented by s. Line n must not already exist. The operator will be prompted by the next sequence number and input terminates upon receiving a null line or encountering an existing record. The default value for s is 1.

Examples:

INSERT 10,10	Insert lines starting at 10 and increment by 10 for next line
INS 15	Insert lines starting at 15 and increment by 1

Another method for inserting lines into the working file is to just enter the desired line number, one blank, and then the text. For example, to enter text on line 55 the user would key-in:

EDIT > 55 ACCT=(USER,BIN);

and then press the RETURN key. This type of insertion will replace existing lines without notification.

MODIFY

This command specifies a range of lines which may be affected by subsequent string modification commands. These lines will be displayed on the console after the MODIFY command has been issued. String modification commands are defined below in section "Intra-Record Commands". This command has the form:

MODIFY n-m

Examples:

EDIT >	MODIFY 5-100	Prepare to modify lines 5 through 100
EDIT >	S/JDE/JOB/	Replace occurrences of "JDE" with "JOB" in lines 5 through 100
EDIT >	MOD 5	Prepare to modify line 5
EDIT >	D/HOST=OSWTR/	Delete text "HOST=OSWTR" from line 5.

If no match is found in the modify range, the message NO MATCH FOUND is displayed.

MOVE

This command specifies that a line or group of lines are to be moved to new line positions. It has the form:

MOVE n [-m] ,p [,s]

The specified lines (n-m) are moved to a new location starting with line number p. The new line numbers will be incremented by s. The default value for s is 1. Lines n-m will be removed (deleted) from the file.

PRINT

This command is used to obtain a hard copy listing delivered to the sample print tray of all or selected lines of the current working file. It has the form:

PRINT n-m

In order to print the file, the user must exit (END command) the EDITOR after the PRINT command is given. The current working file will be intact if the user re-enters the EDITOR after the file is printed.

REMOVE

This command specifies that a line or group of lines are to be removed (deleted) from the working file. It has the form:

REMOVE n-m

For this command, n is a required parameter; if there is a range, m is also required (i.e., hyphen can not be used to specify initial or terminal line).

Examples:

REMOVE 5-25	Remove lines 5 through 25
REM 15	Remove line 15

Another way to remove a line from the working file is to enter the line number (of the line to be removed) followed by the RETURN key. To delete a group of lines, the user enters the beginning and ending line numbers separated by a dash.

Examples:

```
EDIT > 55 (RETURN)          Delete line 55
     1 RECORD DELETED

EDIT > 10-40 (RETURN)       Delete lines 10 through 40
     4 RECORDS DELETED      Assuming line numbers are incremented by 10
```

RENUMBER

This command is used to renumber the lines of the working file. Renumbering begins with line number n and successive data lines are assigned numbers in increments of s. The default values for n and s are 10. This command has the form:

RENUMBER [n [,s]]

STEP

This command will display each record of a file, one at a time, starting with the next record as specified in the MODIFY command. It has the form:

STEP

Example:

```
EDIT > MODIFY 10            Set modify range
     ...line 10 is printed...
EDIT > STEP
     ...text of next line after line 10 is printed...
EDIT > S/MOST/HOST/         Change text in next line after line 10
     1 STRING CHANGED
EDIT > STEP
     ... next line is printed
EDIT > ...user may enter an appropriate intra-record command (i.e., S,F,P or D) or another STEP
     :
     :
```

Intra-Record Commands

The following commands allow the user to modify parts of a record (or records). These commands are used with the MODIFY command which specifies the record (or records) to be altered. After a line has been altered it will be displayed.

D

Delete a string from a record (or records) in the working file. The command has the form:

D [k] /string/

Where k indicates that the kth occurrence of "string" in each line will be affected (k=0 indicates that all occurrences of specified string are to be affected). When k is specified, it must be preceded by one blank character. The default value of k is 1. "string" is deleted from lines specified in the MODIFY command.

Examples:

EDIT > MODIFY 5	Modify line 5
EDIT > D/SYSTEM/	Delete the first occurrence of "SYSTEM" from line 5.
EDIT > MOD 5-70	Range of lines which may be altered is 5 through 70.
EDIT> D 0/JDE/	Delete all occurrences of "JDE" from each line, starting with line 5 and proceeding through line 70.

F

Insert following a string. This command has the form:

F [k] /string1/string2/

k has the same meaning as in D (delete string). "string2" is inserted following "string1" for lines specified in the MODIFY command.

P

Insert prior to string. This command has the form:

P [k] /string1/string2/

k has the same meaning as in D (delete string). "string2" is inserted just prior to "string1" for lines specified in the MODIFY command.

[S]

Substitute string. This command has the form:

S [k] /string1/string2/

k has the same meaning as in D (delete string). "string2" is substituted for "string1" for lines specified in the MODIFY command.

Usage Examples

The following figures illustrate the use of some of the frequently used EDITOR commands. All commands in these examples are terminated by pressing the RETURN key. Underlined lines are those messages displayed on the console by the EDITOR.

```
OS1000 READY FOR COMMANDS
EDIT
EDIT > CLEAR
WORK FILE CLEARED
EDIT > INS 10,10
000010 SYSPDL:SYSTEM;
000020 VOLUME HOST = POWERVS, PLABEL = YES;
000030 BLOCK LENGTH = 2048;
000040 ACCT USER = (BIN,TRAY);
000050 21:JOB;
000060 END;
000070 (RETURN)
EDIT> SAVE SYSPDL.JSL
EDIT> CLEAR
EDIT> END
OS1000 READY FOR COMMANDS
```

Figure 8-1. Create and Save a File in the JSL File Directory.

```
OS1000 READY FOR COMMANDS
EDIT
EDIT > CLEAR
WORK FILE CLEARED
EDIT > GET SYSPDL.JSL
EDIT > KEYS
BEGINNING LINE NUMBER 000010
ENDING LINE NUMBER 000060
INS 11
000011 /* SYS JDL */
000012 (RETURN)
EDIT > FIND 10-60/JOB/
21:JOB;
EDIT > MODIFY 50
21:JOB;
EDIT > S/JOB/JDE/
21:JDE;
EDIT > DIS 40
ACCT USER = (BIN,TRAY);
EDIT > REMOVE 40
      1 RECORDS DELETED
EDIT > REN
EDIT > DISPLAY
        ———
        ———
        ———   } Lines displayed on console
        :
        ———
EDIT > SAVE SYSPDL.JSL
EDIT > CLEAR
WORK FILE CLEARED
EDIT > END
OS1000 READY FOR COMMANDS
```

Figure 8-2. Modify and Save File of Figure 8-1.

```
OS1000 READY FOR COMMANDS
EDIT
EDIT > COPY SYSPDL.JSL TSTSYS.JSL
CREATING FILE TSTSYS.JSL
EDIT > FILE JSL
      OSWTR
      IBMPDL
      TSTPDL   } List of all files in JSL directory
        :
      TSTSYS
EDIT > GET TSTSYS.JSL
EDIT > PRINT
PRINT JOB QUEUED, MUST EXIT TO PRINT
EDIT > END
OS1010 STARTING JOB 00003
OS1000 READY FOR COMMANDS
OS1020 JOB 00003 HAS COMPLETED INPUT PHASE
OS0020 RESUMING OUTPUT
OS1030 JOB 00003 HAS COMPLETED PRINTING
OS1000 READY FOR COMMANDS
EDIT
EDIT > DELETE SYSPDL.JSL
FILE SYSPDL.JSL DELETED
EDIT > END
OS1000 READY FOR COMMANDS
```

Figure 8-3. File Manipulation Commands

9. OPERATING SYSTEM DIAGNOSTIC SOFTWARE

Introduction

Operating System Diagnostic Software (OSDS) is a task of the 9700 Operating System Software (OSS) which provides the primary diagnostic software support for input and output hardware subsystems including their controlle s It provides analysis and display of information logged by OSS for recoverable failures in the control subsystem. OSDS also provides guidance to the operator for recovery and generates codes for the dispatch of a customer engineer OSDS is also the use s primary software tool for verifying the operability of the 9700 hardware.

OSDS Communications

While OSDS is operating, it provides a continuous display of information to the user as to what OSDS is doing and/or what it expects of the user. The user communicates with OSDS by requesting a particular mode (PROBLEM or VERIFY) and OSDS responds to the user via formatted displays to the keyboard/display. OSDS also produces printer test data (to test or verify operation of the printer) and a hard copy of error log information.

PROBLEM Mode

The PROBLEM mode of OSDS is activated by the operator because of a request by OSS or by a operator recognized problem.

The system requests the operator (via a message to the keyboard/display) to initiate PROBLEM mode analysis when further productive operation of the 9700 is not possible. This is done only after all appropriate automatic and operator assisted recovery techniques have been exhausted. The operator requests the system to initiate the PROBLEM mode when a problem exists which is not detectable by OSS (e.g., copy quality) or degraded operation of the 9700 is possible but not acceptable.

The PROBLEM mode is initiated by the operator typing "PROBLEM" on the keyboard/display. After the operator's request, OSDS will respond by displaying on the console its selection/display options. An example of an operator/OSDS dialogue is illustrated in Figure 9-1.

Effective dispatch of a customer engineer is provided by OSDS in the PROBLEM mode by determining the nature of the problem from the user inputs solicited or from the OSS error log. OSDS will inform the user that a customer engineer is required under the following conditions:

- A OSS reported problem indicates the need for a service call.
- An operator reported problem indicates the need for a service call.
- A diagnostic test failure indicates the need for a service call.

VERIFY Mode

OSDS provides the verification of the tape subsystem, the printer subsystem, and/or the entire system by performing the following actions:

- Analyzes any system reported problem.
- Reports on system or subsystem capability for job processing (i.e., no detected problems operating possible with minor problems, printer inoperable).
- Provides guidance on further operator action.

The VERIFY mode of OSDS is initiated by the operator typing "VERIFY" on the keyboard/display.

```
OS1000 READY FOR COMMANDS                          (OSS command level)
PROBLEM                                            (user types in PROBLEM)
OSDS          ANALYSIS: RUNNING                    (system responds)
OSDS          SYSTEM ERROR LOG SAVE: RUNNING
OSDS          PROBLEM ANALYSIS: RUNNING

WHICH OF THE FOLLOWING TYPES OF PROBLEMS DO YOU WISH TO REPORT?

    1.  COPY QUALITY PROBLEMS.
    2.  FREQUENT JAMS.
    3.  BIN PROBLEMS.
    4.  TRAY PROBLEMS.
    5.  TAPE PROBLEMS.
    6.  OTHER SYSTEM PROBLEMS.

        INDICATE SELECTION BY ENTERING NUMBER
>1(RETURN)                                         (user makes selection)
WHICH COPY QUALITY PROBLEM DO YOU HAVE?

    1.  COPY IS DIRTY.
    2.  COPY HAS DARK BACKGROUND.
    3.  COPY HAS LINES.
    4.  COPY HAS STREAKS.
    5.  COPY IS BLANK.
    6.  COPY HAS DELETIONS (BLANK SPOTS).
    7.  COPY IS BLACK.
    8.  COPY HAS UNFUSED TONER.
    9.  OTHER.
```

Figure 9-1. Operator/OSDS Sample Display

```
        INDICATE SELECTION BY ENTERING NUMBER
>1(RETURN)                                          (user makes selection)
DO YOU WANT TO REPORT ANY PROBLEMS IN ADDITION TO THOSE BELOW?

    10:00:50    COPY IS DIRTY

    1.  YES. I WISH TO REPORT ADDITIONAL PROBLEMS.
    2.  NO MORE PROBLEMS TO REPORT.

        INDICATE SELECTION BY ENTERING NUMBER
>2(RETURN)                                          (user makes selection)
CALL XEROX SERVICE AND REPORT THESE NUMBERS

    10:00:50

OPERATOR - INDICATE THE ACTION YOU HAVE TAKEN.

    1.  SERVICE CALL HAS BEEN PLACED. WAITING FOR SERVICE.
    2.  SERVICE CALL HAS BEEN PLACED. RETURN TO THE OPERATING SYSTEM.
    3.  SERVICE NUMBERS COPIED FOR LATER CALL. RETURN TO OPERATING SYSTEM.
    4.  NONE OF THE ABOVE. RETURN TO THE OPERATING SYSTEM.

        INDICATE SELECTION BY ENTERING NUMBER
>2(RETURN)                                          (user makes selection)
OSDS HAS RETURNED TO OSS
OS1000 READY FOR COMMANDS                           (return to OSS command level)
```

Figure 9-1. Operator/OSDS Sample Display (Continued)

10. ELECTRONIC FORMS

General

The Electronic forms facility is a major feature of the Xerox 9700. Electronic forms provide the capability for precisely defining forms in order to highlight information for maximum visual impact. They also improve the readability of the printed output without the typical disadvantages associated with preprinted forms (i.e., registration and form-alignment problems, expensive inventory of preprinted forms, and expenditure of operator's time to change forms). The Forms Description Language (FDL) provides the user with the facilities to create "electronic" forms. It is an easy to use, keyword oriented language which allows the user to specify:

- Page orientation - protrait or landscape.
- Font selection - multiple fonts can be specified.
- Vertical and horizontal lines - specified and positioned in line printer measurements (line number and character position) or inches, centimeters, or dots.
- Line width and type - hairline, medium, or bold and solid, broken, or dotted.
- Shading - to define shaded areas and a variety of shading densities.
- Positioning of captions - exact location on a page or automatic placement within a specified area.
- Selection and positioning of logos.

A summary of the FDL command syntax and usage of the FDL processor is provided in the following section. Further details on using FDL to create forms is available in the Forms Creation Guide (Xerox Publication No. 91 00 01).

FDL Processor

The FDL processor (a 9700 system task) accepts source language FDL statements created by the user. The source statements may be processed by FDL from either tape or a file on the system disk. Source statements processed by FDL from tape are automatically cataloged in the FSL library on disk, thereby allowing for corrections and modifications via the 9700's editing facilities (described in Chapter 8). The editing facilities can also be used to create FDL source files directly on disk.

The FDL processor is invoked by the following command entered from the operator's console:

FDL [filename] [,NOPRINT]

where

filename — specifies the name of a source Forms Description Language file in the Form Source Library (FSL) on the system disk that is to be used for source input to the FDL task. If no filename is specified, it is assumed that source input is to come from tape.

NOPRINT — specifies that no listing of the FDL source statements, no form summary, and no sample form are to be printed. However, if an error is detected during the processing of the form, the source statement listing and form summary are unconditionally printed.

FDL Command Syntax Summary

In defining the FDL source statement syntax the following symbols are used:

SYMBOL	MEANING	
c_o	Coordinate, origin	Defined in y unit, x unit, where y is measured along the vertical axis, and x along the horizontal axis
c_s	Corrdinate, start	
c_e	Coordinate, end	
c_a	Coordinate, absolute	
c_i	Coordinate, incremental	
unit	IN, INCHES, CM, CENTIMETERS, or DOTS	
id	Identifier consisting of one to six alphanumeric characters	

Set-up Commands

```
FORM id;
LANDSCAPE;
PORTRAIT
GRID [UNIT][IS] { FMTm
                  n IN
                  n INCHES
                  n CM
                  n CENTIMETERS
                  n CPI n LPI
                  DOTS } [ ORIGIN y [ IN
                                      INCHES
                                      CM
                                      CENTIMETERS
                                      DOTS ] x [ IN
                                                 INCHES
                                                 CM
                                                 CENTIMETERS
                                                 DOTS ] ];
FONT id;
FONTS id ... id ;
```

Section Commands

```
[ BEGIN ]SECTION id;
DO SECTION id AT y [unit] x [unit];
END SECTION;
```

Form Structure Commands

LINE or LINES:

AT c_o [unit] [DRAW] [n] [HORIZONTAL / VERTICAL] {LINE / LINES} [IN unit] [FROM] c_s [unit]

TO c_e [unit] [USING [SOLID / BROKEN / DOTTED] [HAIRLINE / 0 / 1 / 2]]

[[AND] [REPEAT] [HORIZONTALLY / VERTICALLY] {AT c_a [unit] [c_a [unit] ...] / EVERY c_i [unit]}];

BOX or BOXES:

AT y [unit] x [unit] [DRAW] [n] {BOX / BOXES} [IN unit] n [unit] [WIDE] [BY] m [unit] [HIGH]

[[USING] [SOLID / BROKEN / DOTTED] [HAIRLINE / 0 / 1 / 2]
[SHADING] [LIGHT / MEDIUM / HEAVY]]

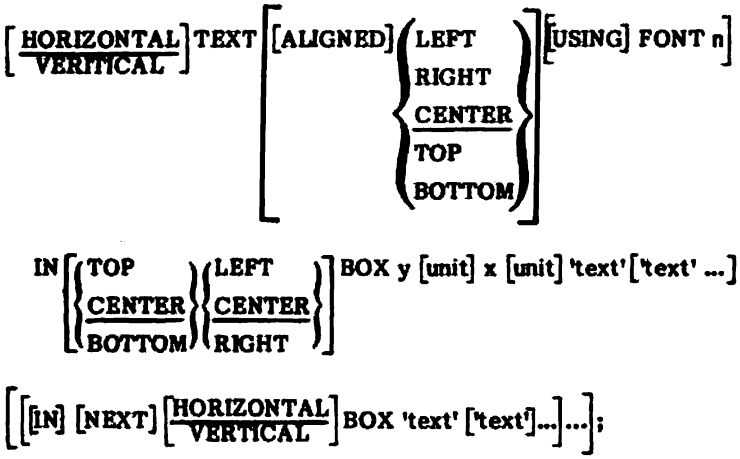
[[AND] [REPEAT] [HORIZONTALLY / VERTICALLY] {AT c_a [unit] [c_a [unit] ...] / EVERY c_i [unit]}];

TEXT:

[HORIZONTAL / VERTICAL] TEXT [[ALIGNED] (LEFT / RIGHT / CENTER / TOP / BOTTOM) [USING] FONT n]

AT y [unit] x [unit] 'text' ['text' ...] ;

TEXT IN BOX:



LOGO, COMMENT and END Commands

```
LOGO id AT y [unit] x [unit];
COMMENT text;
END;
```

11. FONTS

Introduction

A font is a specific set of characters of one size, style, and orientation (portrait or landscape). A font may have up to 240 alphanumeric characters, punctuation marks, line segments, shading elements, special symbols, and form characters. The number of lines per inch (3 to 18) as well as the number of characters per inch (4 to 30) are both inversely proportional to the font size. That is, small font sizes allow more lines and characters per inch than large font sizes. Xerox 9700 fonts range from 4 to 24 point characters. Within the Xerox 9700 the number of lines per inch divided into 72 will give the approximate point size of the characters of the font (i.e., 8 lpi is 9 points).

As listed in the Xerox 9700 Font Users Guide (Publication No. 91 00 03) standard fonts are available in a variety of point sizes, character spacing (fixed and proportional), line spacing, character orientation (landscape or portrait), and styles (bold, script, etc.). Typical electronic form and variable data fonts are as follows:

- Univers
- Press Roman
- OCR A readable
- OCR B readable
- Sanserif based on Xerox 1200 Computer Printing System
- Serifed typewritier style

Special fonts may be created (or digitized) at the Xerox Digitization Center upon user's request. Fonts created at the Xerox Digitization Center are output on magnetic tape in a binary format. These fonts may be loaded from tape to the Xerox 9700 disk by entering the COPY command via the keyboard (refer to "Copying a File", Chapter 7).

Font Usage

The fonts to be utilized in a print job are specified by the PDE command in the Job Descriptor Entry (see Chapter 5). The fonts are identified by the name under which they have been previously cataloged. In addition, if a form is required by a particular print job, additional fonts may be specified within the forms definition (see Forms Creation Guide, Xerox Publication No. 91 00 01).

During the printing of a job, any font from the list of fonts on the PDE command may be selected on a character-to-character basis. If different fonts are specified on the same print line, the largest line spacing value associated with those fonts will be used to determine the position of the print line relative to the previous line.

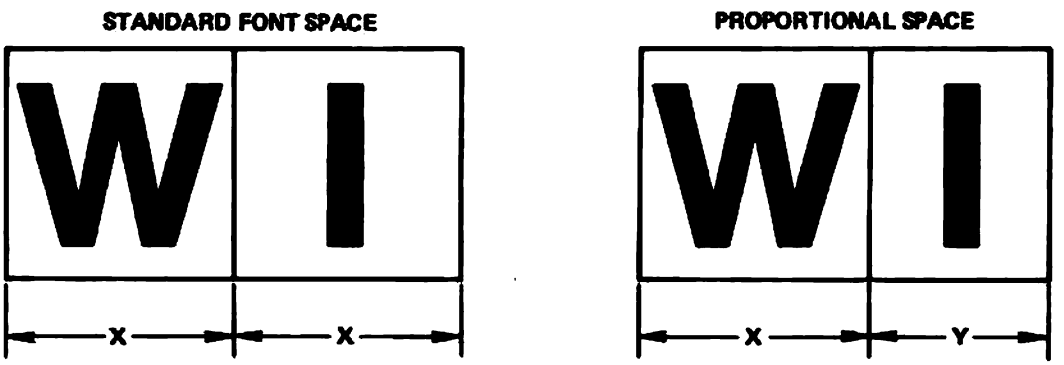
Font Data

Font data consists of encoded bit matrices of dots which represent each character or graphic symbol in the font. Also associated with each font are optimal line spacing and character spacing parameters, and centering information. When a landscape character is printed, it is centered in its character cell horizontally; portrait characters are aligned vertically.

Font characters may also be combined to create logos. The largest character which will be created is .85" x .85". Therefore, logos which exceed this size are created from a series of font characters. Information carried in the logo file specifies the relative placement of the characters so that the user may treat the logo, as well as other complete figures (i.e., signatures), as a single character.

Proportional Spacing

Provisions are made for proportionally spaced printing as an option. This requires a special font for each proportional style, size and orientation. Care must be taken by the customer when using a proportionally spaced font, to preclude form line data from interfering with the variable data, and to stay within character limitations stated above.



12. ON-LINE PRINTING SYSTEM

Introduction

This chapter describes various features of the 9700 On-Line Printing System and how these features are made available to the user.

The 9700 on-line system emulates the IBM 3211 Printer/3811 Printer Control Unit, with the restrictions noted below, under "System Restrictions". The 9700 on-line system may be attached to a Byte Multiplexer, Block Multiplexer or Selector Channel of an IBM 360 Computer Models 22, 30 and up, IBM 370 Models 135 and up, and operates in either multiplexer or burst mode. No changes to the IBM software are required.

The 9700 on-line system is an integrated hardware/software system consisting of a xerographic printer, laser imaging subsystem, an output stacker, host input interface, a disk memory subsystem and an intelligent software programmable processor. The operating system program resides in the programmable processor.

Description of System

The 9700 on-line system shares most of the same hardware with the off-line. The common hardware elements are:

- Xerographic printer
- Laser imaging unit
- Output stacker
- Disk system
- Keyboard/display unit
- Programmable processor

The major difference lies in the replacement of the off-line magnetic tape input unit with the direct interface to an IBM host computer. The on-line/host interface is via the host I/O channel. The device is an On-Line Interface (OLI).

System Features

The general features of the 9700 on-line system are listed below. For a detailed description of processing of each 3211 command, see "Processing of 3211 Commands" below.

- Requires no changes in the host computer hardware or operating system.
- Accepts and prints data in a manner which emulates the IBM 3211/3811.
  - Respond to 3211 diagnostic commands which are appropriate to 9700 (as documented in 3211/3811 Component Description Manual).
  - The 3211 Universal Character Set Buffer (UCSB) feature is supported. The UCSB is used by 9700 to generate folded and unfolded character set translation tables.
  - The 3211 Forms Control Buffer (FCB) feature is supported.
- Processing commands which could result in the loss of data will be flagged at job start time by a warning message. Examples of this are BDELETE, BSELECT, RDELETE, RSELECT, and COPIES = 0.
- The 9700 can be powered on or off without adversely affecting the host system operation.
- Full upward compatibility with the Xerox 1200 On-Line Printing System.
- Full availability of 9700 software features except as specifically called out in this chapter.

System Restrictions

- 3211 Dualing feature is not supported. However, it will not impede the 9700 user. The bits, if set, will be ignored.
- Copy sensitive features of the 9700 are not supported. For example, the MODIFY and FORMS parameters of the OUTPUT command may not specify a copy range.

Processing of 3211 Commands

While certain minor differences exist between the 3211/3811 and the 9700 on-line, the responses are appropriate and cause no programming changes to IBM software.

The 3211 commands are processed by 9700 on-line as follows:

- All write commands perform the included carriage functions after the line is printed.
- Carriage Control commands are processed exactly as the 3211 does, including the overprint function.
- LOAD UCSB is processed by 9700 on-line as follows:
  - The train-image field is accepted and ignored, because it is not needed.
  - The associative field is used to build the translate table based upon the FOLD or UNFOLD command in effect.
- FOLD and UNFOLD are processed as the 3211 does.
- BLOCK (Disallow) DATA CHECK is processed as the 3211 does.
- ALLOW DATA CHECK is processed as the 3211 does.
- LOAD FCB is processed as the 3211 does.
  - The print position indexing byte is processed as the 3211 does.
- The 132 or 150 character per line options are supported.
- The Diagnostic Commands for the 3211 are emulated as follows:
  - TEST I/O is processed exactly as the 3211 does.
  - SENSE is processed exactly as the 3211 does.
  - NO-OP is processed exactly as the 3211 does.
  - READ PLB is processed as the 3211 does.
  - READ UCSB is processed as the 3211 does.
  - READ FCB is processed as the 3211 does but will not result in forms misalignment. Both 6 and 8 lines per inch options are supported.
  - CHECK READ is processed as the 3211 does except if a parity error occurs, all bytes in PLB will be flagged as having bad parity. This has no effect on operational software since this command is only used by IBM diagnostics.
  - DIAGNOSTIC WRITE is processed as in the 3211.
  - RAISE COVER is accepted but functionally ignored.
  - DIAGNOSTIC GATE command is functionally equivalent to the 3211.
    - 9700/OS sets a flag for use during the processing of other commands (in particular, 'CHECK READ'). This flag is cleared at the completion of the command following the 'DIAGNOSTIC GATE' command.

System Software Structure

The 9700 on-line system software consists of an operating system called the 9700 Operating System (9700/OS). Its functions are to accept print data from the host computer and process the print data through the xerographic printer and stacker.

The 9700/OS is functionally divided into tasks. The System Executive passes logical control between Input Data Processing and Output Data Processing. The latter task is identical to the off-line version (see Chapter 5, "Output Data Processing").

The Input Data Processing is conceptually similar to the off-line version. The differences all center about the difference in physical devices (tape vs. channel) and the on-line's emulation of the 3211 printer. Basic on-line/off-line functions of Input Processing are amplified by the following functions of on-line.

- Handles all interactions with the On-Line Interface.
- Builds translate tables based on the contents of the Universal Character Set Buffer (UCSB) command.
- Processes FCB to:
  - Emulate 6 or 8 lines per inch
  - Assign channel to line numbers
  - Sets margins based on Print Position Indexing Byte

JDE, DJDE, FCB and UCSB Relationship

Both on-line and off-line printing are controlled through parameters contained in the Job Control Block (JCB). The JCB is built from the JDE and modified by DJDEs. On-line users may optionally restrict the FCB and UCSB information from modifying the JCB. The user is thus given the flexibility of controlling the updating of the JCB by accepting or rejecting the merging of host transmitted FCB and UCSB inputs.

Print Positioning

The control of line positioning on the page is determined by the various JDE parameters. The user has control of font selection and line spacing. The BEGIN command locates the first line. The 3211 emulation is a subset of the control features already in the 9700 repertoire.

Job Separation

The system relates to each 'job' a JDE which describes the processing to be performed. DJDEs and FCBs are applied to this JDE. The off-line job is defined in terms of end of files on a tape or special processing. For the on-line stream, jobs are defined by special processing criteria. These are site and software system dependent. PDL statements are used to defined offset criteria or banner pages (e.g., pattern, mask, length, offset) which identifiy job transition.

Overprinting

The 9700 on-line system overprints as a 3211 impact printer does.

Operating the System

The operation of the 9700 on-line system is an extension of the off-line system. The differences are described below:

On-line Configuration

Initializing the disk program, fonts, forms, logos, and JDLs/JDEs, may be accomplished through the use of the HOSTLOAD utility on the on-line or switchable system. This program must be disk resident at installation time. The operator calls HOSTLOAD off the disk. It accepts data from the host, distinguishes between the aforementioned classes of files, places these on the disk and properly catalogs them in the file system. HOSTLOAD will distinguish between the different classes of files by requiring the data coming from the host to be preceded by unique identifying records.

On-line/Off-line Control

Two selectable utilities give the operator control of performing a logical connect or disconnect from the host. These are called ONLINE and OFFLINE respectively. An acknowledgement of the completion of these requested actions is made at the 9700 console.

Multiple Copies

The "copies" option of the START command has the same meaning in on-line as for tape input. Printing collated multiple copies (without retransmission from the host) may lead to a situation requiring the operator's attention. When the number of pages of a multicopy document is less than the capacity of the disk, or when printing uncollated multiple copies (regardless of report length) or when each copy is transmitted separately by the host, no difficulty exists. The copies will be correctly generated and set in the bins. However, if the number of pages of a document is greater than the disk buffer can hold, then the collated copies will be generated X pages at a time, where X is the number of pages that the disk can hold at one time. This situation will entail the manual merging of the sets. When this condition occurs a message is displayed and the operator can optionally select to retransmit the job to avoid this manual step.

Host Console Message Handling

The messages output to the host operator contain a class that affects the 3211 emulation. The class contains 'MOUNT FORM X ON YY' where YY is the particular 3211. X can imply a change of stock, a 9700 cataloged form, or a multicopy form. When this message shows a change in forms or copies, the 9700 on-line must be externally informed of the end of the present report being input. This is done by the operator issuing a STOP INPUT command followed by an END. This additional on-line command causes the last page of the present job being input to be formatted and placed on disk and the job made ready for printing. The operator then enters CONTINUE FORMS = xxx, COPIES = yyy, COLLATE = YES. This now establishes the new copies and forms parameters to be associated with all subsequent reports until changed explicitly by a new operator entry.

APPENDIX A. SYSTEM ERROR MESSAGES

Level 0 - Confirmation Messages

Level 0

OS0010 RESUMING INPUT
OS0020 RESUMING OUTPUT
OS0050 MAIN TRAY LOWERING
OS0110 BIN 1 LOWERING
OS0120 BIN 2 LOWERING
OS0200 MAIN TRAY SELECTED
OS0210 AUX TRAY SELECTED
OS0310 BIN 1 SELECTED
OS0320 BIN 2 SELECTED
OS0500 OUTPUT STOPPED
OS0510 INPUT STOPPED
OS0610 PAGE SPACING FORWARD
OS0620 PAGE SPACING BACKWARD
OS0900 JOB ***** ABORTED
OS0950 TASK ABORTED
OS0990 RESETTING THE SYSTEM

Level 1 - Information Messages

Level 1

OS1000 READY FOR COMMANDS
OS1010 STARTING JOB *****
OS1020 JOB ***** HAS COMPLETED INPUT PHASE
OS1030 JOB ***** HAS COMPLETED PRINTING
OS1050 REWIND WILL BE DONE AT END-OF-JOB
OS1080 START ACCOUNTING REPORT
OS1090 END ACCOUNTING REPORT
OS1110 BIN 1 BUSY
OS1120 BIN 2 BUSY
OS1150 JOB QUEUE FULL
OS1210 BIN 1 ALREADY UNLOADED
OS1220 BIN 2 ALREADY UNLOADED
OS1300 NUMBER OF ACTIVE FORMS IS ***
OS1310 NUMBER OF ACTIVE FONTS IS ***
OS1380 ALIGNMENT IS ***** SCAN LINES AND ***** DOTS
OS1390 SCAN AND DOT VALUES MUST BE A POSITIVE NUMBER, RETRY ALI
OS1400 SAMPLE IGNORED...PRINTER IDLE
OS1410 SAMPLE NOT ALLOWED BY JDE
OS1420 PAGE SPACING NOT ALLOWED BY JDE
OS1430 BLOCK SPACING NOT ALLOWED BY JDE
OS1500 PAGE SPACING STOPPED BY BEGINNING-OF-REPORT
OS1510 PAGE SPACING STOPPED BY END-OF-REPORT
OS1520 BLOCK SPACING STOPPED BY END OF FILE

OS1550 SPACE OR MOVE FUNCTION STOPPED BY END OF DATA
OS1560 TAPE REWIND COMPLETE
OS1600 INPUT TASK NOT ACTIVE
OS1610 OUTPUT TASK NOT ACTIVE
OS1650 OUTPUT PROCESSING HAS CAUGHT-UP WITH INPUT PROCESSING
OS1700 INPUT PROCESSING NOT CURRENTLY STOPPED
OS1710 INPUT PROCESSING ALREADY STOPPED
OS1730 CLEAR INVALID WITH DISPLAY, CLEAR IGNORED
OS1750 NOTHING TO ABORT
OS1760 TASK TERMINATED
OS1800 INPUT PROCESSING ABORTING
OS1810 PRINTING ABORTING
OS1850 JOB STARTED AT END OF DATA, NOTHING FOUND TO PRINT
OS1910 THIS FUNCTION OF INPUT NOT IMPLEMENTED
OS1920 THIS SYSTEM FUNCTION NOT IMPLEMENTED

Level 2 - Routine Maintenance or Action Messages Level 2

OS2000 ENTER "CONTINUE 0" TO RESUME PRINTING
OS2010 MOUNT INPUT TAPE; "CONTINUE I" WHEN READY
OS2020 MOUNT NEXT VOLUME; "CONTINUE I" WHEN READY
OS2030 TAPE AT EOV; "CONTINUE I" WILL REWIND TAPE
OS2060 REFILL MAIN TRAY
OS2070 REFILL AUX TRAY
OS2100 MAIN TRAY NOT READY
OS2110 AUX TRAY NOT READY
OS2150 DISPLAY (Y/N) ?
OS2210 BIN 1 FULL
OS2220 BIN 2 FULL
OS2310 BIN 1 NOT READY
OS2320 BIN 2 NOT READY
OS2400 PRINTER MISFEED DETECTED. CHECK PAPER SUPPLY
OS2500 REFILL DRY INK HOPPER
OS2550 DRY INK RECLAIM BOTTLE OR FILTER BAG FULL
CHANGE, THEN PUSH 'BOTTLE/FILTER RESET' BUTTON
OS2580 POP SENSOR REQUIRES CLEANING
OS2600 TAPE VOLUME OUT OF SEQUENCE; MOUNT CORRECT VOLUME
OS2700 KEY-IN LOST. RE-ENTER
OS2710 INVALID COMMAND RE-ENTER
OS2720 INVALID CONTROL KEY. RETRY
OS2730 REQUESTED TASK NOT FOUND IN SYSTEM. CHECK & RETRY
OS2740 JDE NOT FOUND. CHECK AND RETRY
OS2741 JDL NOT FOUND. CHECK AND RETRY
OS2750 JOB NOT FOUND. CHECK & RETRY
OS2760 SAMPLE LOGO NOT FOUND. CHECK AND RE-ENTER
OS2800 MOVE OR SPACE FUNCTION COMPLETE
ENTER 'CON I' OR 'CON JDE, JDL' TO START NEXT REPORT
OS2820 TASK NOT ALLOWED TILL SYSTEM STATUS = 'IDLE'

OS2840 OUTPUT MUST BE STOPPED BEFORE PAGE SPACING CAN BE DONE

OS2880 MAX FONTS & FORMS EXCEEDED. ENTER NEW VALUE. RESTART JOB

OS2885 MAX NUMBER OF FONTS EXCEEDED. ENTER NEW VALUE. RESTART JOB

OS2900 TAPE BLOCK LENGTH EXCEEDS JDE MAX. DO THE FOLLOWING:

* ABORT AND RETRY. SPECIFYING ANOTHER JDE/JDL

OS2910 NO ACCOUNTING FILE ENTRY FOR DEPARTMENT ***

OS2950 SYSTEM ERROR HAS CAUSED A PRINTER SOFT STOP

OS2980 USE LOGON COMMAND TO CHANGE OPERATOR CLASS CODE

OS2990 USE 'PROBLEM' AT EARLIEST OPPORTUNITY

Level 3 - Printer Problem Messages Level 3

OS3010 PRINTER IS WARMING UP

OS3100 XEROGRAPHIC ENGINE INTERLOCK OPEN. CLOSE DOORS & COVER

OS3120 IMAGING MODULE INTERLOCK OPEN. CLOSE COVER

OS3150 OUTPUT MODULE INTERLOCK OPEN. CLOSE COVER

OS3200 FUSER TEMPERATURE IS BELOW MINIMUM OPERATING TEMPERATURE
WAIT THREE MINUTES

OS3300 REMOVE JAMMED SHEET FROM SAMPLE TRAY

OS3310 REMOVE JAMMED SHEET FROM BIN 1

OS3320 REMOVE JAMMED SHEET FROM BIN 2

OS3330 BIN JAM DETECTED

OS3400 PRINTER JAM...CLEAR PAPER PATH

OS3450 PRINTER JAM...CLEAR PAPER PATH INCLUDING PHOTORECEPTOR

OS3700 SUSPECTED PAGE-DELIVERY ERROR. CHECK OUTPUT

Level 4 - System or Tape Problem Messages Level 4

OS4010 CANNOT FIND END OF TAPE RELECTOR STRIP. "CONTINUE I"

OS4050 TOO MANY FORMS/FONTS SPECIFIED VIA FORMS/FONTS COMMANDS
ENTER NEW VALUES VIA THE FORMS/FONTS COMMAND.

OS4100 PRINTER IS IN MANUAL MODE. USE 'PROBLEM'

OS4150 TAPE DRIVE IS OFF-LINE. 'CONTINUE I' WHEN TAPE IS READY

OS4200 TAPE DRIVE NOT RESPONDING. DO ONE OF THE FOLLOWING:

* VERIFY DRIVE IS ON-LINE. "CONTINUE I" WHEN READY
* ENTER 'RESET'. THEN USE 'PROBLEM'

OS4310 IRRECOVERABLE TAPE DUMP READ ERROR

OS4500 BAD BLOCK ON TAPE. DO ONE OF THE FOLLOWING:

* 'CONTINUE I' TO RETRY READ
* ABORT JOB AND CLEAN THE TAPE DRIVE THEN
  RESTART JOB
* SPACE 1 REPORT
* MOVE 1 BLOCK
* IF PROBLEM CONTINUES, RUN ANOTHER JOB TAPE

OS4990 SYSTEM RELIABILITY LOG BEING LOST. ENTER 'RESET'. USE 'PROBLEM'

OS6000 INSUFFICIENT MEMORY FOR "ACCTINFO"; CONTINUE OR ABORT?
OS6010 LABEL ERROR : INVALID LABEL FORMAT; CONTINUE OR ABORT?
OS6011 LABEL ERROR : VOL1; CONTINUE OR ABORT?
OS6012 LABEL ERROR : HDR1; CONTINUE OR ABORT?
OS6013 LABEL ERROR : UHL, TM, OR HDR2; CONTINUE OR ABORT?
OS6014 LABEL ERROR : EOF OR EOV; CONTINUE OR ABORT?
OS6015 LABEL ERROR : TM, HDR, OR UHL; CONTINUE OR ABORT?
OS6016 LABEL ERROR : TM OR USER; CONTINUE OR ABORT?
OS6017 LABEL ERROR : TAPE MARK; CONTINUE OR ABORT?
OS6018 LABEL ERROR : EOF; CONTINUE OR ABORT?
OS6019 LABEL ERROR : EOV; CONTINUE OR ABORT?
OS6020 LABEL ERROR : UVL OR HDR1; CONTINUE OR ABORT?
OS6021 LABEL ERROR : UTL, TM, OR EOF; CONTINUE OR ABORT?
OS6022 LABEL ERROR : ANSI OPTION 3; CONTINUE OR ABORT?
OS6023 LABEL ERROR : 1HDR; CONTINUE OR ABORT?
OS6024 LABEL ERROR : 1EOR, TM, OR 1EOF; CONTINUE OR ABORT?
OS6025 LABEL ERROR : 1EOR OR 1EOF; CONTINUE OR ABORT?
OS6026 LABEL ERROR : 1EOR; CONTINUE OR ABORT?
OS6027 LABEL ERROR : 1EOF; CONTINUE OR ABORT?
OS6028 LABEL ERROR : 1ERI, TM, OR 1HDR; CONTINUE OR ABORT?
OS6029 LABEL ERROR : BASIC TAPE; CONTINUE OR ABORT?
OS6030 LABEL ERROR : EOF OR EOR; CONTINUE OR ABORT?
OS6031 LABEL ERROR : EOR; CONTINUE OR ABORT?
OS6032 LABEL ERROR : HDR1, UVL, OR VOL; CONTINUE OR ABORT?
OS6033 LABEL ERROR : HDR1 OR UVL; CONTINUE OR ABORT?
OS6034 LABEL ERROR : TM, EOF2, OR UTL; CONTINUE OR ABORT?
OS6035 LABEL ERROR : TM OR TRAILER; CONTINUE OR ABORT?
OS6036 LABEL ERROR : TM OR HDR1; CONTINUE OR ABORT?
OS6037 LABEL ERROR : STANDARD HDR; CONTINUE OR ABORT?
OS6038 LABEL ERROR : STANDARD EOF OR EOV; CONTINUE OR ABORT
OS6039 LABEL ERROR : STANDARD EOF; CONTINUE OR ABORT?
OS6040 LABEL ERROR : STANDARD EOV; CONTINUE OR ABORT?
OS6041 LABEL ERROR : SPECIAL BLOCK LBL; CONTINUE OR ABORT?
OS6042 LABEL ERROR : ILLEGAL POWER V/S FORMAT; CONTINUE OR ABORT?
OS6080 LABEL ERROR : BAD RECORD FORMAT; CONTINUE OR ABORT?
OS6090 LABEL ERROR : ILLEGAL BLOCK LENGTH; CONTINUE OR ABORT?
OS6200 LABEL AND FILE BLOCK COUNT MISMATCH. CONTINUE OR ABORT?
OS6500 CANNOT VALIDATE FIRST DATA RECORD; SPACE TO NEXT REPORT
OS6550 DATA NOT FORMATTED AS SPECIFIED; SPACE TO NEXT REPORT
OS6700 SYNTAX ERROR IN DJDE. CONTINUE OR ABORT?
OS6710 DISK ERROR ON DJDE READ. CONTINUE OR ABORT?
OS6750 OUT OF DYNAMIC MEMORY, DJDE IGNORED. CONTINUE OR ABORT?
RUN NEXT JOB AND REPORT PROBLEM
OS6800 OUTPUT TASK CRASH * RELOADING. CHECK OUTPUT AT END OF JOB
OS6900 DATA ON PAGE EXCEEDS 8½ X 11. CHECK OUTPUT
OS6950 LINE DENSITY EXCEEDED. PAGE WILL NOT BE PRINTED. ABORT OR
ENTER 'CONTINUE O' TO RESUME PRINTING

Level 7 - System Problem Messages — Level 7

OS7100 PCC OR TRANSLATE TABLE UNREADABLE
OS7110 CME FILE NOT FOUND
OS7120 PDE FILE NOT FOUND
OS7130 FONT FILE NOT FOUND
OS7140 FORM FILE NOT FOUND
OS7150 FORM FONT NOT FOUND
OS7300 ACCOUNT FORM FILE NOT FOUND/ACCOUNTING CAN'T PRINT
OS7500 INSUFFICIENT MEMORY FOR JDE
OS7510 INSUFFICIENT MEMORY FOR JDE TABLES
OS7520 INSUFFICIENT MEMORY FOR VFU TABLES
OS7530 INSUFFICIENT MEMORY FOR CME TABLES
OS7540 INSUFFICIENT MEMORY FOR TAPE DUMP PRINT BUFFER
OS7550 INSUFFICIENT MEMORY FOR INPUT BUFFERS
OS7800 INSUFFICIENT MEMORY FOR FORM PRINT
OS7810 INSUFFICIENT MEMORY FOR SAMPLE FONT
OS7820 INSUFFICIENT DYNAMIC MEMORY FOR SAMPLE LOGO
OS7850 TOO MANY DATA AND FORM FONTS AND FORMS SPECIFIED IN JDE
OS7900 FONT MEMORY SIZE EXCEEDED. RUN NEXT JOB. REPORT ERROR
OS7910 JOB TOO BIG FOR AVAILABLE MEMORY-OUTPUT

Level 8 - Probable Severe Software Error Messages — Level 8

OS8010 INPUT FOUND NOTHING TO PRINT
OS8100 REQUESTED TASK ALREADY ACTIVE - REBOOT THE SYSTEM
OS8200 INVALID TMCB RECEIVED FROM A TASK
OS8300 INPUT HAD AN I/O ERROR DURING FORM FILE READ
OS8320 INPUT CANNOT OPEN ACCOUNTING FILE
OS8410 BYTE ALIGNED DISK I/O REQUEST...INPUT ABORTING
OS8420 BAD LBN ON DISK I/O...INPUT ABORTING
OS8430 ILLEGAL ADDRESS SPACE ON DISK I/O...INPUT ABORTING
OS8500 SYSTEM RELIABILITY LOG LOST
OS8600 UNSUCCESSFUL COMPLETION OF SEND DATA DIRECTIVE
OS8700 ILLEGAL META-CODE IN DATA
OS8800 INSUFFICIENT DYNAMIC MEMORY - INPUT
OS8850 FILE MANAGEMENT INITIALIZATION FAILURE. TRY REBOOTING
OS8940 INSUFFICIENT INPUT TASK MEMORY
OS8950 INSUFFICIENT DYNAMIC MEMORY TO RUN REQUESTED TASK
OS8980 UNABLE TO INITIATE INPUT TASK
OS8990 UNABLE TO INITIATE OUTPUT TASK

Level 9 - Probable Severe Hardware Error Messages Level 9

OS9200 TAPE DRIVE ERROR. ENTER 'RESET'. USE 'PROBLEM'

OS9250 FATAL HARDWARE ERROR ON TAPE DRIVE. ENTER 'RESET'. USE 'PROBLEM'

OS9300 IRRECOVERABLE DISPATCHING/IMAGING ERROR

OS9350 PRINTER CONTROLLER REJECTED COMMAND. ENTER 'RESET'. USE 'PROBLEM'

OS9380 PRINTER FAILURE
ENTER 'RESET', THEN USE 'PROBLEM'; OR
ENTER 'CONTINUE 0 TO RESUME PRINTING

OS9400 DISK ERROR/JDE - INPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9420 DISK ERROR/PRINT FILE - INPUT. ENTER 'RESET'. USE 'PROBLEM',

OS9440 DISK ERROR/ACCOUNTING FILE - OUTPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9460 DISK ERROR. ENTER 'RESET'. USE 'PROBLEM'

OS9500 BAD BLOCK ON DISK - INPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9530 DISK ERROR - INPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9550 DISK ERROR - OUTPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9580 IRRECOVERABLE TAPE READ ERROR. ENTER 'RESET'. USE 'PROBLEM'

OS9800 FATAL HARDWARE ERROR ON DISK - INPUT. ENTER 'RESET'. USE 'PROBLEM'

OS9920 NO SYSTEM LOG. CALL DISPATCH, CODE 300099

OS9950 INSUFFICIENT MAIN MEMORY-MEMORY IS DEGRADED. ENTER 'RESET'. USE 'PROBLEM'

APPENDIX B. PDL AND FDL MESSAGES TO KEYBOARD/DISPLAY

PDL Messages

PD0100	NOREPLACE SPECIFIED
PD0150	REPLACE SPECIFIED
PD0200	REPLACE AUTHORIZED BY OPERATOR
PD0250	REPLACE DENIED BY OPERATOR
PD0500	NO SOURCE FILE WILL BE CREATED
PD0900	PDL ABORTED BY OPERATOR
PD1000	PDL TERMINATED
PD2700	OPERATOR COMMAND ERROR, RETRY
PD2740	JSL NOT FOUND, CHECK AND RETRY
PD4500	TAPE READ ERROR
PD4550	PDL INTERNAL ERROR - TAPE I/O
PD7050	TAPE RECORD SIZE ERROR
PD7200	PDL INTERNAL TABLE OVERFLOW
PD8200	INTERNAL ERROR IN PDL
PD8210	INTERNAL ERROR IN PDL
PD8800	INSUFFICIENT DYNAMIC MEMORY TRY REBOOTING THE SYSTEM
PD9400	DISK ERROR - JDL FILE
PD9410	DISK ERROR - WORK FILE
PD9420	DISK ERROR - SOURCE FILE
PD9430	DISK ERROR - CME FILE
PD9440	DISK ERROR - PDE FILE
PD9450	DISK ERROR - PRINT FILE
PD9460	DISK ERROR - PCC TABLE
PD9470	DISK ERROR - CATALOG FILE
PD9500	OPEN ERROR - PRINT FILE
PD9510	PDL RESTART ERROR OPENING SAVE FILE
PD9520	SOURCE OUTPUT FILE OPEN ERROR

FDL Messages

FD0900	OPERATOR REQUESTED ABORT
FD1000	ALL FORMS COMPILED, FORMS COMPILER EXITING
FD1050	FORMS COMPILER RESUMING
FD1800	FORMS COMPILER ABORTING
FD2700	UNRECOGNIZED KEY-IN, KEY-IN IGNORED, MAY BE RE-ENTERED
FD2710	INVALID CHARACTER, RE-ENTER
FD2720	KEY-IN TOO LONG, RE-ENTER
FD2730	PARAMETER TOO LONG, RE-ENTER
FD2740	FSL NOT FOUND, CHECK AND RETRY
FD4500	ERROR IN READING MAGNETIC TAPE
FD8500	STACK UNDERFLOW
FD9400	ERROR IN CLOSING THE SOURCE-INPUT FILE
FD9410	ERROR IN CLOSING THE SOURCE-OUTPUT FILE
FD9420	ERROR IN CLOSING THE LISTING/SUMMARY FILE
FD9430	ERROR IN OPENING THE SOURCE-OUTPUT FILE
FD9440	ERROR IN OPENING THE LISTING/SUMMARY FILE
FD9450	ERROR IN READING THE SOURCE-INPUT FILE
FD9460	ERROR IN READING THE SOURCE-OUTPUT FILE
FD9470	ERROR IN WRITING THE SOURCE-OUTPUT FILE
FD9480	ERROR IN WRITING THE LISTING/SUMMARY FILE
FD9510	ERROR IN CLOSING CONTEXT FILE
FD9520	ERROR IN OPENING CONTEXT FILE
FD9530	ERROR IN READING CONTEXT FILE
FD9540	ERROR IN WRITING CONTEXT FILE

APPENDIX C. CHARACTER SETS

Table C-1. Input/Output Correspondence - USASCII

		Most Significant Bits							
Hexadecimal (rows) (col's.)→		0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
Least Significant Bits ↓	Binary	x000	x001	x010	x011	x100	x101	x110	x111
0	0000			space	0	@	P	`	p
1	0001			!	1	A	Q	a	q
2	0010			"	2	B	R	b	r
3	0011			#	3	C	S	c	s
4	0100			$	4	D	T	d	t
5	0101			%	5	E	U	e	u
6	0110			&	6	F	V	f	v
7	0111			'	7	G	W	g	w
8	1000			(	8	H	X	h	x
9	1001			)	9	I	Y	i	y
A	1010			*	:	J	Z	j	z
B	1011			+	;	K	[	k	{
C	1100			,	<	L	\	l	|
D	1101			-	=	M	]	m	}
E	1110			.	>	N	^	n	~
F	1111			/	?	O	_	o	

Note: Unassigned codes print as blank (space).

Table C-2. Input/Output Correspondence - EBCDIC

Least Significant Bits: Hexadecimal (rows)	Binary	Most Significant Bits: 0 (0000)	1 (0001)	2 (0010)	3 (0011)	4 (0100)	5 (0101)	6 (0110)	7 (0111)	8 (1000)	9 (1001)	A (1010)	B (1011)	C (1100)	D (1101)	E (1110)	F (1111)
0	0000					space	&	-									0
1	0001							/		a	j		\[3]	A	J		1
2	0010									b	k	s	{[3]	B	K	S	2
3	0011									c	l	t	}[3]	C	L	T	3
4	0100									d	m	u	[[3]	D	M	U	4
5	0101									e	n	v	][3]	E	N	V	5
6	0110									f	o	w		F	O	W	6
7	0111									g	p	x		G	P	X	7
8	1000									h	q	y		H	Q	Y	8
9	1001									i	r	z		I	R	Z	9
A	1010					¢[1]/\[2]	![3]	^	:								
B	1011					.	$	,	#								
C	1100					<	*	%	@								
D	1101					(	)	_	'								
E	1110					+	;	>	=								
F	1111					|[1]/:[2]	¬[1]/~[2]	?	"								

Notes:

1. Usual EBCDIC character.
2. Printer character.
3. USASCII graphics added to character set.
4. Unassigned codes print as blanks (hexadecimal 40).

Table C-3. IBM BCD Table (6-Bit Representation)[t]

			Most Significant Bits							
	Octal	(columns)	0	1	2	3	4	5	6	7
	(rows)	Binary	000	001	010	011	100	101	110	111
Least Significant Bits	0	000		8		Y	–	Q	&	H
	1	001	1	9	/	Z	J	R	A	I
	2	010	2	0	S		K		B	&
	3	011	3	#	T	,	L	$	C	.
	4	100	4	@	U	%	M	*	D	<
	5	101	5	'	V	_	N	)	E	(
	6	110	6	=	W	>	O	;	F	+
	7	111	7		X	?	P		G	

Note: 0'20' is the official blank character.

[t]Corresponds to BCD code set used by IBM users and defined by PDL command CODE = IBMBCD.

Table C-4. Honeywell 200/2000 BCD Table (6-Bit Representation)[t]

			Most Significant Bits							
	Octal	(columns)	0	1	2	3	4	5	6	7
	(rows)	Binary	000	001	010	011	100	101	110	111
Least Significant Bits	0	000	0	8	+	H	–	Q	<	Y
	1	001	1	9	A	I	J	R	/	Z
	2	010	2	'	B	;	K	#	S	@
	3	011	3	=	C	.	L	$	T	,
	4	100	4	:	D	)	M	*	U	(
	5	101	5		E	%	N	"	V	{
	6	110	6	>	F	[	O	]	W	}
	7	111	7	&	G	?	P	!	X	

Notes: 0'15' is the official blank character; 0'77' is the padding character.

[t]Corresponds to BCD code set used by Honeywell 200/2000 users and defined by PDL command CODE = H2BCD.

Table C-5. Honeywell 6000 BCD Table (6-Bit Representation)[t]

			Most Significant Bits							
	Octal (columns)		ø	1	2	3	4	5	6	7
	(rows)	Binary	øøø	001	010	011	100	101	110	111
Least Significant Bits	ø	øøø	ø	8	space	H	¬ 1 / ^ 2	Q	+	Y
	1	øø1	1	9	A	I	J	R	/	Z
	2	ø1ø	2	[	B	&	K	-	S	_
	3	ø11	3	#	C	.	L	$	T	,
	4	1øø	4	@	D	]	M	*	U	%
	5	1ø1	5	:	E	(	N	)	V	=
	6	11ø	6	>	F	<	O	;	W	"
	7	111	7	?	G	\	P	'	X	!

Notes: 1. Usual BCD Character

2. Printer Character

[t] Corresponds to BCD code set used by Honeywell 600/6000 series SSF tapes and defined by PDL command CODE = BCD or CODE = H6BCD.

Table C-6. Fieldata Translation Table

			Most Significant Bits							
	Octal (columns)		0	1	2	3	4	5	6	7
	(rows)	Binary	000	001	010	011	100	101	110	111
Least Significant Bits	0	000	@	C	K	S	)	*	0	8
	1	001	[	D	L	T	-	(	1	9
	2	010	]	E	M	U	+	%	2	'
	3	011	#	F	N	V	<	:	3	;
	4	100	^	G	O	W	=	?	4	/
	5	101	(blank)	H	P	X	>	!	5	.
	6	110	A	I	Q	Y	&	,	6	"
	7	111	B	J	R	Z	$	\	7	_

Table C-7. Univac ASCII Character Set

Octal	Character	Octal	Character	Octal	Character
040	(blank)	100	@	140	`
041	!	101	A	141	a
042	"	102	B	142	b
043	#	103	C	143	c
044	$	104	D	144	d
045	%	105	E	145	e
046	&	106	F	146	f
047	'	107	G	147	g
050	(	110	H	150	h
051	)	111	I	151	i
052	*	112	J	152	j
053	+	113	K	153	k
054	,	114	L	154	l
055	–	115	M	155	m
056	.	116	N	156	n
057	/	117	O	157	o
060	0	120	P	160	p
061	1	121	Q	161	q
062	2	122	R	162	r
063	3	123	S	163	s
064	4	124	T	164	t
065	5	125	U	165	u
066	6	126	V	166	v
067	7	127	W	167	w
070	8	130	X	170	x
071	9	131	Y	171	y
072	:	132	Z	172	z
073	;	133	[	173	{
074	<	134	\	174	¦
075	=	135	]	175	}
076	>	136	^	176	~
077	?	137	_	177	(null)

INDEX

A

B

C

D

E

F

H

I

J

K

L

M

O

P

R

S

T

U

V

X

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication.

Publication No.	Rev. Letter	Title	Current Date

How did you use this publication?

☐ Learning ☐ Installing ☐ Sales
☐ Reference ☐ Maintaining ☐ Operating

Is the material presented effectively?

☐ Fully Covered ☐ Well Illustrated ☐ Well Organized ☐ Clear

What is your overall rating of this publication?

☐ Very Good ☐ Fair ☐ Very Poor
☐ Good ☐ Poor

What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable.

Your Name & Return Address

2190(12/72)

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

Staple

Staple

Fold

First Class
Permit No. 229
El Segundo,
California

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

Attn: Programming Publications

Fold

XEROX

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

 Printed in U S A. 91 00 02