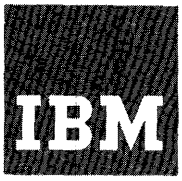
File Number S360(Mod.20)-32
Form C26-3810-3

DPS



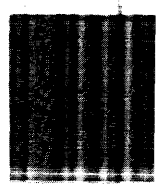
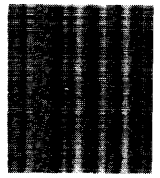
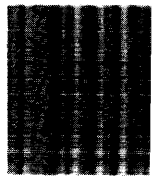
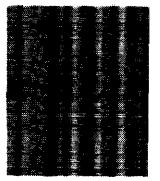
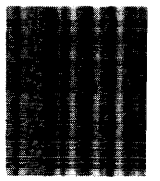
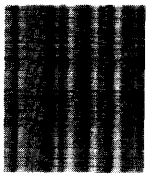
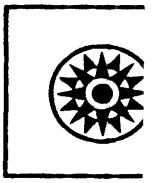
Systems Reference Library

IBM System/360 Model 20 Disk Programming System Disk Utility Programs

This reference publication provides programmers with the information needed in order to use the IBM System/360 Model 20 Disk Utility Programs. The programs described are:

1. Six file-to-file programs for transferring files from input media to output media.
2. A disk initialization program that prepares IBM 1316 Disk Packs for use on IBM 2311 Model 11 or 12 Disk Storage Drives.
3. A program to clear one or more areas of IBM 1316 Disk Packs.
4. A program that establishes an alternate disk track for a defective track, and transfers data from the defective to the alternate track ("Disk Recovery").
5. A disk dump program that prints the contents of data and count fields.

For a list of associated publications and their abstracts, see IBM System/360 Model 20 Bibliography (Form A26-3565).


Preface

This manual describes the IBM System/360 Model 20 Disk Utility programs of the Disk Programming System (DPS). It includes a list of the machine configurations to which the programs apply, and a description of each of the ten programs with specific information about the control statements required to tailor each program to a particular job.

Ample details, technical data, procedures for program modification, and background information are presented in the appendixes.

The programs operate under control of the IBM System/360 Model 20 Disk Programming System control programs. Therefore, you should have a working knowledge of the System/360 Model 20 and be familiar with the publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006. If you wish to include your own routines you should be familiar with the publication IBM System/360 Model 20, Disk and Tape Programming Systems, Assembler Language, Form C24-9002, or IBM System/360 Model 20, Basic Assembler Language, Form C26-3602.

A glossary is appended to explain terminology used in this publication that may be unfamiliar to you.

Fourth Edition (March, 1969)

This is a major revision of, and obsoletes, C26-3810-2.

The changes and amendments consist of the technical changes due to the availability of Model 20, Submodel 5, and the inclusion of a Disk Dump Utility program. Other changes have been made throughout the text and new material, including tables and illustrations, has been incorporated. Therefore, this edition should be reviewed in its entirety.

This edition applies to the following components of IBM System/360 Model 20 and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

	Version	Modification
Disk-to-Disk Utility program	3	0
Disk-to-Tape Utility program	3	0
Tape-to-Disk Utility program	3	0
Disk-to-Card Utility program	3	0
Card-to-Disk Utility program	3	0
Disk-to-Printer Utility program	3	0
Initialize-Disk Utility program	3	0
Clear-Disk Utility program	3	0
Alternate-Track Assignment Utility program	3	0
Disk Dump Utility program	1	0

Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 Model 20 SRL Newsletter, Form N20-0361, for editions that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratories, Programming Publications, 703 Boeblingen/Germany, P.O. Box 210.

Contents

Whatever the specific uses of a data processing system may be, there are certain operations which must be performed frequently. These operations may differ in detail, varying with the particular machine configuration and data format requirements, while the essential functions remain the same. To reprogram these operations each time they are required by a specific and perhaps recurring job would be wasteful, even if advanced languages were used. Therefore, a set of IBM-supplied programs, called the DPS Disk Utility programs, are placed at your disposal to perform these recurring functions and save you performing time. Moreover, the Utility programs are flexible: you can modify them to suit your particular problem with a minimum of programming effort.

There are many standard operations which involve disks, such as reading information from cards and writing it on disk, or printing data stored on disk. Such operations can be performed by the Disk Utility programs. You need only specify certain items of information, such as which card columns are to be copied onto the disk or which portions of a disk should be printed. These specifications you enter in your job-control and utility control statements.

The purpose of this manual is to show you how to use and modify the Disk Utility programs by describing the job-control and utility control statememts and indicating the variety of disk operations these programs can perform.

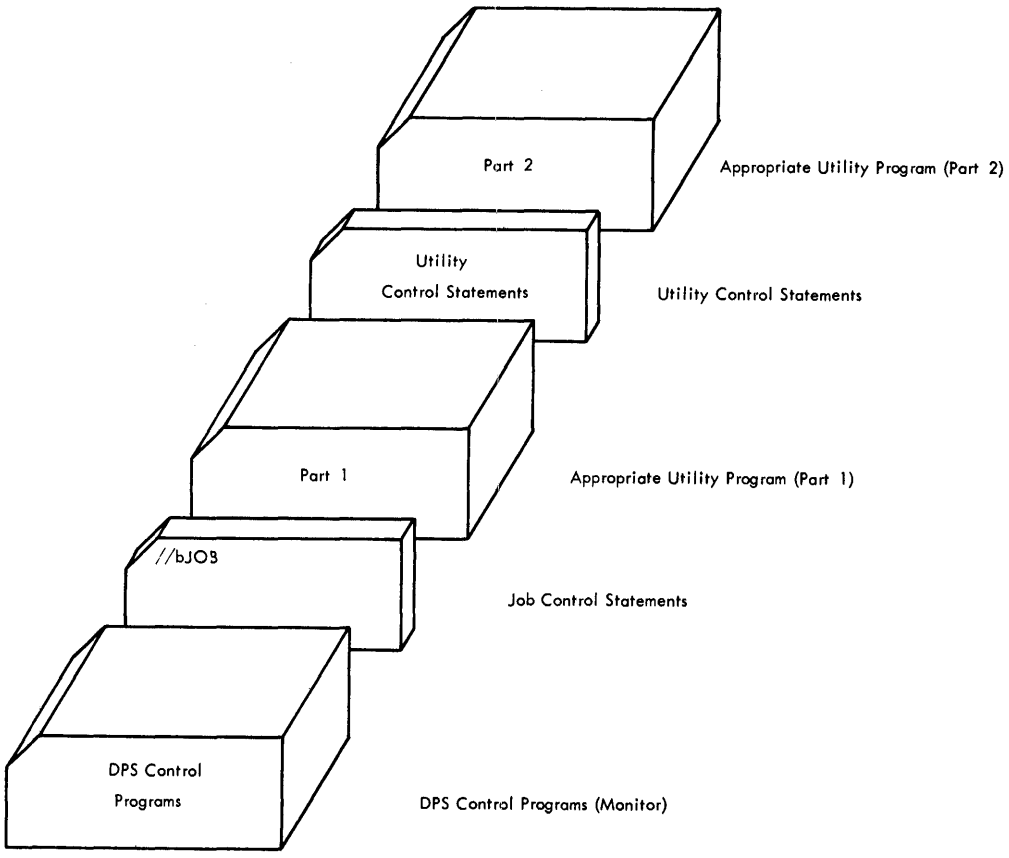


Figure 1. Arrangement of Card Input (Card-Resident System Run)

Each DPS Disk Utility program operation, except the Disk Dump program (which does not need any utility control statements), requires four components.

- DPS control programs, as described in IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006
- Job-control statements
- The DPS Disk Utility program supplied by IBM
- Utility control statements

The utility program may be either a card deck or it may be on the system disk pack. To execute a utility operation, the program must be supplemented by detailed specifications which are contained in the utility control statements. Job-control statements provide information to the DPS Control programs, such as a description of the machine configuration.

If the card-resident version of the DPS Control programs and the DPS Utility programs is used, the DPS Control programs deck is followed by the deck for a single utility program (see Figure 1). If the programs are disk-resident, the Control programs and the DPS Utility programs are stored on the system disk pack (see Figure 2). The appropriate utility program is identified from the job-control statements and is read into main storage.

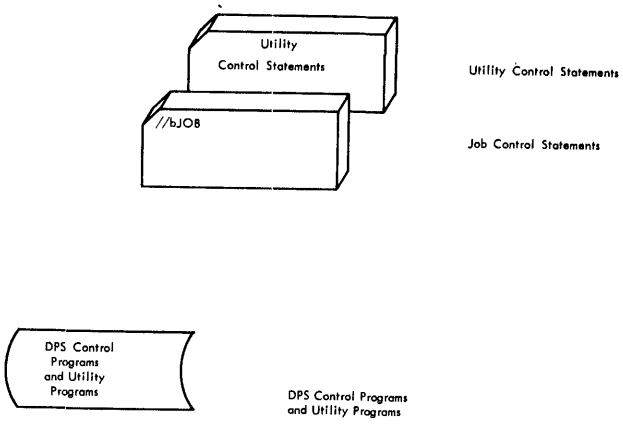


Figure 2. Arrangement of Card Input (Disk-Resident System Run)

DESCRIPTION OF PROGRAMS

There is an important restriction on the use of the six file-to-file utility programs. They cannot process indexed-sequential files. The utility programs do not read or write the labels necessary for indexed-sequential files, nor do they include routines to create indexes. Therefore, the utility programs can be used only with sequentially organized data files.

Each disk utility program performs a particular operation.

1. The Disk-to-Disk program transfers data from disk packs on one drive to disk packs on other drives or between different locations on the same disk pack.
2. The Disk-to-Tape program transfers a data file from one or more disk packs to one or more reels of magnetic tape.
3. The Tape-to-Disk program transfers a data file stored on one or more reels of magnetic tape to one or more disk packs.
4. The Disk-to-Card program punches a data file stored on one or more disk packs into cards.
5. The Card-to-Disk program transfers a data file from punched cards to one or more disk packs.
6. The Disk-to-Printer program prints a data file contained on one or more disk packs.
7. The Initialize-Disk program checks the surfaces of one or more disk packs for defective areas (assigning alternate tracks when appropriate), and formats cylinder 0 and the Volume Table of Contents.
8. The Clear-Disk program clears one or more disk areas by writing a user-specified character in these areas.
9. The Alternate-Track Assignment program pairs an alternate track to a defective track and copies the information stored on the defective track onto the alternate.
10. The Disk Dump program prints data and count fields.

The first six DPS Disk Utility programs listed above can be grouped together as logical file-to-file utility programs because they all transfer data from one storage medium to another.

The last four utility programs, the disk maintenance utilities, do not process data files. However, they are essential for establishing the files and ensuring that all records can be retrieved.

To carry out your specific job, you must modify the program with control information punched into cards. When a choice is not indicated by card, the program assumes a specification called the default specification.

The utility control statements for the file-to-file utility programs are upward compatible with the IBM System/360 Basic Programming Support and Basic Operating System Disk Utility programs when all relevant operands are explicitly stated. IBM System/360 Model 20 standards are used for disk labels and IBM System/360 standards for tape labels.

MINIMUM SYSTEM CONFIGURATION

Submodel 2

An IBM 2020 Central Processing Unit Model BC2 (12,288 bytes of main storage);

an IBM 2311 Disk Storage Drive Model 11 or 12;

one of the following card reading devices:

IBM 2501 Card Reader Model A1 or A2,
IBM 2520 Card Read-Punch Model A1,
IBM 2560 Multi-Function Card Machine (MFCM) Model A1;

one of the following printers:

IBM 1403 Printer Model N1, 2, or 7,
IBM 2203 Printer Model A1.

Submodel 4

An IBM 2020 Central Processing Unit Model BC4 (12,288 bytes of main storage);

an IBM 2311 Disk Storage Drive Model 12;

an IBM 2560 MFCM Model A2;

an IBM 2203 Printer Model A2.

Submodel 5

An IBM 2020 Central Processing Unit Model BC5 (12,288 bytes of main storage);

an IBM Disk Storage Drive Model 11 or 12;

one of the following card reading devices:

IBM 2501 Card Reader Model A1 or A2,
IBM 2520 Card Read-Punch Model A1,
IBM 2560 Multi-Function Card Machine (MFCM) Model A1;

one of the following printers:

IBM 1403 Printer Model N1, 2, or 7,
IBM 2203 Printer Model A1.

MAXIMUM SYSTEM CONFIGURATION

Submodel 2

An IBM 2020 Central Processing Unit Model D2 (16,384 bytes of main storage);

two IBM 2311 Disk Storage Drives Model 11 or 12 (both must be the same model);

an IBM 2415 Magnetic Tape Unit Model 1 through 6;

an IBM 2501 Card Reader Model A1 or A2;

an IBM 1442 Card Punch Model 5;

one of the following card units:

IBM 2520 Card Read-Punch Model A1,
IBM 2520 Card Punch Model A2 or A3,
IBM 2560 MFCM Model A1;

one of the following printers:

IBM 1403 Printer Model N1, 2, or 7,
IBM 2203 Printer Model A1;

an IBM 2152 Printer-Keyboard (can be used only for inquiries, not as normal I/O unit).

Submodel 4

An IBM 2020 Central Processing Unit Model D4 (16,384 bytes of main storage);

two IBM 2311 Disk Storage Drives Model 12;

an IBM 2560 MFCM Model A2;

an IBM 2203 Printer Model A2.

an IBM 2152 Printer-Keyboard (can be used only for inquiries, not as normal I/O unit).

Submodel 5

An IBM Central Processing Unit Model E5 (32,768 bytes of main storage);

four IBM 2311 Disk Storage Drives Model 11 or 12;

an IBM 2415 Magnetic Tape Unit Model 1 through 6;

an IBM 2501 Card Reader Model A1 or A2;

an IBM 1442 Card Punch Model 5;

one of the following card units:

IBM 2520 Card Read-Punch Model A1,
IBM 2520 Card Punch Model A2 or A3,
IBM 2560 MFCM Model A1;

one of the following printers:

IBM 1403 Printer Model N1, 2, or 7,
IBM 2203 Printer Model A1;

an IBM 2152 Printer-Keyboard (can be used only for inquiries, not as normal I/O unit).

PROGRAM FEATURES

The DPS Disk Utility programs process fixed-length records (records of equal length) of sequentially organized files on disk or tape. These records may be blocked or unblocked. If they are blocked, all blocks must contain the same number of records. (See exception in Tape-to-Disk program.) Blocking is a method of file organization whereby a number of records on a disk pack or on magnetic tape is grouped together for more efficient data handling. Appendixes E and F discuss record and block formats in detail.

The DPS Disk Utility programs perform a number of valuable functions for you:

- They transfer data files between or within media in several ways; e.g., from disk to magnetic tape, from one disk pack to another, or from one area of a disk pack to another area on the same disk pack. They can copy records without change. They can also rearrange or delete data within each record and change the data format. They will change the number of records in each block if you wish.
- They automatically process a data file which is stored on more than one disk pack or more than one reel of magnetic tape. This is called multi-volume processing. (For further details refer to Appendix J).
- If required they can check the sequence of input cards and number output cards consecutively.
- They can read data files from cards in standard EBCDIC format (one character per card column), or in column binary format (see Glossary). But a card data file may be punched only in the card code equivalent of the EBCDIC format.
- They represent numeric information either in zoned-decimal format with one digit per byte, or in packed-decimal format with two digits per byte. They may also change the input data format for output.
- They print each output record either as a whole (display format), or in parts (list format).
- A byte can be printed in one of two forms: (1) as a single character representing the EBCDIC graphic for the bit configuration; or (2) by two characters, one for the left half-byte (four bits), and one for the right half-byte. The possible half-byte values, which range from 0-15, are represented by the characters 0-9 and A-F. This latter type of representation is called hexadecimal.
- The Disk Dump program serves as a good debugging aid in that it prints the contents of any data or count field(s).
- The programs process seven-track as well as nine-track magnetic tapes.
- They process IBM standard volume and file labels for both disk and tape. If you want to use your own standard tape file labels, you must also supply the appropriate routines to process these labels. Special exits are provided to link up with your routines.
- Exits are also available for use with special routines for sterling-currency conversion in the Disk-to-Card, Card-to-Disk, and Disk-to-Printer programs.
- The programs also process unlabeled tape files. Several unlabeled tape files on one tape volume (multi-file-volume) are always separated by one tapemark. A tapemark preceding the file at the beginning of the tape reel may be present but is not required.
- The utility programs may be executed as either mainline or inquiry programs. An exception is the Initialize Disk program, which can be used only as mainline program. Except the Alternate-Track Assignment and the Disk Dump program, all utility programs used as mainline programs may be interrupted by inquiry programs.
- The file-to-file utility programs normally process a file until an end-of-file indicator, /*, is encountered. To ignore this indicator, a UPSI statement may be included in the job-control statements which allows the entire specified extent or volume to be processed.

  Note: The block length in the file to be transferred must not exceed the size of the extent specified in the XTENT statement. Otherwise, this extent will not be processed.

ORGANIZATION OF THE PUBLICATION

The manual describes each of the ten DPS Utility programs and the utility control statements you must provide to tailor the program to your specific task. The common features of utility control statements are discussed first. Building on the discussion of the general control statements, the manual continues with ten sections, each containing a description of the capabilities and use of each utility program.

These utility programs operate under control of the Model 20 DPS Control programs. The loading and execution of a utility program is requested by means of job-control statements. The job-control statements required for each of the DPS Disk Utility programs are described in Appendix A. The function of each job-control statement and appropriate entries for it are discussed in greater detail in the publication IBM System/360 Model 20, Disk Programming System, Control and Service Programs, Form C24-9006.

The appendixes also contain more detailed discussions of various program features and technical points.

General Description of Utility Control Statements

There are two utility control statements that can be used with each of the six logical file-to-file utility programs: the utility-modifier statement and the field-select statement. The utility-modifier statement outlines the method of transferring a set of data records, called a file, from one storage medium (disk, magnetic tape, or cards) to another (disk, magnetic tape, cards, or printer). The field-select statement describes the transfer in more detail. A third utility control statement, the print-header statement is used with the Disk-to-Printer program and is discussed in the section dealing with that program. In addition, the END statement is required as the last utility control statement in all programs except the Disk Dump program. If any of the other control statements is not present, the program assumes certain standard specifications (called default specifications).

The Initialize Disk, Clear Disk, and Alternate-Track Assignment Utility programs require utility control statements of their own. The Disk Dump program does not require any utility control statements.

Throughout this manual, the character "b" will be used to designate a blank column. One, and only one, blank column must be left between the name and the operation fields, and between the operation and the operand fields.

A group of characters set off by a space or comma is called an operand; e.g., A=(input),B=(output). No blank spaces are allowed within the operand list. The last operand of a control statement is followed by a space instead of a comma. The lowercase letter or letters in each operand indicate(s) that you have to assign the appropriate value(s).

UTILITY-MODIFIER STATEMENT

The utility-modifier statement outlines the job. It

- indicates the type of transfer;
- describes the input and output files;
- controls certain input and output device actions;
- specifies incidental functions (page numbering, sequence checking).

The general format of the utility-modifier statement is

```
┌────┬─────────┬──────────────────────────┐
│Name│Operation│Operand                   │
├────┼─────────┼──────────────────────────┤
│//b │Uxxb     │Tt,FF,A=(input),          │
│    │         │  B=(output),Ix,Ox,Sx,Px, │
│    │         │  Rx,Q=(x,y)              │
└────┴─────────┴──────────────────────────┘
```

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

xx Program initials to designate the particular Disk Utility program:

DD = disk-to-disk
DT = disk-to-tape
TD = tape-to-disk
DC = disk-to-card
CD = card-to-disk
DP = disk-to-printer

The four required operands are:

Tt - type-of-function operand
FF - record-format operand

A = (input) - input record length and block length operand
B = (output) - output record length and block length operand

These required operands (Tt through B=(output)) must be specified in the utility-modifier statement when it is present. The remaining operands (Ix through Q=(x,y)) are optional, and may appear in any order after the four which are required. The mandatory operands and the optional entries, which have somewhat different meanings in different utility programs, are described separately with each program.

When any optional operand or the utility-modifier statement itself is omitted from a Disk Utility program, the program automatically assumes specifications (called "default specifications") for the missing operand(s). If all the default specifications correspond to the specifications required for a particular job, the program may be run with no utility control statement but the END statement. The default specifications for each of the six file-to-file Disk Utility programs are listed in Figure 3.

	Tt	FF	A=(input)	B=(output)	Ix	Ox	Sx	Px	Rx	Q=(x,y)
Disk-to-disk	TC	FF	A=(270,270)	B=(270,270)	-	OY	-	-	-	-
Disk-to-tape	TC	FF	A=(270,270)	B=(270,270)	-	OU	-	-	-	-
Tape-to-disk	TC	FF	A=(270,270)	B=(270,270)	IU	OY	-	-	-	-
Disk-to-card	TR	FF	A=(80,240)	B=(80,80)	-	-	-	-	-	(omit)
Card-to-disk	TR	FF	A=(80,80)	B=(80,240)	I1	OY	-	-	-	(omit)
Disk-to-printer	TD	FF	A=(270,270)	B=(120)	-	OX	S1	PY	R1	-

Figure 3. Utility-Modifier Statement Default Specifications

FIELD-SELECT STATEMENT

The field-select statement can be used only if the type-of-function operand (Tt) of the utility-modifier statement contains the specification TF (field-select), TRF (reblock and field-select), or TLF (list and field-select). The field-select option allows:

- movement of input record data fields to different relative positions in the output record;
- omission of input data fields from output records;
- conversion of data format, by field.

When the field-select option is specified, each data field that is to be included in the output is transferred from input to output area in one of four ways:

- It is moved with no change in field length, or
- moved and converted from zoned to packed decimal (the output field is normally shorter than the input field), or
- moved and converted from packed to zoned decimal (the output field is normally longer than the input field), or
- moved and converted from zoned or packed decimal to hexadecimal characters (the output field is twice as long as the input field; this option is used only by the Disk-to-Printer program).

When the field-select option is used, only those bytes in the input record that are specified in the field-select statement will be moved. The rest will be omitted from the output record. As a result of dropping fields and/or converting the data format, output record length may be different from input length.

If fields are specified to overlap, data in a field moved to the output area will replace any data previously moved to the same positions. If fields overlap, a warning message will be printed when the field-select statement is read.

Data is moved in the sequence in which the entries in the field-select statements appear, i.e., the first operand is processed first, the second operand next, etc. Areas in the output record not filled by selected data are blank.

Entries can extend to column 80 of a field-select (FS) statement. The information for a particular field, however, must be completed in one card. A continuation card is not allowed. Instead, several FS statements may be used. Blank columns at the end of a card are permitted.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

The field operands are separated by slashes; however, the last operand on a statement must not be followed by a slash. The field operands may occur in any order, so long as each is defined correctly, relative to the first byte of the record.

When a data field is to be moved and converted (see Appendix C), the operand for that field assumes a slightly different form. The entries r and t retain the meaning stated above. The entry s, indicating the length of a field, is replaced by an entry defining the conversion to be done and the length of both input and output fields. The change in record size must be considered when filling out the utility-modifier statement.

In the unpacked or zoned-decimal format, numbers are stored one per byte and the sign is in the left four bits (half-byte) of the rightmost byte of the field. In the packed mode, numeric fields are stored with two digits per byte. An extra half-byte is now needed to store the sign. For example, a four-digit unpacked number would need only five half-bytes when packed. Actually, three bytes must be used, since all fields must consist of full bytes. The sign in packed-decimal format is stored in the rightmost half-byte of the field. Note that System/360 requires a sign position regardless of whether numeric values are positive or negative.

Pack

When the input field is to be packed for output (in the case of decimal arithmetic, e.g.,), the operand entry is

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P identifies the pack operation

n size in bytes of the input field (unpacked)

m size in bytes of the output field (packed)

t starting position in the output record of the receiving field

The parentheses and commas must appear as shown.

The following formulas may be used to determine the number of bytes required in the output field:

$$\text{If n is odd, } m=\frac{n+1}{2}$$

$$\text{If n is even, } m=\frac{n+2}{2}$$

For instance, if n=4 or 5, m=3; if n=6 or 7, m=4; etc. However, if you want to make the packed field larger than required, m may be a value greater than necessary to accommodate the input field after packing. In this case, the excess leftmost positions of the packed field will be filled with zeros. An extra zero also is placed in the leftmost half-byte whenever a zoned-decimal number of an even number of bytes is packed. If not enough space is provided in the output field, the most significant (leftmost) digits will be lost. A warning message will then be printed.

Unpack

When the input field is to be converted from packed to zoned decimal for output (for printing, e.g.), the operand entry is

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U identifies the unpack operation

n size in bytes of the input field (packed)

m size in bytes of the output field (unpacked)

t starting position in the output record of the receiving field

The parentheses and commas must appear as shown.

To ensure that the entire input field will fit into the output field, the size in bytes of the unpacked output filed must be twice the size of the input field, minus one: m=2n-1. For example, if n=3, m=5; if n=4, m=7; etc. Of course, if you know that the leftmost (high-order) half-byte (four bits) in the packed field is zero, you can use the formula m=2n-2 to eliminate the high-order zero. You may specify an output field that is larger than required; the excess high-order bytes will then contain zoned zero values. If not enough space is provided in the output field, the most significant (high-order) digits will be lost. A warning message will then be printed.

Hexadecimal (Disk-to-Printer Program Only)

The utility-modifier statement for the Disk-to-Printer program includes the option to present the entire record in hexadecimal format. For a detailed description of the hexadecimal option refer to the section Disk-to-Printer Utility Program.

Sample Field-Select Problem

A disk file contains four data fields in each record.

- Bytes 1 to 5 contain a packed field.
- Bytes 6 to 10 contain a zoned-decimal field.
- Bytes 11 to 12 contain a zoned-decimal field.
- Bytes 13 to 30 contain an alphameric field.

Output cards are to be created as follows:

Columns
- 1-18 the alphameric field (Field 4)
- 19-31 blank
- 32-40 Field 1, unpacked
- 41-46 blank
- 47-49 Field 2, packed
- 50-80 blank

The two layouts are illustrated in Figure 4.

Two utility control statements (besides the END statement) are required for this job - a utility-modifier statement and a field-select statement. They are shown in Figure 5.

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

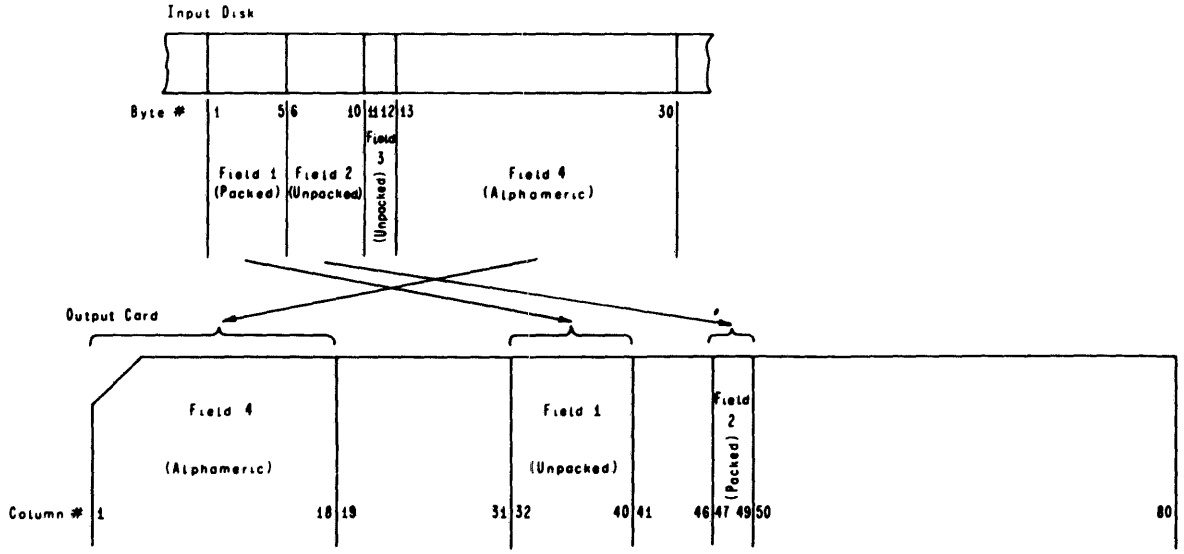
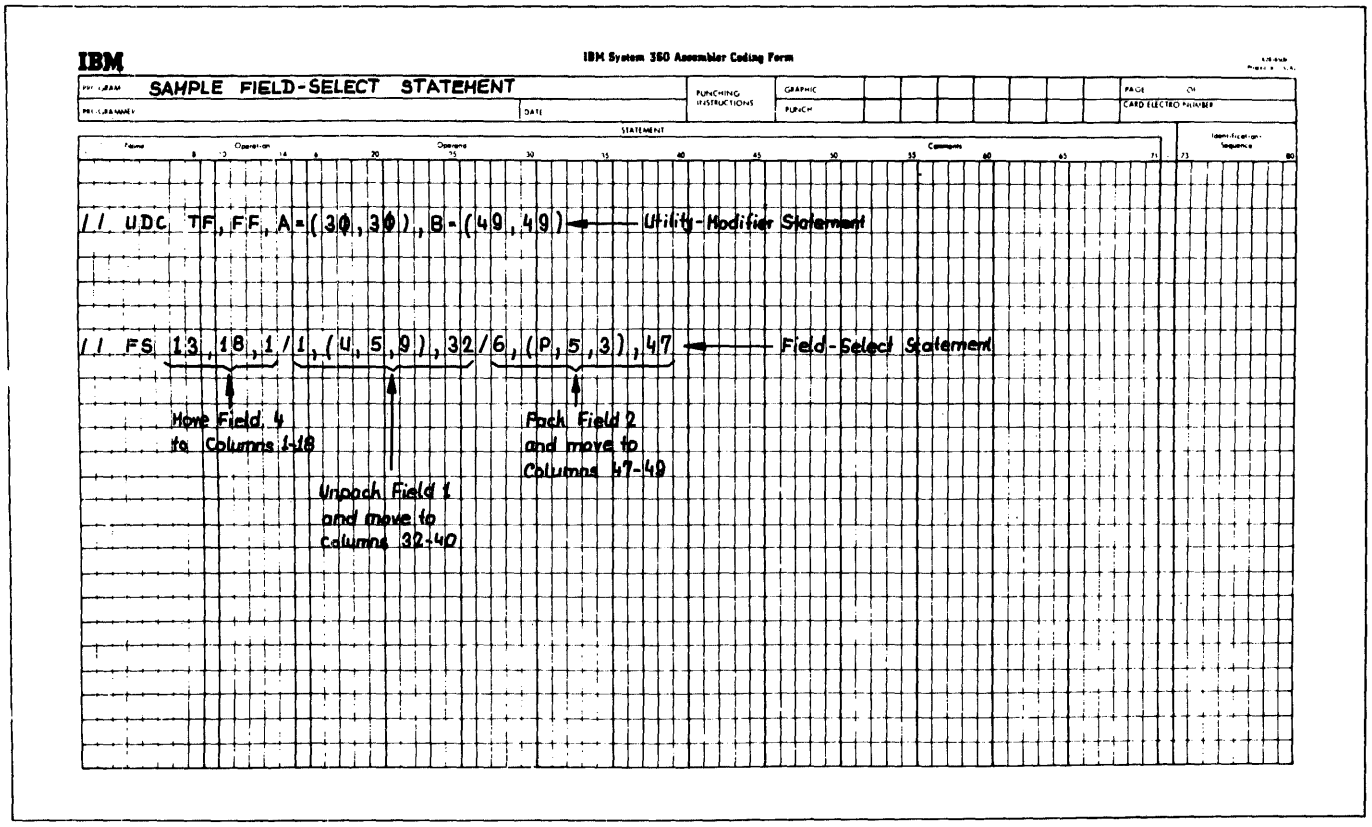


Figure 4. Sample Field-Select Problem



● Figure 5. Sample Field-Select Problem Control Statements

The Disk-to-Disk utility program transfers a file from one disk pack to another (max. two drives for Submodels 2 and 4 and four drives for Submodel 5), or between areas (called extents) of the same disk pack. (Using different cylinders of the same pack for input and output will reduce the efficiency of the program).

All input and output records must be fixed-length, blocked or unblocked.

It is possible to copy the contents of an entire disk pack by using the Disk-to-Disk utility program with a special UPSI statement. This feature is further explained in Appendix I.

The following job-control and utility control statements are used by the Disk-to-Disk utility program.

Job-Control Statements

Statement	Use	Operand Entries for Disk-to-Disk Utility
// JOB	Required	Program name = DSKDSK
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Alternate input devices = SYS002-SYS004 Alternate output devices SYS005-SYS007
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// UPSI	Optional	
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	Input file name = UIN Output file name = UOUT
// DLAB	Required	One set each per input and output file
// XTENT	Required	More than one XTENT statement can be used per set
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

*Unless information is still in main storage.

A detailed description of the job-control statements is given in Appendix A.

Utility Control Statements

Two utility control statements may be used with the Disk-to-Disk program:

The utility-modifier statement, which outlines the method of transferring a set of data records (a file) from disk to disk.

The field-select statement, which describes the transfer of fields from the input record to the output record.

A third utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bEND	Required

DISK-TO-DISK UTILITY-MODIFIER STATEMENT

The utility-modifier statement controls the file transfer by indicating the type of transfer, describing the input and output files, and specifying certain input and output device actions. Its general format is

Name	Operation	Operand
//b	UDDb	Tt,FF,A=(input),B=(output),Ox

Possible operand entries are shown in Figure 6.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

DD Program initials for the Disk-to-Disk program. They may be omitted since the JOB statement actually specifies a disk-to-disk program (Program name = DSKDSK). If used at all, the initials must be DD; otherwise a warning message will be printed.

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (DD) are used, or in column 6 if DD is omitted. These four operands must be listed in the above order, and all must appear unless the card is omitted entirely. The additional operand Ox may appear after the four required ones.

Tt - Type-of-Function Operand

This operand specifies the transfer to be performed, when "t" is replaced by one or two initials. Files can be transferred in four ways:

- TC -- Copy. No change is made in the records or the file itself. This option is used to produce an identical file.
- TR -- Reblock. The number of records per block in the output file is different from that of the input file. The record format remains unchanged.
- TF -- Field-Select. The record format is changed. Data fields within each record are rearranged, omitted, or converted to zoned or packed decimal. The output record length may differ from the input record length; however, the number of records per block remains the same as in the input file.
- TRF -- Reblock and Field-Select. This is a combination of the two preceding options. It is used when both the record format and the number of records per block are to be changed.

FF - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e., all records contain the same number of bytes. The characters FF must be entered.

A=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read. The format of this operand is

A=(n,m), where

n is the number of bytes in each record, and

m is the number of bytes in each block.

Since the number of records per block is constant, m can be worked out by multiplying the number of bytes per record by the number of records per block. The values n and m must be separated by a comma and enclosed in parentheses.

B= (output) - Output Record Length and Block Length Operand

This operand describes the records and blocks of the output file. Its format is

B=(a,c), where

a is the number of bytes in each record, and

c is the number of bytes in each block.

The value c equals the number of bytes per record multiplied by the number of records per block. The values a and c must be separated by a comma and enclosed in parentheses.

Ox - Write-Disk Check

Ox is an optional operand that may be placed behind the four required operands. If it is omitted, the default specification (OY) in Figure 6 is assumed and a write-disk check is made. The write-disk check option is used to read and check data immediately after it has been written on disk. If the data cannot be written properly, a message is printed and the program halts. It is recommended that the write-disk check option should be used.

If the optional operand (Ox) or the utility-modifier statement itself is omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s).

You can omit the entire Disk-to-Disk utility-modifier statement (and run your program with the END statement only), if all your requirements match the default specifications as shown in Figure 6:

- Copy only, no field-selection or reblocking.
- Fixed-length records.
- Input record and block lengths are both 270 bytes.
- Output record and block lengths are both 270 bytes.
- Write-disk check is to be made.

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Disk-to-Disk program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

Operand	Possible Forms	Operand Specification	Explanation	Default Specification	

Statement ID
//bUxxb
	//bUDDb
	
	//bUb
		DD	Disk-to-Disk utility identifier
			General utility identifier
				//bUb	

Type of Function Tt	TC TF TR TRF	C F R RF	Copy Field-Select Reblock Reblock and Field-Select	TC	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,m)	 n m	Fixed-length records Input record length, Input block length	 A=(270,270)	*
Output Description B=(output)	B=(a,c)	 a c	Fixed-length records Output record length, Output block length	 B=(270,270)	*
Disk Check Ox	OY ON	Y N	Write-disk check Do not write-disk check	OY	

*Default permitted only if entire statement is omitted.

```
//bUDDbTt,FF,A=(input),B=(output),Ox
```

Figure 6. Disk-to-Disk Program Utility-Modifier Statement

The FS statement is only required if TF or TRF is specified in the Tt operand of the Disk-to-Disk utility-modifier statement.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

$$\text{If n is odd, } m = \frac{n+1}{2}$$

$$\text{If n is even, } m = \frac{n+2}{2}$$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

Sample Problems

1. Copy data from one disk pack to another pack. Both the input and the output packs contain eighty-byte records, three to a block. Do not make a write-disk check.

```
//bUDDbTC,FF,A=(80,240),B=(80,240),ON
//bEND
```

No field-select statement is needed. The control-statement identification field for a Disk-to-Disk utility program is //bUDDb, the DD entry being optional. The job is to copy (TC) fixed-length records (FF). The size of the input and output blocks is calculated by multiplying the number of bytes per record by the number of records per block (80x3=240).

2. Transfer a file from one area of a disk pack to another area of the same pack. The input records are unblocked, eighty bytes long. Output records are to be blocked six records to a block. Make a write-disk check.

```
//bUDDbTR,FF,A=(80,80),B=(80,480),OY
//bEND
```

or

```
//bUbTR,FF,A=(80,80),B=(80,480)
//END
```

Both pairs of statements are acceptable since the program assumes a default value of OY and the DD code for Disk-to-Disk is optional. No field-select statement is needed. Since the number of records per block is changed, the reblock transfer (TR) is used. Block size is calculated by multiplying the number of bytes per record by the number of records per block. For the input blocks this is 80x1=80 bytes. For output blocks it is 80x6=480 bytes.

Note that the utility control statements do not indicate whether the transfer is between different disk packs or between areas of the same pack. This assignment is made by means of job-control statements.

3. Transfer a file from one disk pack to another. Make a write-disk check. The input file was written with 76-byte records, three to a block. Three fields in each record are to be packed and the output blocks are to contain eight records, each 67 bytes long. The record formats are shown in Figure 7 and in the following table:

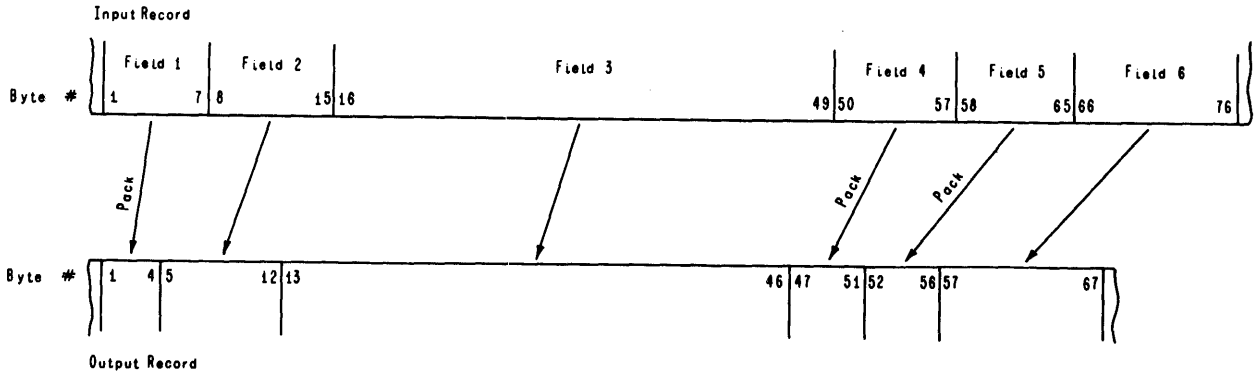


Figure 7. Record Formats for Disk-to-Disk Transfer Problem

Field	Input Record bytes	Output Record bytes	Conversion
1	1-7	1-4	To be packed
2	8-15	5-12	Unchanged
3	16-49	13-46	Unchanged
4	50-57	47-51	To be packed
5	58-65	52-56	To be packed
6	66-76	57-67	Unchanged

The required utility control statements are:

```
//bUDDbTRF,FF,A=(76,228),B=(67,536),OY
//bFSb1,(P,7,4),1/8,8,5/16,34,13
//bFSb50,(P,8,5),47/58,(P,8,5),52/66,11,57
//bEND
```

The job requires the block size to be changed from three records per block in the input file to eight records per block in the output file. At the same time the record format is changed: three selected fields are to be converted to packed decimal, the remaining three are moved to different relative positions in the output record. Hence the reblock and field-select option (TRF) is used.

Input records are 76 bytes long and the blocks are 76x3=228 bytes long. The output records are 67 bytes long and the blocks are 67x8=536 bytes long. The optional operand OY could be omitted, since the program would automatically assume a default specification of OY.

The field-select statement controls the change in record format. Since field 1 is to be packed for output, its operand in the field-select statement is r,(P,n,m),t. Field 1 starts at byte 1 of the input record (r=1); it is 7 bytes long in the input record (n=7) and must be at least (7+1)/2=4 bytes long in the output record (m=4). Its starting position in the output record is byte 1 (t=1).

The second field operand is separated from the first by a slash (/). Since no conversion is required, the operand assumes the form r,s,t. The starting point of field 2 in the input record is byte 8 (r=8). Field length is 8 (s=8); transfer is to byte 5 of the output record (t=5).

Fields 3 and 6 are also moved without change. Fields 4 and 5 are eight bytes long for input and must each be at least (8+2)/2 = 5 bytes long in the output record. (The formulas for computing packed-field lengths are presented in the section Field-Select Statement).

One FS statement could have accommodated all field-select specifications. The second statement was used for illustrative purposes only. Note, also, the absence of a slash (/) after the final operand on each of the FS statements.

4. Transfer a file from one disk pack to another pack. The input and output files contain unblocked 270-byte records. Write-disk check.

The only utility control statement required is

```
//bEND
```

No utility-modifier or field-select statement is required. The default specifications satisfy the job (see Figure 6).

Disk-to-Tape Utility Program (DSKTAP)

The Disk-to-Tape utility program transfers a data file from one or more disk packs mounted on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5), to one or more reels of tape.

All input and output records must be fixed-length, blocked or unblocked.

It is possible to transfer the contents of an entire disk pack to a magnetic tape by using the Disk-to-Tape utility program with a special UPSI statement. This feature is further explained in Appendix I.

The following job-control and utility control statements are used by the Disk-to-Tape utility program.

Job-Control Statements

Statement	Use	Operand Entries for Disk-to-Tape Utility
// JOB	Required	Program name = DSKTAP
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Alternate output device = SYS001
// ASSGN	Optional	Alternate input devices = SYS002-SYS004
// ASSGN	Optional	Logging unit = SYSLOG (See LOG statement)
// FILES	Optional	
// UPSI	Optional	
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	Input file name = UIN
// DLAB	Required	One set for input file
// XTENT	Required	More than one XTENT statement can be used in the set
// VOL	Required	If label processing on output tape Output file name = UOUT
// TPLAB	Required	
// LOG or NOLOG	Optional	
// EXEC	Required	

*Unless information is still in main storage.

For a detailed description of the job-control statements refer to Appendix A.

Note: When the Disk-to-Tape utility program is used as inquiry program, the tape files are treated as unlabeled files and tape labels are not processed. If you have issued a TPLAB statement, it is ignored.

Tapemark Option

If for an unlabeled output tape a tapemark is not desired at the beginning of the reel (Disk-to-Tape only), an UPSI statement must be provided with the job-control statements. The UPSI statement must have the following format:

```
//bUPSIbnnnn1nnn
```

where n represents any non-blank character.

Utility Control Statements

Two utility control statements may be used with the Disk-to-Tape program:

The utility-modifier statement, which outlines the method of transferring a set of data records (a file) from disk to tape.

The field-select statement, which describes the transfer of individual fields from the input record to the output record.

A third utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bEND	Required

DISK-TO-TAPE UTILITY-MODIFIER STATEMENT

The utility-modifier statement controls the file transfer by indicating the type of transfer, describing the input and output files, and specifying certain input and output device actions. Its general format is

Name	Operation	Operand
//b	UDTb	Tt,FF,A=(input),B=(output),Ox

Possible operand entries are shown in Figure 8.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

DT Program initials for the Disk-to-Tape program. They may be omitted since the JOB statement actually specifies a disk-to-tape program (Program name = DSKTAP). If used at all, the initials must be DT; otherwise a warning message will be printed.

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (DT) are used, or in column 6 if DT is omitted. These four operands must be listed in the above order, and all must appear unless the card is omitted entirely. The additional operand Ox may appear after the four required ones.

Tt - Type-of-Function Operand

This operand specifies the transfer to be performed, when "t" is replaced by one or two initials. Files can be transferred in four ways:

- TC -- Copy. No change is made in the records or the file itself. This option is used to produce an identical file.
- TR -- Reblock. The number of records per block on the output tape is different from that in the input disk file. The record format remains unchanged.
- TF -- Field-Select. The record format is changed. Data fields within each record are rearranged, omitted, or converted to zoned or packed decimal. The output record length may differ from the input record length; however, the number of records per block remains the same as in the input file.
- TRF -- Reblock and Field-Select. This is a combination of the two preceding options. It is used when both the record format and the number of records per block are to be changed.

FF - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e., all records contain the same number of bytes. The characters FF must be entered.

A=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read. The format of this operand is

A=(n,m), where

n is the number of bytes in each record, and

m is the number of bytes in each block.

Since the number of records per block is constant, m can be worked out by multiplying the number of bytes per record by the number of records per block. The values n and m must be separated by a comma and enclosed in parentheses.

B=(output) - Output Record Length and Block Length Operand

This operand describes the records and blocks of the output file. Its format is

B=(a,c), where

a is the number of bytes in each record, and

c is the number of bytes in each block.

The value c equals the number of bytes per record multiplied by the number of records per block. The values a and c must be separated by a comma and enclosed in parentheses.

Ox - Rewind Tape

Ox is an optional operand you may specify after the four required operands. If omitted, its default specification in Figure 8 (OU) is assumed. The rewind-tape option determines whether the last or only output tape is to be rewound (OR), rewound and unloaded (OU), or not rewound (ON) after use. If there is more than one output tape, all but the last are rewound and unloaded after use. Output tapes are not rewound before use.

If the optional operand (Ox) or the utility-modifier statement itself is omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s).

Operand	Possible Forms	Operand Specification	Explanation	Default Specification	

Statement ID
//bUxxb
	//bUDTb
	
	//bUb
		DT	Disk-to-Tape utility identifier
			General utility identifier
				//bUb	

Type of Function Tt	TC TF TR TRF	C F R RF	Copy Field-Select Reblock Reblock and Field-Select	TC	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,m)	 n m	Fixed-length records Input record length, Input block length	 A=(270,270)	*
Output Description B=(output)	B=(a,c)	 a c	Fixed-length records Output record length, Output block length	 B=(270,270)	*
Rewind Output Tape Ox	ON OR OU	N R U	Do not rewind Rewind Rewind and unload	OU	

*Default permitted only if entire statement is omitted.

```
//bUDTbTt,FF,A=(input),B=(output),Ox
```

Figure 8. Disk-to-Tape Program Utility-Modifier Statement

You can omit the entire Disk-to-Tape utility-modifier statement (and run your program with the END statement only), if all your requirements match the default specifications as shown in Figure 8:

- Copy only, no field-selection or reblocking.
- Fixed-length records.
- Input record and block lengths are both 270 bytes.
- Output record and block lengths are both 270 bytes.
- Rewind and unload output tape.

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Disk-to-Tape program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

The FS statement is only required if TF or TRF is specified in the Tt operand of the Disk-to-Tape utility-modifier statement.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

$$\text{If n is odd, } m = \frac{n+1}{2}$$

$$\text{If n is even, } m = \frac{n+2}{2}$$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

Sample Problems

1. Copy a file from a disk pack to a magnetic tape. Both the input and the output medium contain 85-byte records, three to a block. Do not rewind the output tape.

```
//bUDTbTC,FF,A=(85,255),B=(85,255),ON
//bEND
```

The control statement identification for Disk-to-Tape is //bUDTb, the DT entry being optional. The job is to copy (TC) fixed-length records (FF). The block length is calculated by multiplying the number of bytes in a record by the number of records in a block (85x3=255). No field-select statement is needed since the job is to copy records.

2. Transfer a file from disk to tape. Rewind and unload the tape. The input records are 80 bytes long and there are three records in a block. On the output tape, there are to be ten records of 80 bytes each to a block.

```
//bUDTbTR,FF,A=(80,240),B=(80,800),OU
//bEND
```

or

```
//bUbTR,FF,A=(80,240),B=(80,800)
//bEND
```

Both sets of statements achieve the same result since the DT code for Disk-to-Tape is optional and OU is the default specification of the tape-rewind option. Since the block size of the output file is to be different from that of the input file, the reblock option (TR) must be used. The records themselves are copied unchanged; therefore no field-select statement is needed. The block sizes are calculated by multiplying the record size by the number of records in a block (80x3=240 and 80x10=800).

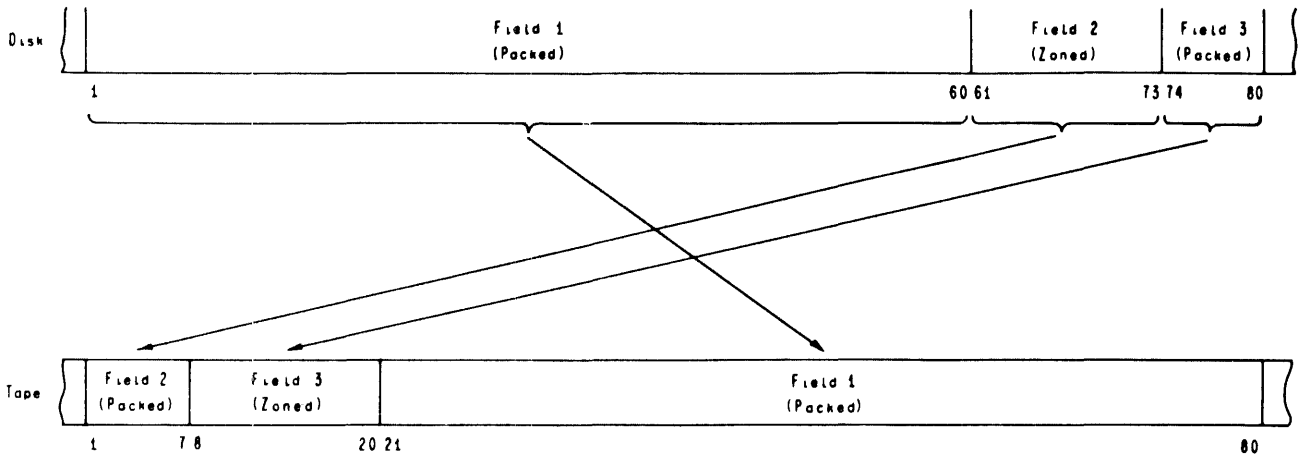


Figure 9. Field Modification for Disk-to-Tape Transfer Problem

3. Transfer a file from disk to tape, changing the format of the records. Rewind, but do not unload, the tape at the end of the job. The input records are 80 bytes long with three records to a block. The output record and block size will be the same. Both input and output records contain three fields.

```
//bUDTbTF,FF,A=(80,240),B=(80,240),OR
//bFSb1,60,21/61,(P,13,7),1/74,(U,7,13),8
//bEND
```

Figure 9 shows how the transfer is made. Further specifications are listed in Figure 11.

Since the record format is to be changed, the field-select statement is needed; the Tt operand in the utility-modifier statement becomes TF. The operands of the field-select statement describe the format change for each field in detail.

Since no conversion is required for field 1, its general operand entry is r,s,t. The field is moved from bytes 1 - 60 (starting point r=1, field length s=60) to bytes 21 - 80 of the output record (t=21).

The field operands are separated from each other by a slash (/). Field 2 must be converted from zoned to packed decimal format; its operand entry must therefore indicate the pack operation: r,(P,n,m),t. Field 2 starts at byte 61 of the input record (r=61) with a length of 13 bytes (n=13); since this is an odd number, the length of the field in the output record must be equal to or greater than (13+1)/2=7 after conversion (m=7). The packed field is transferred to the first 7 bytes of the output record (t=1).

Field 3 is to be unpacked; operand entry is r,(U,n,m),t. It is seven bytes long during input (n=7) and must have at least (2x7)-1=13 bytes in the output record (m=13). It is moved from bytes 74-80 (r=74) to bytes 8-20 (t=8). (The formulas for minimum field size for packing and unpacking are presented in the section Field-Select Statement).

In this problem, the input records and the output records are of the same size; therefore, the output description operand of the utility-modifier statement, B=(a,c), is identical to the input description operand, A=(n,m).

4. Transfer a file from disk to tape without change. Records are unblocked, 270 bytes each. Rewind and unload the tape.

The only utility control statement needed is

```
//bEND
```

Default specifications for all operands satisfy the job requirements.

The Tape-to-Disk utility program transfers a data file from one or more reels of tape to one or more disk packs mounted on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5).

All input and output records must be fixed-length, blocked or unblocked. Input blocks may (in this program only) contain a variable number of fixed-length records. The following job-control and utility control statements are used by the Tape-to-Disk utility program.

Job-Control Statements

Statement	Use	Operand Entries for Tape-to-Disk Utility
// JOB	Required	Program name = TAPDSK
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Alternate input device = SYS000
// ASSGN	Optional	Alternate output device = SYS002-SYS004
// ASSGN	Optional	Logging device = SYS- (See LOG Statement)
// FILES	Optional	
// UPSI	Optional	
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	If label processing on input tape Input file name = UIN
// TPLAB	Required	If Label processing or input take
// VOL	Required	Output file name = UOUT
// DLAB	Required	One set for output file.
// XTENT	Required	More than one XTENT statement may be used in the set.

// LOG
or
// NOLOG
	Optional	

// EXEC	Required	

*Unless information is still in main storage.

For a detailed description of the job-control statements refer to Appendix A.

Note: When the Tape-to-Disk utility program is used as inquiry program, the tape files are treated as unlabeled files and tape labels are not processed. If you have issued TPLAB statement, it is ignored.

Utility Control Statements

Two utility control statements may be used with the Tape-to-Disk program:

The utility-modifier statement, which outlines the method of transferring a set of data records (a file) from tape to disk.

The field-select statement, which describes the transfer of individual fields from the input record to the output record.

A third utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bEND	Required

TAPE-TO-DISK UTILITY-MODIFIER STATEMENT

The utility-modifier statement controls the file transfer by indicating the type of transfer, describing the input and output files, and specifying certain input and output device actions. Its general format is

Name	Operation	Operand
//b	UTDb	Tt,FF,A=(input),B=(output),Ix,Ox

Possible operand entries are shown in Figure 10.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

TD Program initials for the Tape-to-Disk program. They may be omitted, since the JOB statement actually specifies a tape-to-disk program (Program name = TAPDSK). If used at all, the initials must be TD; otherwise a warning message will be printed.

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (TD) are used, or in column 6 if TD is omitted. These four operands must

Operand	Possible Forms	Operand Specification	Explanation	Default Specification	

Statement ID
//bUxxb
	//bUTDb
	
	//bUb
		TD	Tape-to-Disk utility identifier
			General utility identifier
				//bUb	

Type of Function Tt	TC TF TR TRF	C F R RF	Copy Field-Select Reblock Reblock and Field-Select	TC	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,m)	 n m	Fixed-length records Input record length, Input block length	 A=(270,270)	*
Output Description B=(output)	B=(a,c)	 a c	Fixed-length records Output record length, Output block length	 B=(270,270)	*
Rewind Input Tape Ix	IN IR IU IM	N R U M	Do not rewind Rewind Rewind and unload Multiple-reel input (All reels are rewound and unloaded)	IU	
Disk Check Ox	OY ON	Y N	Write-disk check Do not write-disk check	OY	

*Default permitted only if entire card is omitted.

```
//bUTDbTt,FF,A=(input),B=(output),Ix,Ox
```

Figure 10. Tape-to-Disk Program Utility-Modifier Statement

be listed in the above order, and all must appear unless the card is omitted entirely. The additional operands (Ix and Ox) may appear in any order after the four required ones.

Tt - Type of Function operand

This operand specifies the transfer to be performed, when "t" is replaced by one or two initials. Files can be transferred in four ways:

- TC -- Copy. No change is made in the records or the file itself. This option is used to produce an identical file.
- TR -- Reblock. The number of records per block in the output (disk) file is different from that of the input (tape) file. The record format remains unchanged.
- TF -- Field-Select. The record format is changed. Data fields within each record are rearranged, omitted, or converted to zoned or packed decimal. The output record length may differ from the input record length; however, the number of records per block remains the same as in the input file.
- TRF -- Reblock and Field-Select. This is a combination of the two preceding options. It is used when both the record format and the number of records per block are to be changed.

FF - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e. all records contain the same number of bytes. The characters FF must be entered.

A=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read. The format of this operand is

A=(n,m), where

n is the number of bytes in each record, and

m is the number of bytes in each block.

If the input (tape) records are blocked and the number of records are block varies, n must designate the maximum block size. The values n and m must be separated by a comma and enclosed in parentheses.

B=(output) - Output Record Length and Block Length Operand

This operand describes the records and blocks of the output file. Its format is

B=(a,c), where

a is the number of bytes in each record, and
c is the number of bytes in each block.

The value c equals the number of bytes per record multiplied by the number of records per block. The values a and c must be separated by a comma and enclosed in parentheses.

Ix - Rewind Tape

Ix is an optional operand which determines whether the input tape is to be rewound (IR), rewound and unloaded (IU), or not rewound (IN) after use. IU is the default value. If multiple-reel input is specified (IM), an alternate input unit may be assigned and all reels will be rewound and unloaded after use. Input tapes are not rewound before use.

Ox - Write-Disk Check

Ox is an optional operand that may occur after the four required operands. The write-disk check option is used to read and check data immediately after it has been written on disk. If the data cannot be written properly, a message is printed and the program halts. It is recommended that the write-disk check option should be used.

If one or both of the optional operands (Ix, Ox) or the utility-modifier statement itself are omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s).

You can omit the entire Tape-to-Disk utility-modifier statement (and run your program with the END statement only), if all your requirements match the default specifications as shown in Figure 10:

- Copy only, no field-selection or reblocking.
- Fixed-length records.
- Input record and block lengths are both 270 bytes.
- Output record and block lengths are both 270 bytes.
- Rewind and unload input tape.
- Write-disk check is to be made.

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Tape-to-Disk program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

The FS statement is only required if TF or TRF is specified in the Tt operand of the Tape-to-Disk utility-modifier statement.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

$$\text{If n is odd, } m = \frac{n+1}{2}$$

$$\text{If n is even, } m = \frac{n+2}{2}$$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

Sample Problems

1. Copy a file from tape to disk. Both the input and the output media contain 90-byte records, three to a block. Do not rewind the input tape. Do not make a write-disk check.

```
//bUTDbTC,FF,A=(90,270),B=(90,270),IN,ON
//bEND
```

The control statement identification for Tape-to-Disk is //bUTDb, the TD entry being optional. The job is to copy (TC) fixed-length records (FF). The block length is calculated by multiplying the number of bytes per record by the number of records per block (90x3=270). No field-select statement is needed since the job is to copy records.

2. Transfer a file from several reels of tape to a disk pack. The input records are eighty bytes long, with four records in a block. On disk the records will be stored sixteen to a block. Make a write-disk check.

```
//bUTDbTR,FF,A=(80,320),B=(80,1280),
IM,OY
//bEND
```

or

```
//bUbTR,FF,A=(80,320),B=(80,1280),IM
//bEND
```

Both pairs of statements are acceptable because certain default specifications are appropriate for this job. An alternate input device may be assigned. IM indicates that more than one reel of tape is used as input. The tapes will be rewound and unloaded after use. Since the block size of the output file is different from that of the input file, the reblock option (TR) must be used. The records themselves are copied unchanged; therefore no field-select statement is needed.

3. Transfer a file from tape to disk, changing the format of the records. Rewind the tape. Do not make a write-disk check. The input records are eighty bytes long with three records to a block. The output record and block sizes will be the same as the input sizes. Both input and output records contain three fields. Further job specifications are listed in Figure 11. Figure 9 shows what is changed during transfer.

```
//bUTDbTF,FF,A=(80,240),B=(80,240),IR,ON
//bFSb1,60,21/61,(P,13,7),1/74,(U,7,13),8
//bEND
```

This field-select statement is discussed in Sample Problem 3 in the section Disk-to-Tape Utility Program.

4. Transfer a file from tape to disk without change. Records are unblocked, 270 bytes each. Make a write-disk check and rewind and unload the tape.

The only utility control statement needed is

```
//bEND
```

Default specifications for all operands satisfy the job requirements.

5. Transfer a file from tape to disk. The 90-byte input records are blocked, with the block size ranging from three to six records per block. The output records on disk are to be unblocked, 90 bytes long. Rewind and unload the input tape. Make a write-disk check.

```
//bUTDbTR,FF,A=(90,540),B=(90,90)
//END
```

Since the input blocks contain a variable number of records, the maximum block size, i.e., 6x90=540 must be specified. The Ix and Ox operands assume their default specifications.

Input Record			Output Record		
	Location (bytes)	Format	Location (bytes)	Format	Conversion
Field 1	1-60	packed decimal	21-80	packed decimal	none
Field 2	61-73	zoned decimal	1-7	packed decimal	to be packed
Field 3	74-80	packed decimal	8-20	zoned decimal	to be unpacked

Figure 11. Field Modification for Tape-to-Disk Transfer Problem

Disk-to-Card Utility Program (DSKCAR)

The Disk-to-Card utility program transfers a data file from one or more disk packs mounted on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5), to punched cards. The output file is always punched in the IBM standard card code equivalent to extended binary coded decimal (EBCDIC). The output (punched card) records must be unblocked and must not exceed 80 bytes each to fit on a single card. Input (disk) records may be blocked.

A method is provided to exit to a problem program for Sterling-currency conversion routines. (See Appendix D).

You can transfer the contents of an entire disk pack to punched cards by using the Disk-to-Card utility program with a special UPSI statement. This feature is further explained in Appendix I.

The following job-control and utility control statements are used by the Disk-to-Card utility program.

Job-Control Statements

Statement	Use	Operand Entries for Disk-to-Card
// JOB	Required	Program name = DSKCAR
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Alternate input devices = SYS002-SYS004
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// UPSI	Optional	
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	Input file name = UIN
// DLAB	Required	One set for input file.
// XTENT	Required	More than one XTENT statement may be used in the set.
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

*Unless information is still in main storage.

A detailed description of the job-control statements is given in Appendix A.

Utility Control Statements

Two utility control statements may be used with the Disk-to-Card program:

The utility-modifier statement, which outlines the method of transferring a set of data records (a file) from disk to card.

The field-select statement, which describes the transfer of individual fields from the input record to the output record.

A third utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bEND	Required

DISK-TO-CARD UTILITY-MODIFIER STATEMENT

The utility-modifier statement controls the file transfer from disk to cards, describes the input and output files, and specifies certain input and output device actions. Its general format is

Name	Operation	Operand
//b	UDCb	Tt,FF,A=(input),B=(output),Q=(x,y)

Possible operand entries are shown in Figure 12.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

DC Program initials for the Disk-to-Card program. They may be omitted since the JOB statement actually specifies a disk-to-card program (Program name = DSKCAR). If used at all the initials must be DC; otherwise a warning message will be printed.

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (DC) are used, or in column 6 if DC is omitted. These four operands must be listed in the above order, and all must appear unless the card is omitted entirely. The additional operand Q=(x,y) may appear after the four required ones.

Tt - Type-of-Function Operand

This operand specifies the transfer to be performed, when "t" is replaced by one or two initials. Files can be transferred in four ways:

- TC -- Copy. No change is made in the records or the file itself. Since this option is used to produce an identical file, the input file must have the same format as the output file, i.e., the disk records must already be unblocked and not exceed 80 bytes each.
- TR -- Reblock. This option is used if the input records are blocked, and the record length is up to 80 bytes. During transfer the file is being unblocked. The record format remains unchanged.
- TF -- Field-Select. The record format is changed. Data fields within each record are rearranged, omitted, or converted to zoned or packed decimal. This option is used if input records are unblocked and more than 80 bytes long.
- TRF -- Reblock and Field-Select. This is a combination of the two preceding options. It is used when the record format is to be changed and the input records are to be blocked.

FF - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e. all records contain the same number of bytes. The characters FF must be entered.

A=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read. The format of this operand is

A=(n,m), where
n is the number of bytes in each record, and
m is the number of bytes in each block. Since the number of records per block is normally constant m can be worked out by multiplying the number of bytes per record by the number of records per block. The values n and m must be separated by a comma and enclosed in parentheses.

B=(output) - Output Record Length and Block Length Operand

This operand describes the output file. Since the output (punched card) records must be unblocked, the block length is equal to the record length; thus B=(a,c) becomes B=(a,a), where a is the number of bytes in each punched card.

Q=(x,y) - Sequence Numbering

The optional operand Q=(x,y) in the utility-modifier statement may be used to punch sequence numbers into the output cards. The numbers begin with the value 1 (with leading zeros) and increase by one with each successive card. The value x designates the first (high-order) column of the sequence-number field in the output cards. The value y is the length of the sequence-number field in the output cards, which may be one to ten columns. If the field is not long enough to number all the cards, the numbers begin again at zero (not 1) and a warning message is printed.

The sequence-number field is the last field moved to the output area and, therefore, replaces any other data that may have been previously placed in that area. Note that the output record length refers to the data transferred from the input medium, and need not allow for the sequence-number field if this is to be punched to the right of the data record. A warning message is printed if the sequence-number field overlays a data output field.

If this operand is omitted, sequence numbers will not be punched.

If the optional operand Q=(x,y) or the utility-modifier statement itself is omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s).

You can omit the entire Disk-to-Card utility-modifier statement (and run your program with the END statement only), if all your requirements match the default specifications as shown in Figure 12:

- Reblocking.
- Fixed-length records.
- Input record length is 80 bytes, input block length is 240 bytes.
- Output record and block lengths are both 80 bytes.
- No sequence numbering.

Operand	Possible Forms	Operand Specifications	Explanation	Default Specifications	

Statement ID
//bUxxb
	//bUDCb
	
	//bUb
		DC	Disk to Card utility identifier
			General utility identifier
				
				//bUb	

Type of Function Tt	TC TF TR TRF	C F R RF	Copy Field-Select Reblock Reblock and Field-Select	TR	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,m)	 n m	Fixed-length records Input record length, Input block length	 A=(80,240)	*
Output Description B=(output)	B=(a,a)	 a a	Fixed-length records Output record length, Output block length	 B=(80,80)	*
Sequence Numbering Q=(x,y)	Q=(x,y)	x y	First (high-order) card column for the sequence-number field Length (number of columns) of the field	Sequence Numbering Omitted	

*Default permitted only if entire statement is omitted.

```
//bUDCbTt,FF,A=(input),B=(output),Q=(x,y)
```

Figure 12. Disk-to-Card Program Utility-Modifier Statement

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Disk-to-Card program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

The FS statement is only required if TF or TRF is specified in the Tt operand of the Disk-to-Card utility-modifier statement.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

If n is odd, $m = \frac{n+1}{2}$

If n is even, $m = \frac{n+2}{2}$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

Sample Problems

1. Punch into cards the data from a disk file consisting of unblocked records, 80 bytes per record. Do not punch sequence numbers.

```
//bUDCbTC,FF,A=(80,80),B=(80,80)
//bEND
```

The job is to copy records without change (TC). Input and output record and block length is eighty bytes. No field-select statement is needed.

2. Punch cards with data from a disk file containing eighty-byte records, six records to a block. Punch sequence numbers into columns 71-80.

```
//bUDCbTR,FF,A=(80,480),B=(80,80),
  Q=(71,10)
//bEND
```

Since the output records must be unblocked, the reblock (TR) option applies. The input block size is 80x6=480 bytes. The sequence-number field starts in column 71 on the output card (x=71); it occupies ten columns (y=10).

Since the sequence numbers are transferred after the data has been moved, any information in bytes 71-80 of the input will be overlayed.

3. Punch cards with data from a disk file containing records 200 bytes long, 13 records to a block.

Figure 13 shows how the transfer is made.

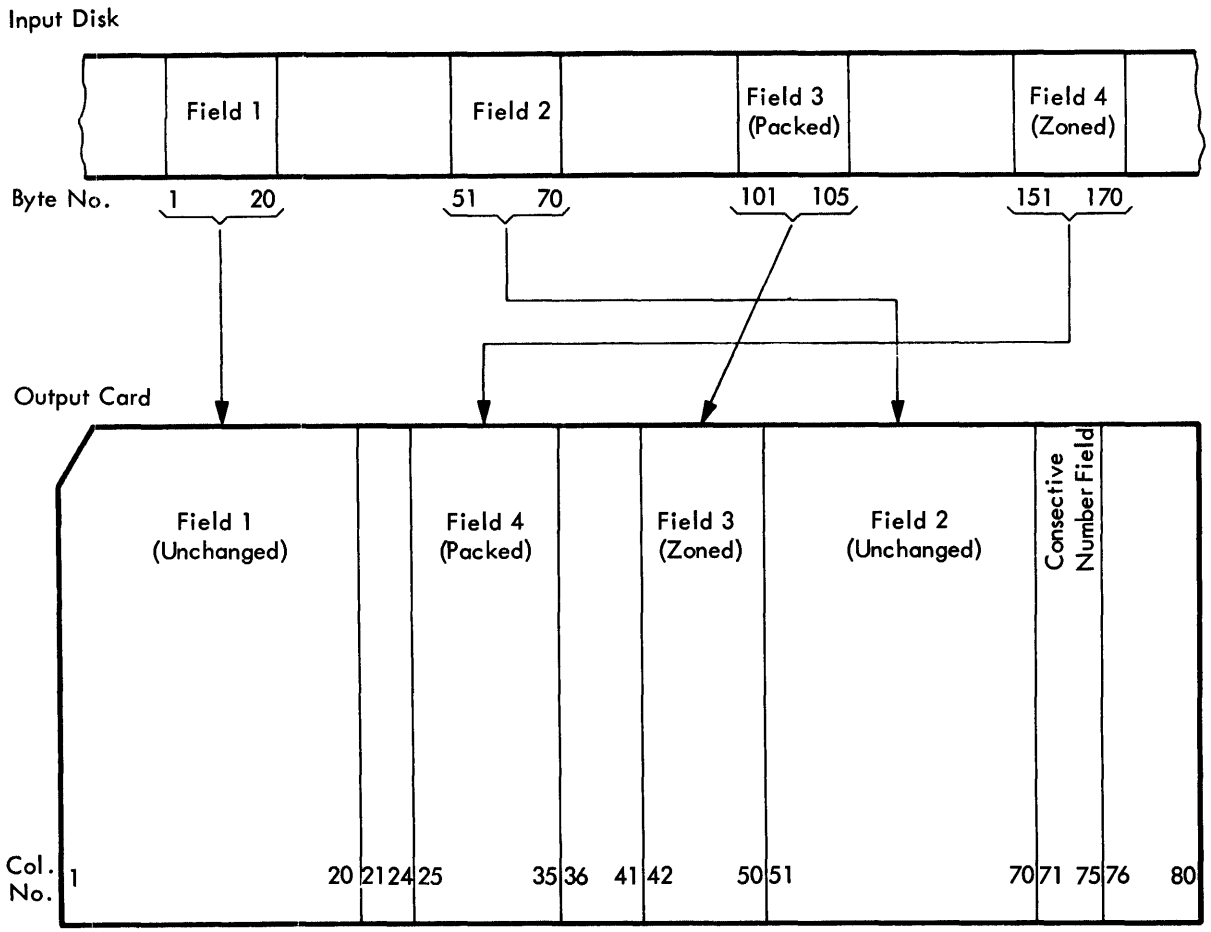
The utility control statements required are

```
//bUDCbTRF,FF,A=(200,2600),B=(70,70),
  Q=(71,5)
//bFSb1,20,1/51,20,51/101,
  (U,5,9),42
//bFSb151,(P,20,11),25
//bEND
```

The record format is changed and the records have to be unblocked (TRF). The input block length is 200x13=2600 bytes.

If a serial card punch is used, throughput speed may be increased by punching less than 80 columns - provided the surplus blank columns are on the right-hand side of the card. Thus, in this example, the output record length has been specified as 70 to save the time of spacing over the last five columns.

The sequence-number field starts in column 71 (x=71) and occupies five digits (y=5).



● Figure 13. Field Modification for Disk-to-Card Problem No. 3

The field-select statement controls the change in record format. Field 1 is moved without change in format; starting position in the input record r=1, field length s=20, starting position in the output record t=1.

Operand entries are separated from each other by a slash (/). Field 2 is likewise moved without change: from r=51 to t=51; length s=20.

Field 3 is to be unpacked; its general operand entry is r,(U,n,m),t. Field 3 starts at byte 101 of the input record (r=101) with a length of five bytes (n=5). It must be assigned at least (5x2)-1=9 bytes in the output record (m=9). The unpacked field is transferred to bytes 42 - 50 of the output record (t=42).

Field 4 is to be converted to packed decimal format: r,(P,n,m),t. It is moved from r=151 on disk to columns 25 - 35 on the card (t=25). It is 20 bytes long on disk (n=20) and must be assigned (20+2)/2=11 bytes (columns) on the card (m=11). (The formulas for computing the length of packed or unpacked fields are given in the section General Description of Field-Select Statement). Fields not listed in the field-select statement are blank in the output cards: columns 21-24, 36-41, and 76-80.

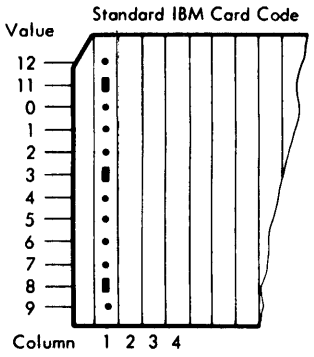
One FS statement would have been enough for all field-select specifications. The second FS statement (no continuation statement is allowed!) is used for illustrative purposes only. Note that there is no slash after the last operand in each of the FS statements.

4. Punch cards with all the data from a disk file composed of 80-byte records, three to a block. Do not punch sequence numbers.
   //bEND

Default specifications describe the problem. The utility-modifier statement may be omitted; and there is no field-selection that would require the FS statement.

The Card-to-Disk utility program transfers the contents of a card file to one or more disk packs mounted on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5). The input (card) file must be unblocked, with each record contained in a single card (i.e., ≤ 80 columns). Output (disk) records may be blocked.

The input card file may be read in the IBM standard card code equivalent to extended binary coded decimal (EBCDIC) or it may be read in as column binary. In the latter format, the punches in each card column are transferred to two successive bytes of main storage, with the two leftmost bits in each byte set to zero - thus, the "image" of punches in a card column is stored in bit equivalents. For example, the standard card code equivalent to EBCDIC for the character $ would be 11-3-8: holes are punched in rows 11, 3, and 8 of a column (see Figure 14).



- Figure 14. Punched Card Code of the $-Sign

If the character $ is read in as column binary, the punches in the upper 6-bit set of the column (B-A-8-4-2-1) are transferred to one byte of main storage with the two leftmost bits set to zero: 0001 0001.

The punches in the lower 6-bit set of the column (B-A-8-4-2-1) are transferred to the adjacent byte in main storage with the two leftmost bits again set to zero: 0000 0010 (See Figure 15).

The maximum input record size is therefore 80 bytes for EBCDIC and 160 for column binary. With column-binary input, the input record size must be specified as twice the number of columns to be read, because two input-area bytes must be available for each column. The output record must also be specified to contain twice as many bytes as the number of input columns to be transferred. (See the section Column Binary Feature in IBM System/360 Model 20, Functional Characteristics, Form A26-5847.)

A method is provided to exit to a user's program for Sterling-currency conversion routines. (See Appendix D.)

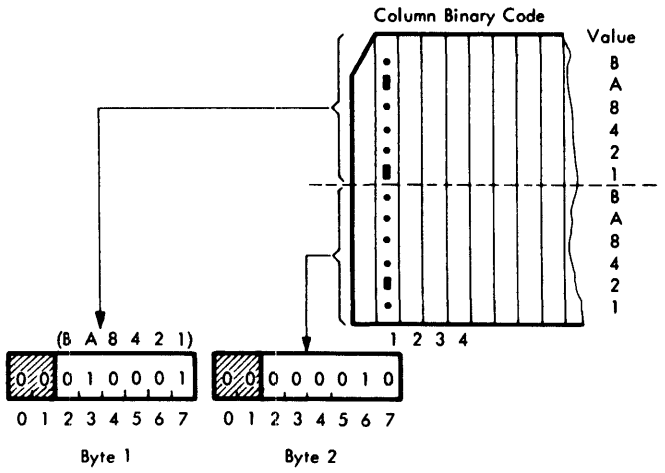
The following job-control and utility control statements are used by the Card-to-Disk utility program.

Job-Control Statements

Statement	Use	Operand Entries for Card-to-Disk
// JOB	Required	Program name = CARDSK
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Alternate output devices = SYS002-SYS004
// ASSGN	Optional	Logging device = SYSLOG (See LOG Statement)
// UPSI	Optional	
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	Output file name = UOUT.
// DLAB	Required	One set for output file.
// XTENT	Required	More than one XTENT statement can be used in the set.
// LOG or // NOLOG	Optional	
// EXEC	Required	

*Unless information is still in main storage.

For a detailed description of the job-control statements refer to Appendix A.



- Figure 15. Column Binary Reading of the $-Sign

Utility Control Statements

Two utility control statements may be used with the Card-to-Disk program:

The utility-modifier statement, which outlines the method of transferring a set of data records (a file) from cards to disk.

The field-select statement, which describes the transfer of individual fields from the input record to the output record.

A third utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bEND	Required

CARD-TO-DISK UTILITY-MODIFIER STATEMENT

The utility-modifier statement controls the file transfer from cards to disk, describes the input and output files, and specifies certain input and output device actions. Its general format is

Name	Operation	Operand
//b	UCDb	Tt,FF,A=(input),B=(output),Ix,Ox,Q=(x,y)

Possible operand entries are shown in Figure 16.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

CD Program initials for the Card-to-Disk program. They may be omitted since the JOB statement actually specifies a card-to-disk program (Program name = CARDSK). If used at all, the initials must be CD; otherwise a warning message will be printed.

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (CD) are used, or in column 6 if CD is omitted. These four operands must be listed in the above order, and all must appear unless the card is omitted entirely.

Ix, Ox, and Q=(x,y) are optional operands and may be omitted from the control statement. These operands may appear in any order after the four required operands.

Tt - Type-of-Function Operand

This operand specifies the transfer to be performed, when "t" is replaced by one or two initials. Files can be transferred in four ways:

- TC -- Copy. No change is made in the records or the file itself. This option is used to produce an identical file (both input and output records are unblocked).

- TR -- Reblock. The unblocked input card file is transferred with more than one record per block to the output disk. This may speed subsequent reading of the disk data and conserve space on disk.

- TF -- Field-Select. The record format is changed. Data fields within each record are rearranged, omitted, or converted to zoned or packed decimal. The output record length may differ from the input record length; however, the number of records per block remains the same as in the input file.

- TRF -- Reblock and Field-Select. This is a combination of the two preceding options. It is used when the record format is to be changed and the output records are to be blocked.

FF - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e. all records contain the same number of bytes. The characters FF must be entered.

Operand	Possible Forms	Operand Specifications	Explanation	Default Specification	

Statement ID
//bUxxb
	//bUCDb
	
	//bUb
		CD	Card-to-Disk utility identifier
			General utility identifier
				//bUb	

Type of Function Tt	TC TF TR TRF	C F R RF	Copy Field-Select Reblock Reblock and Field-Select	TR	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,n)	 n n	Fixed-length records Input record length, Input block length	 A=(80,80)	*
Output Description B=(output)	B=(a,b)	 a c	Fixed-length records Output record length, Output block length	 B=(80,240)	*
Card-Data Code Structure Ix	I1 I2	1 2	Standard punch card code (EBCDIC equivalent) Column binary	I1	
Disk Check Ox	OY ON	Y N	Write-disk check Do not write-disk check	OY	
Sequence Checking Q=(x,y)	Q=(x,y)	x y	First (high-order) card column of the sequence-number field Length (number of columns) of the field	Sequence Checking omitted	

*Default permitted only if entire statement is omitted.

```
//bUCDbTt,FF,A=(input),B=(output),Ix,Ox,Q=(x,y)
```

Figure 16. Card-to-Disk Program Utility-Modifier Statement

A=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read.

Because the input (card) records are unblocked, the block length is identical with the record length; thus A=(n,m) becomes A=(n,n), where n represents the number of bytes in each record/block.

B=(output) - Output Record Length and Block Length Operand

This operand describes the records and blocks of the output file. Its format is

B=(a,c) where

a is the number of bytes in each record, and
c is the number of bytes in each block.

The value c equals the number of bytes per record multiplied by the number of records per block. The values a and c must be separated by a comma and enclosed in parentheses.

Ix - Input Format

Ix is an optional operand that specifies standard card code (I1) or column binary input (I2).

Ox - Write-Disk Check

The optional write-disk check operand is used to read back and verify data immediately after it has been written on disk. If the data cannot be written properly, a message is printed, and the program halts. It is recommended that the write-disk check option should be used.

Q=(x,y) - Sequence Checking

If a sequence-number field is specified in the utility-modifier statement by Q=(x,y), the program will check the numbers in that field for ascending numeric sequence. (The numbers need not be consecutive). The value x designates the first (high order) column of the sequence-number field in the input cards. The value y is the length of the sequence-number field in the input cards, which may be one to ten columns.

The sequence begins with the number in the first data card. If a card is out of sequence, an error message is printed and processing continues. An error is defined as either a step-down in sequence, or repetition of the preceding number. The error message shows of the contents of the sequence-number field of the first card that failed to contain an ascending-sequence number. The error message also shows the sequence-number field of the preceding (last correctly sequenced) card. The number that caused the error message then starts a new series.

Say, the following numbers occur in the sequence-number field of a series of cards:

1,3,4,5,4,5,6,6,7

The first error message would print out numbers 5 and 4. 4 then starts a new series, i.e., the following number must be higher than 4. Since number 6 occurs twice, it is printed out in the second error message.

If the operand is omitted, no sequence checking is performed. Neither is it performed if column-binary input (I2) has been specified.

If one or more of the optional operands Ix, Ox, Q=(x,y) or the utility-modifier statement itself are omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s).

You can omit the entire Card-to-Disk utility-modifier statement (and run your program with the END statement only), if all your requirements match the default specifications as shown in Figure 16:

- Reblocking.
- Fixed-length records.
- Input record and block lengths are both 80 bytes.
- Output record length is 80 bytes, output block length is 240 bytes.
- Standard card code.
- Write-disk check is to be made.
- No sequence checking.

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Card-to-Disk program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

The FS statement is only required if TF or TRF is specified in the Tt operand of the Card-to-Disk utility-modifier statement.

The general format of the field-select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

$$\text{If n is odd, } m = \frac{n+1}{2}$$

$$\text{If n is even, } m = \frac{n+2}{2}$$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

END-OF-FILE STATEMENT

The end of the data deck is defined by a card containing /* in columns 1-2. Unless columns 3-80 are blank, the card is treated as a data card.

Sample Problems

1. Create a disk file from a deck of cards punched in column binary, 80 columns. Do not block. Perform the write-disk check.

```
//bUCDbTC,FF,A=(160,160),B=(160,160),
  I2,OY
//bEND
```

The control statement identification for a Card-to-Disk program is //bUCDb, the CD being optional. The job is to copy (TC).

No field-select statement is used. Column-binary data is counted as two bytes per column to allow as many as 160 bytes per card (160,160). The data from each column is transferred into two bytes on disk. Sequence checking is not possible with column binary (I2) input.

2. Write a file on disk, placing the data from seven 80-column cards in each block. The cards are punched in IBM standard card code. Do not make write-disk check. Check the last seven card columns for sequence.

```
//bUCDbTR,FF,A=(80,80),B=(80,560),
  I1,ON,Q=(74,7)
//bEND
```

or

```
//bUbTR,FF,A=(73,73),B=(73,511),ON,
  Q=(74,7)
//bEND
```

The alternate (second) utility-modifier statement is to be preferred. Besides omitting the optional program initials CD (since the JOB control statement actually specifies the Card-to-Disk utility program) and the card-data code I1 (the default specification), an important point is illustrated: The input and output record lengths refer to the data to be transferred and need not allow for the sequence-number field if this is to the right of the last

data field. This can be significant. In this example, combining seven 80-column input card records into one disk block requires 560 bytes per block. Since each block on disk must begin at a multiple of 270 (see Appendix E), each block would occupy three disk sectors. However, if the record length is specified as 73 -- since the last seven columns of each card contain only the sequence-number field -- a block of seven records requires only 511 bytes on disk, thus occupying only two sectors.

3. Write the data from card columns 32-69 onto a disk file, seven records per block. The cards are punched in standard IBM card code. Check the first five columns for sequence numbers. Do not write-disk check.

```
//bUCDbTRF,FF,A=(69,69),B=(38,266),
  I1,ON,Q=(1,5)
//bFSb32,38,1
//bEND
```

Columns 1 to 69 will be read. However, only columns 32 to 69 (38 columns) will be transferred to disk, therefore 38 bytes are needed in the output record, seven records per block. The field-select statement causes the data from card columns 32-69 (r=32) to be transferred to bytes 1-38 (s=38, t=1) of the output record. The sequence-number field in the input cards starts in column 1 and occupies 5 columns.

4. Write a disk file from cards, three records to a block, transferring each record completely. The card data occupies all 80 columns and is punched in IBM standard card code. Make a write-disk check. Do not check sequence. The only utility control statement required is

```
//bEND
```

Neither the utility-modifier nor the field-select statement is needed; default specifications will be used.

The Disk-to-Printer utility program prints data from a disk file contained in one or more disk packs mounted on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5).

The program uses the standard printer carriage-control tape. A punch in channel 1 indicates the first print line of a page and a punch in channel 12 indicates the last print line of a page.

An optional exit to a problem program for Sterling-currency conversion is provided. (See Appendix D).

It is possible to list or display the contents of an entire disk pack by using the Disk-to-Printer program with a special UPSI statement. This feature is further explained in Appendix I.

The following job-control and utility control statements are used by the Disk-to-Printer utility program:

Job Control Statements

Statement	Use	Operand Entries for Disk-to-Printer
// JOB	Required	Program name = DSKPRT
// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Required	Primary output device=SYSLST
// ASSGN	Optional	Alternate input devices = SYS002 SYS004
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// UPSI	Optional	See Appendix I
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	Input File Name - UIN. still in main storage.
// DLAB	Required	One set for input file.
// XTENT	Required	More than one XTENT statement may be used in the set.
// LOG or // NOLOG	Optional	
// EXEC	Required	

*Unless information is still in main storage.

For details on the job-control statements refer to Appendix A.

Utility Control Statements

Three utility control statements may be used with the Disk-to-Printer program:

The utility-modifier statement, which outlines the method of transferring a file from disk to the printer.

The field-select statement, which describes the transfer of individual fields from the input record to the output record.

The print-header statement, which allows a heading to be printed at the beginning of a page.

A fourth utility control statement, the END statement, must be used with every utility program.

The sequence of statements is as follows:

//bU	Optional
//bFS	Optional (More than one may be used)
//bHn	Optional (One or two statements)
//bEND	Required

DISK-TO-PRINTER UTILITY-MODIFIER STATEMENT

The utility-modifier statement contains the information required to transfer files from disk to the printer. The general format of the statement is

Name	Operation	Operand
//b	UDPb	Tt,FF,A=(input),B=(output),Ox,Sx,Px,Rx

Possible operand entries are shown in Figure 17.

//bU Identifies the statement as a utility-modifier statement. It must be punched starting in column 1.

DP Program initials for the Disk-to-Printer program. They may be omitted since the JOB statement actually specifies a disk-to-printer program (Program name = DSKPRT). If used at all, the initials must be DP; otherwise a warning message will be printed.

Operand	Possible Forms	Operand Specifications	Explanation	Default Specification	

Statement ID
//bUxxb
	//bUDPb
	
	//bUb
		DP	Disk-to-Printer utility identifier
			General utility identifier
				//bUb	

Type of Function Tt	TD TL TLF	D L LF	Display List List and Field Select	TD	*
Format FF	FF	F	Fixed-length records	FF	*
Input Description A=(input)	A=(n,m)	n m	Fixed-length records Input record length, Input block length	A=(270,270)	*
Output Description B=(output)	B=(p)		Printer output Size of the print line (Maximum: 120,132, or 144)	B=(120)	*
Print Output Ox	OX OC	X C	Hexadecimal printout Character (alphameric) printout	OX (if TD) OC (if TL or TLF)	
Spacing Sx (only if TL or TLF)	S1 S2 S3	1 2 3	Single spacing Double spacing Triple spacing	S1	
Page Numbering Px	PY PN	Y N	Number pages Do not number pages	PY	
First Record Printed Rx	Rx	x	Sequential position in input file of first record to be printed; x-1 records bypassed	R1	

*Default permitted only if entire statement is omitted.

```
//bUDPbTt,FF,A=(input),B=(output),Ox,Sx,Px,Rx
```

Figure 17. Disk-to-Printer Program Utility-Modifier Statement

The required operands are:

Tt,FF,A=(input),B=(output)

They start in column 8 if the program code initials (DP) are used, or in column 6 if DP is omitted. These four operands must be listed in the above order, and all must appear unless the card is omitted entirely.

Tt - Type-of-Function Operand

This operand specifies the transfer to be performed, when "t" is replaced by "D" (Display), or "L" (List), or "LF" (List and Field-Select).

1. TD -- Display. A complete, byte-for-byte representation of the file is printed, starting with the first byte of the logical record. Blocked or unblocked fixed-length records can be displayed. No change is made in the format of the records or file.

   The first twenty print positions are reserved for file description: block number, block size and record number. (These are printed at the beginning of every record).

Output is normally (default specification of the Ox operand) hexadecimal, with two characters representing the left and right halves of a byte, although character-format output may be specified instead. Print lines are single-spaced with a blank line between data blocks.

A scale line is printed at the top and bottom of each page. This line identifies every tenth print position (i.e., 10, 20, 30, etc.).

Each record begins on a new line; a number designating the sequential position of the record within the block is printed preceding the data of each record. If the record requires more than one print line, the file description is not printed on the overflow lines. Overflow of a single record occurs when the specified print span limit is reached, regardless of the position within the record.

When a data transmission error occurs, the program attempts to print the record.

2. TL -- List. Blocked or unblocked fixed-length records can be listed. Each record starts on a new line and is restricted to a single line of print. The entire print line is available for data; no file description is printed. If a record exceeds the specified print span, the excess right-hand positions of the record are truncated. Output is normally (default specification of the Ox operand) in character format, although hexadecimal format may be specified instead. Hexadecimal output requires two print positions to display each byte; therefore, care must be taken to avoid exceeding the print span.

   Single-spacing between lines is normal (default specification of the Sx operand), but double-and triple-spacing are available.

3. TLF -- List and Field Select. This is a combination of the list (TL) option with field selection (TF) and requires the use of a field-select statement. It allows you to select input data fields and to arrange them in any print order on the line. (Editing, in the sense of zero suppression, punctuation, or sign symbols, is not available). Fields may also be unpacked before printout.

   All selected fields are printed in character (alphameric) format, unless individual fields are specified - with a field-select statement - to be printed in hexadecimal format. (The number of print positions is then twice the number of data bytes).

All three print options permit header line printing on each page.

Consecutive page numbers, starting with 1, are also normally (default specification of the Px operand) printed at the bottom of each page on a predetermined line controlled by carriage-tape. The maximum size of the page number is nine digits; heading zeros are dropped. Page numbering can be suppressed.

Ff - Record-Format Operand

The second operand indicates that the records to be transferred are always in fixed-length format, i.e. all records contain the same number of bytes. The characters FF must be entered.

=(input) - Input Record Length and Block Length Operand

It indicates the length of the record and the block to be read, where

n is the number of bytes in each record, and

m is the number of bytes in each block.

Since the number of records per block is constant, m can be worked out by multiplying the number of bytes per record by the number of records per block. The values n and m must be separated by a comma and enclosed in parentheses.

B=(output) - Output Description

This operand is used with records to be printed. Its format is B=(p)

The value p must be enclosed in parentheses; it represents the maximum number of characters to be printed on one line. It must not exceed the print span of the printer on the system (120 for the 1403 and 2203 Printer, 132 for the 1403, or 144 for the 2203 Printer). A warning message is printed if, as a result of unpack or hexadecimal operands in a field-select statement, the output record size exceeds the value p.

Ox, Sx, Px and Rx are optional operands. They may be omitted from the control statement. These operands may occur in any order after the four required operands.

Ox - Print Format

Ox indicates whether the printout is to be in hexadecimal (OX) or character (OC) format. The default specification is hexadecimal for data Display (TD) and character for data List (TL). If the TLF option is selected, the Ox operand is ignored and OC is assumed. Individual fields can still be printed in hexadecimal format by the r,(X,n),t operand in a field-select statement. (See the section Hexadecimal, under Field-Select Statement).

Packed numbers are efficiently represented in hexadecimal without unpacking.

Sx - Spacing Control

Sx controls line spacing for the data List (TL or TLF) option. S1, S2, and S3 cause single, double and triple spacing, respectively: S1 is the default specification. This operand is ignored if it is specified for a Display (TD) function.

Px - Page Numbering

Px is the operand that indicates whether pages are to be numbered. PY causes page numbers to be printed; it is the default specification. PN causes page numbering to be suppressed.

Rx - File Positioning

Rx controls the number of records at the beginning of the file to be bypassed before printing. The first x-1 records are bypassed. The default specification is 1; since 1-1=0, printing would begin with the first record of the file. The maximum value of this operand is R99999.

When one or more of the optional operands or the utility-modifier statement itself are omitted from the utility program, the program automatically assumes certain standard ("default") specifications for the missing operand(s). If all the default specifications correspond to the specifications required for a particular job, the program may be run with the END statement only. Figure 17 lists the default specifications that apply when the utility-modifier statement is omitted.

FIELD-SELECT STATEMENT

This section provides you with the basic information you need for writing the Field-Select statement for the Disk-to-Printer program. For a more detailed discussion see the section Field-Select Statement under General Description of Utility Control Statements.

The FS statement is only required if the TLF function is specified in the Tt operand of the Disk-to-Printer Utility-Modifier statement.

The general format of the Field-Select statement (when there is no data format conversion) is

Name	Operation	Operand
//b	FSb	r,s,t/.../r,s,t

where

//bFSb identifies the statement as the field-select control statement. It must be punched starting in column 1.

r,s,t is the operand for a particular field, where

r - is the starting position in the input record of the field to be transferred

s - is the number of bytes to be transferred

t - is the starting position in the output record of the receiving field

To both move and convert a data field, specify the operand for that field in one of the following forms, depending on whether the input field is to be packed or unpacked:

Pack -- the input field is to be packed for output:

r,(P,n,m),t

r starting position in the input record of the field to be transferred

P pack

n length (in bytes) of the unpacked input field

m length (in bytes) of the packed output field

t starting position in the output record of the receiving field

To determine the minimum number of bytes required in the output field the following formulas may be used:

$$\text{If n is odd, } m = \frac{n+1}{2}$$

$$\text{If n is even, } m = \frac{n+2}{2}$$

Unpack - the input field is to be converted from packed to zoned decimal for output:

r,(U,n,m),t

r starting position in the input record of the field to be transferred

U unpack

n length (in bytes) of the packed input field

m length (in bytes) of the unpacked output field

t starting position in the output record of the receiving field

The formula for determining m is: m=2n-1

Hexadecimal - the input field is to be printed in hexadecimal format:

In the List mode (TL or TLF) of the Disk-to-Printer program, the entire record is normally printed in character format. The utility-modifier statement for that program includes an optional operand (OX) to present the entire record in hexadecimal format.

Specifying hexadecimal for one or more operands in the field-select statement and listing with the TLF option will cause one or more particular fields to be printed in hexadecimal format and the remainder of the record in character format. The specification hexadecimal in the field-select statement supersedes -- for the specific field(s) only -- the stated data format or default specification in the utility-modifier statement for a Disk-to-Printer program used to list (TLF) data.

The conversion of a field to hexadecimal format is specified by

r,(X,n),t

r is the starting position in the input record of the field to be transferred

x causes conversion to hexadecimal

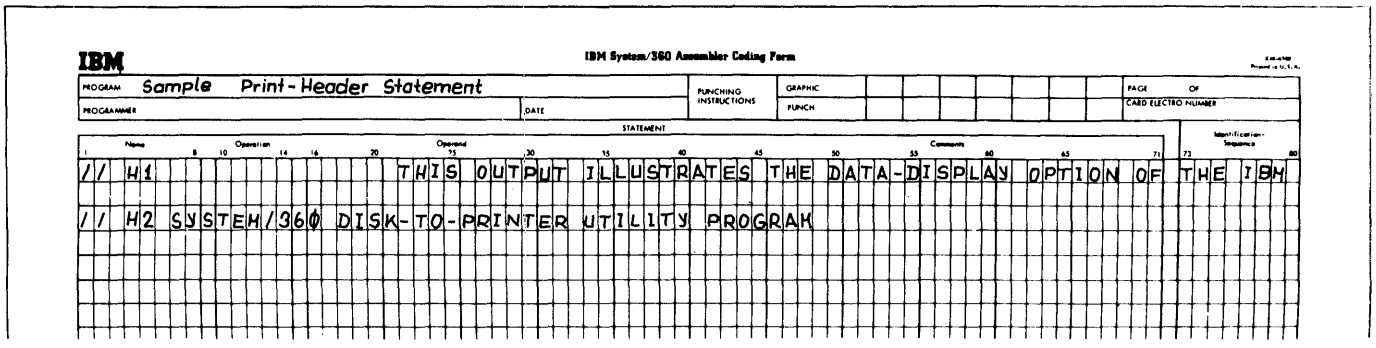
n is the number of bytes in the input field. The output field is assumed to require twice as many print positions as there are bytes in the input field. Care should be taken when filling out the utility-modifier statement that sufficient space is allotted for this expansion.

t is the starting position in the output record of the receiving field.

PRINT-HEADER STATEMENT

One line of heading information can be printed at the beginning of each page under both the Display and the List options (TD, TL, TLF). The heading is punched into one or two header cards. The format of the first is //bH1b in columns 1-6, followed in columns 7-80 by the heading to be printed in print positions 1-74. If a second card is needed, it contains //bH2b in columns 1-6, and is followed by the information to be printed on the rest of the print line.

For example, print the heading THIS OUTPUT ILLUSTRATES THE DATA-DISPLAY OPTION OF THE IBM SYSTEM/360 DISK-TO-PRINTER UTILITY PROGRAM. One hundred thirty-two print positions have been specified as the print span of the printer.



● Figure 18. Sample Print-Header Statement

The heading is 101 characters long. Centering it requires 15 blank spaces to the left and 16 to the right of the heading. The header cards are shown in Figure 18. Place them between the FS and the END statements.

END CONTROL STATEMENT

The END statement is the last of the utility control statements and must appear even if none of the others is needed. It is punched

Name	Operation	Operand
//b	END	

in columns 1-6.

Sample Problems

1. Print, in Display format, data from a disk file containing unblocked records, 200 bytes each. Print in character mode; number the pages. Print every record. The printer 1403 has 132 print positions.

```
//bUDPbTD,FF,A=(200,200),B=(132),OC,
  PY,R1
//bEND
```

   The field-select statement is not necessary, nor can it be used with the TD option.

2. Print the data from a disk file, in List format, double-spaced. The records are unblocked and 80 bytes long. The printer (2203) has 144 positions. The printout should be in hexadecimal format. Pages are not to be numbered. Bypass the first record of the file.

```
//bUDPbTL,FF,A=(80,80),B=(144),OX,
  PN,S2,R2
//bEND
```

   Hexadecimal printout requires two print positions per input byte: 80x2=160. Therefore, the rightmost eight bytes of input will be truncated, and not printed: (160-144):2=8. This could be avoided by using the TLF option, and selecting the essential fields, omitting at least eight bytes. Alternatively, one might specify the character mode with the TLF option and use the field-select statement to convert most fields to hexadecimal, leaving enough of them in character mode to stay within the print span.

   Since the printout is to start with the second record, Rx must be specified as R2; then 2-1 = 1 record will be bypassed.

3. Print data from a disk file in List format on a printer with 120 positions. Records are 100 bytes long, five to a block. Only five fields are to be printed.

Input Disk Bytes	Output Printer Positions	Special Conversions
1-20	1-20	
21-30	31-49	unpack
91-100	56-75	hexadecimal
56-75	76-95	
76-80	106-114	unpack

   Except for the third field, printing will be in character format, single-spaced. Pages are to be numbered. Do not print the first five records.

```
//bUbTLF,FF,A=(100,500),B=(114),R6
//bFSb1,20,1/21,(U,10,19),31/91,(X,10),56
//bFSb56,20,76/76,(U,5,9),106
//bEND
```

   Since the output record requires 114 print positions, B=(p) has been specified as 114 rather than the maximum number of positions for the printer (120). Using this technique saves processing time.

   The field-select statements control the change in record format. Field 1 is printed without change; its general operand entry is r,s,t. Starting position in the input record r=1, field length s=20, starting position in the printed output record t=1.

   Operand entries are separated from each other by a slash (/). Field 2 is to be unpacked; its general operand entry is r,(U,n,m),t. It starts at byte 21 of the input record (r=21) with a length of ten bytes (n=10). It must be assigned at least 2x10-1=19 bytes in the output record (m=19). The unpacked field is transferred to bytes 31-49 of the output record (t=31).

   Field 3 is to be printed in hexadecimal format; its general operand entry is r,(X,n),t. It is moved from input bytes 91-100 (r=91) to bytes 56-75 in the output record (t=56). Input field length n=10. The output field length is not explicitly stated; it is assumed to occupy twice as many print positions as there are bytes in the input field.

   Field 4 is moved unchanged from r=56 to t=76 with a length of s=20.

The last field is unpacked and transferred from r=76 to t=106; its input field length n=5. Therefore 2x5-1=9 bytes must be allotted for it in the output record (m=9).

The default specifications are the same as those requested for Ox, Sx, and Px. These operands may therefore be omitted. Example 2, above, explained the field expansion for hexadecimal printout.

4. Print data from a disk file in the Display mode. Input records are 270 bytes long. Records are unblocked. The print line is 120 characters long. Hexadecimal printout is requested. Pages are to be numbered, and printing is to begin with the first record.

```
//bEND
```

No utility-modifier statement is necessary since default specifications describe the problem, nor is a field-select statement used. Note that each record requires six print lines (540 print positions for the data in hexadecimal format, plus the first 20 positions on each line reserved for file description).

Initialize Disk Utility Program (INTDSK)

This program prepares one or more IBM 1316 Disk Packs for use on IBM 2311 Model 11 or 12 Disk Storage Drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5). Three separate options are available with the Initialize Disk Utility program.

1) Primary Initialization (TPI) is performed to initialize a disk pack that is to be used on either an IBM 2311 Model 12 Disk Drive or an IBM 2311 Model 11 Disk Drive. The following functions are performed:

   a) Checking the Volume Table of Contents, if present, for unexpired files (see Appendix B).

   b) Performing a surface analysis of the alternate track area (cylinders 1-3) to locate defective tracks that should not be assigned as alternate tracks.

   c) Performing an analysis of the disk surface to locate defective tracks and assign alternate tracks.0 In the course of this analysis, the data fields are set to binary zeros.

   d) Recording an IBM standard volume label on cylinder 0, track 1, sector 0.

   e) Assigning a cylinder for the Volume Table of Contents (VTOC) and recording the VTOC label (i.e., the Format-4 file label for the VTOC itself - see Appendix B).

   f) Assigning one or two tracks for the Label Information Area (LIA) used by the Job Control program.

2) Secondary Initialization (TSI) is performed to complete the initialization of a disk pack, previously initialized for an IBM 2311 Model 12 Disk Storage Drive, that is intended for use on Model 11. The preparation consists of two steps.

   a) Checking the VTOC label to verify that the second half of the disk pack (cylinders 103-202) contains no unexpired files.
   b) Performing a surface analysis of cylinders 103 through 202 to locate defective tracks and assign alternates. The data fields are set to binary zeros in cylinders 103 through 202.

3) List VTOC (TL) - This option of the program lists the Volume Table of Contents from one or more disk packs. When this option is selected, other activities cannot be performed simultaneously. The print format of the VTOC listing is presented in Figures 19 and 20.

VTOC - Label Checking

The Initialize-Disk program first checks for the presence of a volume label. If one is found, the program searches the file labels in the existing VTOC for unexpired files. If such files are found, a message is written, the program halts and you may either continue to process or, if you want to save the file, remove that disk pack before continuing to process.

Surface Analysis

Surface analysis involves testing a track for areas upon which data cannot be written owing to a disk surface defect.

It is performed first on the alternate track area (cylinders 1-3), which - unless defective - will be used as alternates for defective tracks. If a track in cylinders 1-3 is found defective, bit 6 of the flag byte of the count field (see Appendix E) in each sector of that track is changed to 1. The track then cannot be assigned as an alternate. Cylinders 1-3 are only available as alternate tracks.

After these cylinders have been analyzed, the remaining cylinders are tested. If a bad track is found, an alternate is set up and a message is written for your information.

When assigning an alternate track, the program changes the flag byte in each sector of both the defective and the alternate tracks. In each sector of the defective track, bit 6 of the flag byte is changed from 0 to 1, indicating a bad track; and the cylinder number and head number of the alternate track (cchh) are written in the track address portion of each count field. In the alternate track, the cylinder number and the track number of the bad track are

VOLUME TABLE OF CONTENTS PHI001

NEXT ALT TRK CCCHH	NUMBER OF ALT TRKS AVAILABLE	DEVICE SIZE	ERASE 0=NO 1=YES	VTOC EXTENT LOWER UPPER CCCHH CCCHH
00100	30	102	1	00003 00100
				LIA EXTENT LOWER UPPER CCCHH CCCHH
				00001 00002

Figure 19. Volume Table of Contents - Volume Information

VOLUME TABLE OF CONTENTS 202020

FILE NAME	FILE SERIAL NO.	VOL SEQ NO.	CREATION DATE	EXPIRATION DATE	EXT CNT	FILE TYPE	REC FMT	WRITE DISK CHECK	BLCK LNTH	RECD LNTH	EXT TYPE	EXT SEQ NO.	LOWER LIMIT CCCHH	UPPER LIMIT CCCHH
SYSTEM/360 MOD 20 DPS WORK FILE 2	202020	0001	67-150	67-150	01	4000	F	U C	00240	00080	1	00	09800	10000

Figure 20. Volume Table of Contents - File Information

written in the track address portion (cchh) of the count fields. Furthermore, bit seven of the flag byte is changed from 0 to 1, indicating an alternate track.

For example, suppose track 7 on cylinder 53 has defective areas and track 0 of cylinder 1 is assigned as the alternate. The track address portion of the count field of every sector on cylinder 53, track 7, will be filled with the address of cylinder 1 track 0. The track address portion of the count fields of the alternate track (cylinder 1, track 0) will contain the address of cylinder 53, track 7.

The flag byte in each sector of the bad track will be changed from all binary zeros to 00000010; and the flag byte in each sector of the alternate track is changed from all binary zeros to 00000001.

Analysis continues until the entire disk pack has been processed. Defective tracks are listed on the system logging device to provide you with a record of the condition of the disk pack. If more than thirty defective tracks are found, the program terminates immediately. When analysis has been completed, each track contains a count field at the beginning of each sector. The data field for all sectors of good tracks contains binary zeros.

Volume Label Creation

The Initialize-Disk Utility program formats cylinder 0 and the Volume Table of Contents as shown in Figure 21. The program writes the volume label (which you have to provide in the VOL control statement) on cylinder 0, track 1, sector 0.

VTOC Format Creation

The Initialize-Disk program pre-formats the Volume Table of Contents (VTOC) by writing the (Format 4) file label for the VTOC itself in the first record location. The VTOC control statement indicates the location on the disk pack in which the VTOC is to be placed.

The standard location of the VTOC is on cylinder 0, and extends over tracks 2-9 of the cylinder. The VTOC can appear on any cylinder, but cannot exceed 10 tracks. A VTOC placed anywhere other than in the standard location may start at any track. The first label begins in the first sector of the first track assigned.

Each record of the VTOC contains a 135-byte data field. The first record of the VTOC is reserved for a specific record - the file label (Format 4) for the Volume Table of Contents itself (see Appendix B).

Cylinder	Track	Sector	Bytes	Contents
0	0	0-9		reserved (for initial program loading)
0	1	0	1-80	Volume label, field 5 pointing to the VTOC file label (cyl. X, track Y)
0	1	1-9		used by Job Control to store label information (LIA)
	2	0-9		used by Job Control to store label information (opt.)
x	y	0	1-135	VTOC file label (Format 4)
			136-270	Binary zeros
		1		Beginning of disk data file labels (see Appendix B)

Figure 21. Formats of the Volume Table of Contents and Cylinder 0

The following job-control and utility control statements are used with the Initialize-Disk Utility program.

Job Control Statements

Statement	Use	Operand Entries for Initialize-Disk
// JOB	Required	Program name = INTDSK
// ASSGN	Required	Primary output device = SYSOPT
// ASSGN	Optional	Additional output devices = SYS002-SYS004
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

*Unless information is still in main storage.

Initialize-Disk Utility Modifier Statement

The format of the first statement is

Name	Operation	Operand
//b	UINb	Tt,CYLNDR=(m),VERIFY=(n),ERASE

To list one or more VTOCs, the following ASSGN statements are used:

// ASSGN	Required	Primary input device = SYSIPT
// ASSGN	Optional	Additional input devices = SYS002-SYS004
// ASSGN	Required	Primary output device= SYSLST

A detailed description of the job control statements is given in Appendix A.

Utility Control Statements

The first three statements comprise a set and, if present, must be submitted in the indicated order.

//bUIN	Optional for TPI	One per INTDSK job
	Required for TSI	One for each disk pack to be initialized
//bVTOC	Optional	One per disk pack or one per INTDSK job
VOL1	Required	One per disk pack being initialized under TPI
//bEND	Required	

Operand	Possible Forms	Default Specification	Explanation
Tt	TPI TSI TL	TPI	Type of initialization to be performed. For Primary Initialization (TPI), processing begins with cylinder 0 and ends with cylinder 102 or cylinder 202. For Secondary Initialization (TSI), processing begins with cylinder 103 and ends with cylinder 202. The List (TL) option causes one or more VTOCs to be listed. If used, it must be the only operand specified on the utility-modifier control statement.
CYLNDR=(m)	CYLNDR=(102) CYLNDR=(202)	102 for Model 12, 202 for Model 11	Last cylinder to be initialized.
VERIFY=(n)	n=1-256	VERIFY=(1)	The number of times the test pattern is to be written and verified during the surface analysis. Due to the amount of time required to execute the VERIFY option, verification should be requested only once.
ERASE	ERASE or blank	Blank do not ERASE	If ERASE is not specified, all the tracks already flagged as defective will remain flagged. If ERASE is specified, the flag of each previously recognized defective track is cleared; the track will be flagged again as defective only if the result of the surface analysis so dictates. For Secondary Initialization this operand will be ignored and the disk pack will be initialized in the same mode as the Primary Initialization of the first half of the pack.

VTOC Control Statement

The VTOC control statement provides the control information necessary to create the VTOC. The VTOC control statement can be written in either of two forms:

```
|Name|Operation|Operands                    |
|----|---------|----------------------------|
|//b |VTOCb    |STANDARD                    |
|----|---------|----------------------------|
|//b |VTOCb    |STRTADR=(ccchhh),           |
|    |         |  EXTENT=(n),LIA=(m)        |
```

Operand	Default Specifications	Explanation
STANDARD		Assume all default specifications.
STRTADR=	0000002	The beginning address of the VTOC (the cylinder and head number).
EXTENT=(n)	8	The number of tracks in the VTOC.
LIA=(m)	1	The number of reserved tracks for Job Control Label Information (m = 1 or 2).

Note: If LIA=(2) is specified, i.e., if two tracks are to be reserved as Label Information Area, the beginning of the VTOC (STRTADR=) must be cylinder 0 track 3 or higher.

Volume-Label Statements

A volume-label (VOL) control statement containing an IBM standard volume label must be prepared for each disk pack to be initialized under TPI. It must not be present for TSI or TL options. The card must contain an exact image of fields 1-8 of the 80-byte label.

The format of the VOL control statement is shown in Figure 22.

The volume label is placed in cylinder 0, track 1, sector 0 of the disk pack. If more than one symbolic device is assigned, the order of the volume-label statements must correspond.

Field	Bytes	Contents	Explanation
1	1- 3	VOL	Label identifier
2	4	1	Indicates standard volume label
3	5-10	Alphameric	Volume serial number. Unique ID for each volume.
4	11	0 or 1	Volume security (not used).
5*	12-21	binary	The first five bytes contain the starting address (cchhr) of the VTOC. The r is always 0. The last 5 bytes are blank.
6-7	22-41	reserved	To be left blank
8	42-51	Alphameric	Installation code Owner's name-and-address code.
9	52	Volume protection byte (1 byte): X'40' or X'0F'	Protection for printer-keyboard support: 40 = volume not protected 0F = volume protected
10	53-54	End pointer (2 bytes, binary)	Contains the disk address (hr) of the last permanent label in the LIA.
11	55-56	Label count (2 bytes, binary)	Number of permanent label blocks in the LIA.
12	80	Extent indicator (1 byte)	Contains number of LIA tracks (1 or 2).

*Note: Field 5 is left blank in the Volume-Label Statement.

Figure 22. Format of the VOL Control Statement

END Control Statement

The END control statement indicates that no more disk packs are to be initialized. The format of the END control statement is

Name	Operation	Operand
//b	END	

Initialization of a Disk Pack

If you want to initialize a disk pack that has never before been used on a Model 20, the ERASE option must be specified on the utility-modifier control statement. Since the default specification is 'no erase', the utility-modifier statement must be present with ERASE specified.

Reinitialization

The reinitialization procedure should involve Primary Initialization (TPI) with CYLNDR=(102) for disk packs to be used on IBM 2311 Model 12 Disk Drives; and Primary Initialization with CYLNDR=(202) for disk packs to be used on Model 11 Drives.

Since Primary Initialization performs surface analysis on the alternate track area (cylinders 1-3), initializing the disk pack to be used on the Model 11 with TPI and CYLNDR=(102) may result in assigning as an alternate to a track within cylinders 0-102 one which is still assigned to a track within cylinders 103-202. When this occurs, the second half of the disk pack (cylinders 103-202) must be considered unusable.

Reinitializing a disk pack to be used on the Model 11 with Secondary Initialization (TSI) is not recommended, since a file may have extents within cylinders 0-102 and cylinders 103-202. In this case, the entire file (including the portion within cylinders 0-102) will be deleted when the file expiration date is reached. To perform TSI safely, the Clear-Disk Utility Program should first be executed over cylinders 103-202, to clear all expired files.

1. Primary Initialization of two disk packs for use on IBM 2311 Model 12 Disk Drives. Verification is to be performed once. Erase existing flags on defective tracks and reflag, if necessary. Volume serial numbers are to be 1275 and 1276. The installation code is ABCD007USA. The VTOC on the first pack is to be on cylinder 0, tracks 2-9. The VTOC on the second pack is to be on cylinder 52, tracks 0-6.

The utility control statements needed are shown in Figure 23.

Two ASSGN statements must be present. The first assigns a disk drive as primary output device (symbolic device SYSOPT). The second assigns the alternate disk drive (SYS002).

Since the VTOC for the first pack is to be on cylinder 0, tracks 2-9, the default specifications apply and the STANDARD option can therefore be used. With the

second pack the position of the VTOC has to be explicitly stated, since it is not the standard location.

For a description of the standard volume label format see Appendix B.

2. Secondary Initialization of a disk pack for use on IBM 2311 Model 11 Disk Drives. Verification is to be performed once.

```
//bUINbTSI,CYLNDR=(202),VERIFY=(1)
//bEND
```

For Secondary Initialization the VTOC statement and the volume-label statement are not required. One ASSGN statement must be present. It assigns SYSOPT as primary output device. The ERASE feature will be applied as specified for the Primary Initialization of the disk pack.

3. List the contents of the VTOC of two disk packs.

```
//bUINbTL
//bEND
```

Two ASSGN statements that assign two disk drives as primary and additional input (symbolic units SYSIPT and SYS002) must be present.

IBM System/360 Assembler Coding Form

PROGRAM: INITIALIZE - DISK UTILITY CONTROL STATEMENTS

```
// UIN TPI, CYLNDR=(102), VERIFY=(1), ERASE
// VTOC STANDARD
VOL100127S1                                    ABCD007USA
// VTOC STRTADR=(00520000), EXTENT=(7)
VOL100127G1
// END
```

Figure 23. Initialize-Disk Utility Control Statements

Clear Disk Utility Program (CLRDSK)

This program clears the data areas of good tracks in one or more IBM 1316 Disk Packs on one or more disk drives (max. two drives for Submodels 2 and 4 and four drives for Submodel 5) placing a character in the cleared areas. This character you have to specify in a control statement in either hexadecimal or EBCDIC format. The extents are specified in XTENT job-control statements. The size of each cleared data area can range from a single track to the entire disk pack. All ten sectors in each designated track are cleared.

The program can also clear the areas of indexed-sequential organized files, but note that the extent type in the XTENT statement must be specified as a 1 also for the areas defined by the cylinder-index extent and the independent-overflow extent (see Appendix A. Job Control Statements - XTENT Statement).

If the extent includes a defective track as indicated by the flag byte (see Initialize Disk Utility Program and Appendix E), the referenced alternate track data areas are cleared.

The area specified in the XTENT statements is first checked to determine whether it contains part or all of an unexpired file. If the area overlaps with any part of an unexpired file, the program halts. By continuing, you indicate that the entire file is to be deleted. Thus more than one file may be deleted with a single Clear Disk job. However, the fill character will be inserted only in the specified extents.

Note: When the Clear Disk Utility program is executed under a Monitor which allows inquiry requests, a temporary file name consisting of 44 dollar signs is created and deleted again at the end of the job. If an identical file name is already present in the VTOC of the disk pack to be cleared, the job must not be prematurely terminated to ensure the deletion of the temporary file name. A file name consisting of 44 dollar signs must not be used in an inquiry program interrupting the Clear Disk program.

The following job-control and utility control statements are used with the Clear Disk program:

Job Control Statements

Statement	Use	Operand Entries for Clear Disk Utility
// JOB	Required	Program name = CLRDSK
// ASSGN	Required	Primary output device= SYSOPT
// ASSGN	Optional	Alternate output devices = SYS002-SYS004
// ASSGN	Optional	Logging device = SYSLOG (see LOG Statement)
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// VOL	Required	File name = UOUT
// DLAB	Required	
// XTENT	Required	More than one XTENT statement may be used.
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

*Unless information is still in main storage.

For a detailed description of the job-control statements refer to Appendix A.

Utility Control Statements

//bU	Optional
//bEND	Required

CLEAR DISK UTILITY-MODIFIER STATEMENT

This utility-modifier statement has two operands. The first defines the fill character and the second the option to make a write-disk check. The statement has the following format:

Name	Operation	Operand
//b	UCLb	fill,Ox

Definition:

//bUCL Clear-Disk utility identifier. The CL is optional.

fill This operand specifies the fill character:

C'a' If the fill character is specified as EBCDIC, a C is punched followed by the EBCDIC fill character enclosed in apostrophes.

X'aa' If the fill character is specified in hexadecimal format, an X is punched followed by two hexadecimal characters enclosed in apostrophes.

Ox Write-disk check

OY indicates write-disk check

ON indicates do not write-disk check.

The default specification is OY. The write-disk check option is used to read back and verify data immediately after it has been written on a disk. If the data cannot be written properly, a message is printed, the system halts, and the program is terminated. It is recommended that you use the write-disk check option.

If the utility-modifier statement is omitted, the following default specifications are used:

```
//bUbX'00',OY
```

When the card is present, the first operand (fill) must be specified.

Four forms of the statement are acceptable:

```
//bUCLbC'a',OY
//bUCLbC'a',ON
//bUCLbX'aa',OY
//bUCLbX'aa',ON
```

END CONTROL STATEMENT

The END control statement indicates that no more disk packs are to be cleared. The format of the END control statement is

Name	Operation	Operand
//b	END	

Sample Problems

1. Clear a disk area and insert zeros specified in EBCDIC. Perform a write-disk check.

```
//bUCLbC'0',OY
//bEND
```

or

```
//bUbC'0'
//bEND
```

The proper disk drive has been assigned as the primary system output device with an ASSGN statement. An XTENT statement is used to indicate the area to be cleared on the disk pack.

2. Clear disk areas on a pack and fill each byte with bits 00000001. Do not make a write-disk check.

```
//bUCLbX'01',ON
//bEND
```

The proper disk drive has been assigned as the primary output device, and the areas to be cleared are indicated on XTENT statements.

3. Clear two disk packs. Fill with binary zeros. Make a write-disk check. The drive with the first disk pack has been assigned as primary output device, and that with the second as alternate output device.

```
//bEND
```

One ASSGN statement and one XTENT statement are needed for each disk pack. No utility-modifier statement is needed, since all job requirements are met by the default specifications.

Alternate-Track Assignment Utility Program (ATASGN)

This program assigns an alternate track to a defective track in an IBM 1316 Disk Pack, and transfers the data contained on the defective track to the alternate track. It also performs a surface analysis of the defective track to determine whether the error is permanent. This program does not detect defective tracks but assumes that you are already aware that a specific track is defective.

Locating a defective track is done by the Initialize Disk program: If a bad track is found, an alternate track is set up and a message is written.

To assign an alternate track, the program does the following:

1. It checks the VTOC file label (see Initialize Disk Utility Program) for the location of the next good alternate track available. If one is found, the data is transferred to it. (See item 2 below.) If no alternate tracks are available, a message is printed and the job is terminated.

2. It prints a message with the location of the alternate track. Data on the defective track is transferred, sector by sector, to the alternate track. A sector containing an error is transferred as it is read. If specified by a control statement, data in the track that contains an error is printed in hexadecimal format.

3. It places the address of the alternate track in the track-address portion of the count field of each sector of the defective track. This address is used as a pointer to the alternate track. Also, the address of the defective track is written in the track-address portion of the count field of each sector of the alternate track. The flag byte in each sector of the alternate track is changed to 00000001.

4. It checks the surface of the defective track. If the defect is permanent, the flag byte in each sector of that track is changed to 00000010 to indicate a defective track. The data remains on the alternate track and the job is complete.

   If the defect was temporary, and the FORCED option is not specified, the data is transferred back to the original track. Any errors written on the alternate track will be transferred back as errors. The program does not correct errors in the data. The flag bytes of the alternate track are reset to binary zeros to indicate that the track is unassigned. If the FORCED option is specified, the data is not transferred back to the original track.

5. The VTOC file-label is modified to contain the address of the next available alternate track.

The following job-control and utility control statements are used with the Alternate-Track Assignment program:

Job Control Statements

Statement	Use	Operand Entries for Alternate-Track Assignment Utility
// JOB	Required	Program name = ATASGN
// ASSGN	Required	Primary output device= SYSOPT
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// CONFG	Required*	Main-storage capacity
// DATE	Required*	Year and day
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

*Unless information is still in main storage.

For details on the job-control statements refer to Appendix A.

Utility Control Statements

//bU	Required	(More than one may be submitted)
//bEND	Required	

ALTERNATE-TRACK ASSIGNMENT UTILITY-MODIFIER STATEMENT

This utility-modifier statement contains the address of the defective track and an option indicating whether sectors containing errors are to be printed. The general format of the statement is

Name	Operation	Operand
//b	UATb	R=(ccchhh),Ox,VERIFY=(n), FORCED

The program handles the transfer of data only from a single track; therefore, one utility-modifier statement per track is used.

Operand definition:

//bUATb — Statement identifier. The AT is optional.

R=(ccchhh) — Defective track location. This operand must be present and it must be the first operand. The address consists of a cylinder number (0000 or 0004-0202 for Model 11; 0000 or 0004-0102 for Model 12) and a head number (000-009). The address must be enclosed in parentheses. If an alternate track becomes defective, a second alternate track is assigned. But in no case should the address of an alternate track (cylinders 1-3) be specified. The address in parentheses is not that of the alternate track but that of the original defective track.

Ox — Print operand. It has two forms:

- OY if the data in a track containing invalid records is to be printed;
- ON if the data in this track is not to be printed.

This operand has a default value of OY.

VERIFY=(n) — The value n indicates the number of times the program should write and verify a test pattern during surface analysis to determine whether the defect is permanent or temporary. The default specification is VERIFY=(1). Any value between 1 and 256 may be specified.

FORCED — This operand indicates that the defect is to be considered permanent. The default specification is the absence of FORCED.

END CONTROL STATEMENT

The END control statement indicates that no more alternate tracks are to be assigned. You must submit this statement. The format of the END control statement is

Name	Operation	Operand
//b	END	

SAMPLE PROBLEMS

1. Track 4 of cylinder 051 has been found to be defective. Assign an alternate track and transfer the data. Print the track containing the errors.
   The required utility control statements are

```
//bUATbR=(0051004),OY
//bEND
```

or

```
//bUbR=(0051004)
//bEND
```

   Both pairs of statements achieve the same results. The second pair uses the default specification for the print option, and omits the optional program initials.

2. Assign an alternate track for track 0 of cylinder 31, and transfer the data from the defective track. Do not print the invalid track.

   The necessary utility control statements are

```
//bUATbR=(0031000),ON
//bEND
```

Disk Dump Utility Program (DDUMP)

The Disk Dump utility program prints the contents of disk data (record) fields and count fields. The output is in hexadecimal notation and in character format. All kinds of data can be dumped, regardless of labels or extents.

The following information for the program must be provided via console switches:

1. Drive number.
2. Lower limit of area to be dumped.
3. Termination of program (by entering X'E' or X'F').

After the program has been loaded, an initial halt occurs indicating that new information is to be entered on the console switches. A halt does not automatically occur after an extent has been dumped; therefore, unless X'E' or X'F' is entered on the console switches, the program continues dumping one track after the other until the end of the pack is reached.

JOB-CONTROL STATEMENTS

The Disk Dump utility program can be either disk-resident or card-resident. If you use the disk-resident version, the program can be executed as either mainline or inquiry program (each under supervision of the DPS control programs). When you use the card-resident resident version, it can be executed under DPS, or it can be used as a stand-alone program consisting of an object deck which is preceded by an absolute loader. An object deck can be obtained with the CSERV program (See the publication IBM System/360 Model 20, DPS Control and Service Programs, Form C23-9006).

The following control statements are used with the Disk Dump utility program (unless it is run as a stand-alone program):

Statement	Use	Operand Entries for Disk Dump
// JOB	Required	Program name = DDUMP
// ASSGN	Required	Primary output device = SYSLST
// ASSGN	Optional	Logging device = SYSLOG (See LOG statement)
// DATE	Required	
// LOG	Optional	
or		
// NOLOG		
// EXEC	Required	

For a detailed description of the job-control statements refer to Appendix A.

The Disk Dump program does not need any utility control statements.

DESCRIPTION OF CONSOLE SWITCH INPUT

Data switch 1: If this switch is changed after the initial halt occurs, the program starts with the new information from the console switches. If it is changed to a value between X'0' and X'D', the information in data switch 2 and the register data switches is analysed. Then a new page is started and the extent is dumped as specified in the switches. If data switch 1 is changed to X'E' or X'F', the program is terminated.

Data switch 2: This switch specifies the drive number, which can be 1,2,3, or 4.

The register data switches specify the lower limit (cchr) of the extent to be dumped:

Register data switches 1 and 2 specify the cylinder number in hexadecimal notation (cc).

Register data switch 3 specifies the head number in hexadecimal notation (h).

Register data switch 4 specifies the record number in hexadecimal notation (r).

SAMPLE PROBLEM

Dump the contents of cylinder 0, head 1, of drives 1, 2, and 4 and the contents of cylinder 106, head 1, record 5, of drive 1.

1. Call the Disk Dump program and wait for initial halt.
2. Specify the following values on the console switches:

D1	D2	R1	R2	R3	R4
0-D*	1	0	0	1	0

*Any value between X'0' and X'D'

3. Press START.

4. Change D2 to 2 and change D1 after completion of output from drive 1 to a value between X'0' and X'D'.

5. Change D2 to 4 and change D1 after completion of output from drive 2 to a value between X'0' and X'D'.

6. Change D2 to 1, R1 to 6, R2 to A (decimal 106 = hexadecimal 6A), R3 to 1, and R4 to 5. Change D1 after completion of output from last extent to a value between X'0' and X'D'.

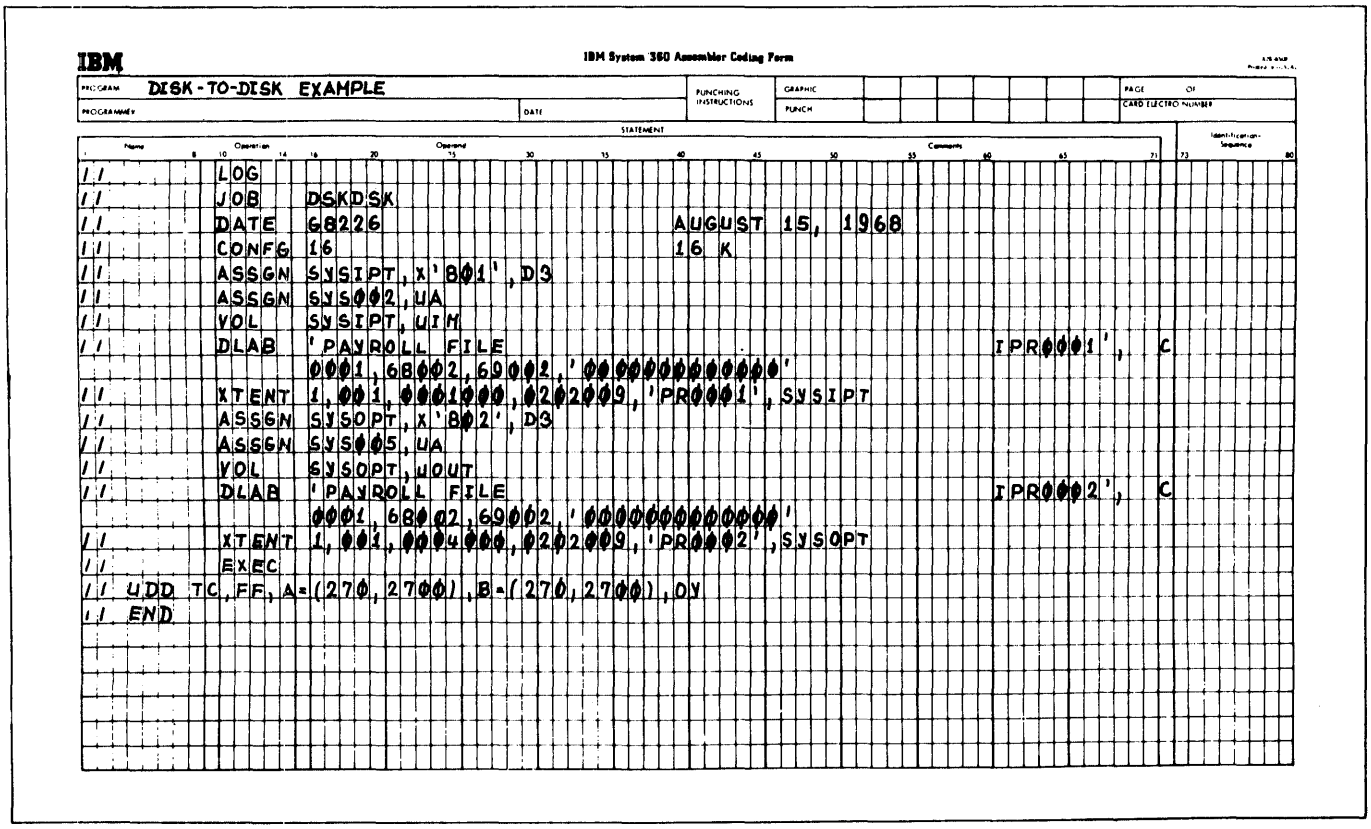
7. When all information from drive 1, cylinder 106, head 1, record 5 has been retrieved, change D1 to X'E' or X'F' to terminate the job.

Sample Problem Shown in Detail

Transfer a file from one disk pack to another to create a back-up file.

For this example, it is assumed that the disk pack onto which the file is to be copied contains the volume serial number PR0002. The records of the file are 270 bytes long, 10 records to a block.

Control statements necessary to define this problem are shown in Figure 24.



● Figure 24. Control Statements for Sample Disk-to-Disk Problem

Job-control statements are required to prepare the system for the execution of a utility program.

All job-control statements contain three fields. The first field (name field) always contains // in columns 1-2 to identify the card as a job-control statement. The name field is followed by at least one blank column and then by the operation field, up to five columns long, specifying the operation code (e.g., JOB or ASSGN). The second field must be separated from the third by at least one blank column. The third field is the operand field. Depending on the type of statement, this field is blank, one operand, or contains a series of operands separated from each other by commas.

For ease of reference, the job control statements are described in alphabetical order below.

ASSGN Statement

One ASSGN statement must be prepared for each input/output device used. It serves to associate a physical input or output device with a symbolic device name. An ASSGN statement may be omitted when the information on it is still in storage from a previous job. The statement contains a symbolic device address, the physical device address and type, and specifications for 7-track or 9-track tape when applicable.

The use of two physical tape devices with multiple-reel input or output makes it possible to mount additional reels without delaying the operation of the program.

When one disk drive serves as both input and output device, the same physical device address is used in two ASSGN statements with two different symbolic device addresses.

The statement has the following format:

Name	Operation	Operand
//	ASSGN	SYSxxx,X'cuu',dd,X'ss'

ASSGN	ASSGN statement identification	
SYSxxx	Symbolic device address SYSIPT -- primary input device SYSOPT -- primary output device SYSLOG -- control-statement logging device (printer) SYS000-SYS007 -- alternate input and output tape and disk drives	
X'cuu'	Physical device address, in hexadecimal notation	See Figure 25 for operand specifications
dd	Type of device	
X'ss'	Tape characteristics, in hexadecimal notation (This field is entered only when a tape device is being assigned)	

For an IBM 2415 with the 9-Track Compatibility special feature (#5320 or #7135) installed, a tape characteristic specification (ss) must be assigned for a 9-track output drive. A tape characteristic must always be assigned for 7-track tape.

```
//ASSGN SYSOPT,X'785',T2,X'C8'
```

assigns a 9-track tape drive with the device address 85 as the primary output device. The tape will be written at 800 bytes per inch. (Model 4, 5, or 6 with 9-Track Compatibility feature is assumed).

```
// ASSGN SYSIPT,X'781',T1,X'68'
```

address 81 as the primary input device. This tape has been recorded at 556 bpi (bytes per inch) with even parity. Every 6-bit character from the 7-track tape is translated to an 8-bit EBCDIC byte in main storage.

```
// ASSGN SYSIPT, X'801',D4
```

assigns a 2311 Disk Storage Drive Model 12 with the device address 01 as the primary input device.

Device	Device Address (cuu)	Type (dd)	Tape Characteristic (ss)
2311 Disk Storage Drives			
Model 11	8xx(xx=01-04)	D3	
Model 12	8xx(xx=01-04)	D4	
1403 Printer	400	L1	
2203 Printer	400	L3	
1442 Card Punch	300	P2	
2520 Card Punch	200	P3	
2501 Card Reader	100	R4	
2560 MFCM Primary Feed	200	R6	
2560 MFCM Secondary Feed	200	R7	
2520 Card Read-Punch	200	R5	
2415 7-track Tape Drives	7xx(xx=00-FF)	T1	See table below
2415 9-track Tape Drives	7xx(xx=00-FF)	T2	C8 (if 800 bpi) C0(if 1600 bpi)

For 7-track tape: BPI (Bytes Per inch)	Parity	Translate Feature	Convert Feature	(ss)
200	odd	off	on	10
	even	off	off	20
	even	on	off	28
	odd	off	off	30
	odd	on	off	38
556	odd	off	on	50
	even	off	off	60
	even	on	off	68
	odd	off	off	70
	odd	on	off	78
800	odd	off	on	90
	even	off	off	A0
	even	on	off	A8
	odd	off	off	B0
	odd	on	off	B8

Figure 25. Operand Specifications for the ASSGN Statement

If you do not wish to use the alternate input or alternate output tape (SYS000 and SYS001, respectively), you must submit ASSGN statements to "unassign" the logical devices.

```
// ASSGN SYS000,UA
// ASSGN SYS001,UA
```

Failure to unassign these tapes will cause an "INVALID ALTERNATE ASSIGNMENT" error halt if SYS000 and/or SYS001 are not already unassigned from a previous job.

CONFG Statement

The CONFG statement defines the amount of main storage available. It is required unless this information is still in main storage from a previous job. If the CONFG statement is submitted for a job operating under a Monitor with inquiry support, the statement is ignored, i.e., a halt occurs and normal processing continues when START is pressed on the CPU. The CONFG statement has the following format:

Name	Operation	Operand
//	CONFG	xx

CONFG CONFG statement identification

xx A decimal representation of main storage capacity in K bytes. The following codes are valid:

12 (for 12,288 bytes)
16 (for 16,384 bytes)
24 (for 24,576 bytes)
32 (for 32,768 bytes)

For example, to represent a Model 20 with a 12,288-byte main storage capacity, the CONFG statement will be

```
// CONFG  12
```

DATE Statement

The DATE statement contains the current date. It is required unless this information is still in storage from a previous job. The format is

Name	Operation	Operand
//	DATE	yyddd

DATE DATE statement identification

yy Last two digits of the year

ddd Number of the day of the year (001-366)

For example, to specify August 31, 1967, the 243rd day of the year, the statement is

```
// DATE  67243
```

DELET Control Statement

The DELET control statement is used to remove one or more permanent file labels from the Label Information Area.

The DELET control statement has one of the following formats:

Name	Operation	Operands
//	DELET	one or more operands
//	DELET	

DELET statements of the first format above may contain one or more file names separated by commas. A blank must follow the last operand. The field must not extend beyond column 71 of the card.

DELET statements of the second format above cause blanks to be moved into the Label Information Area, (i.e., the area is cleared).

Note: A DELET statement must precede the first VOL statement within a Job Control deck. Otherwise, an error halt occurs.

DLA Statement

The disk-label statement (completed in a continuation statement) contains file label (Format 1) information for disk label checking and creation. This statement must immediately follow its related volume (VOL) statement. The DLAB statement has the following format:

Basic Statement:

Name	Operation	Operand
//	DLAB	'Model 20 standard disk file label fields 1-3',P

DLAB	DLAB statement identification.
'(apostrophe)	Start of disk file label fields 1-3.
Model 20 standard disk file label fields 1-3	(See Appendix B for details of file label fields).
'(apostrophe)	End of disk file label fields 1-3.
P	Optional; if specified, the label information on disk is permanent.

Continuation Statement:

The entire DLAB information cannot be entered in one card. Therefore, a character (any character) must be punched in column 72 to indicate that a continuation statement follows. The continuation statement contains blanks in columns 1-15, and fields 4-6 of the Model 20 standard disk file label starting in column 16. (See Appendix B for a complete description of the fields in the Model 20 standard disk file label). The continuation statement has the following format:

```
xxxx,yyddd,yyddd,'xxxxxxxxxxxxx'
```

xxxx	Volume sequence number. This 4-digit number is the decimal equivalent of the 2-byte binary volume sequence number in Field 4 of the label.
yyddd, yyddd	The file creation date followed by the file expiration date. These two 5-digit numbers are the decimal equivalents of the 3-byte discontinuous binary dates in Fields 5 and 6 of the label. yy is the year (00-99) ddd is the day of the year (001-366).
'xxxxxxxxxxxxx'	System code (optional). This 13 character field must be enclosed in apostrophes.

DSPLY Control Statement

If SYSLOG is assigned, the DSPLY statement causes the listing of all permanent labels contained in the Label Information Area. If SYSLOG is unassigned or NOLOG is encountered, the DSPLY statement causes no print request.

The DSPLY control statement has the following format:

Name	Operation	Operands
//	DSPLY	

If no permanent labels are present in the Label Information Area on disk, and the DSPLY statement is encountered, the message NO LABEL FOUND is listed on the printer assigned to SYSLOG before processing continues.

The listing consists of the file name, the expiration date, (i.e., the format used in the DLAB statement), the symbolic device address, the type of extent, and the lower and upper extent limits in decimal notation.

EXEC Statement (Required)

The EXEC statement is the last statement in the group of job-control statements. It indicates that execution is to begin. Its format is

Name	Operation	Operand
//	EXEC	

FILES Statement

The FILES control statement is used to position a reel of magnetic tape at a specific file by skipping a specified number of tapemarks. The statement has the following format:

Name	Operation	Operand
//	FILES	SYSxxx,n

FILES FILES statement identification

SYSxxx Symbolic device address of tape drive concerned

n Number of tapemarks to be skipped

The number of tapemarks to be skipped (n) is counted from the load point, and may be any number from 1-999. It must take into account the tapemarks associated with labels, as well as file tapemarks.

JOB Statement

This is the first control statement for a job, and contains the name of the program to be run. The operand field must contain the name of the program. The statement format is:

Name	Operation	Operand
//	JOB	xxxxxx

JOB JOB statement identification
xxxxxx Name of program to be run, such as CLRDSK, DSKTAP, etc.

LOG or NOLOG Statement (Optional)

To log is to list control statements on the printer assigned to the symbolic device SYSLOG. If a LOG statement is submitted, job-control statements will be listed until a NOLOG statement is read. A LOG statement used in one job will cause logging in all subsequent jobs until a NOLOG statement is read. Conversely, if a NOLOG statement is submitted, no job-control cards are listed in this and all subsequent jobs until a LOG statement is encountered.

The formats of the two statements are:

Name	Operation	Operand
//	LOG	
//	NOLOG	

Utility programs will list the various operand specifications established by the job -- such as record size, block size, etc. -- whenever SYSLOG is assigned.

OPTN Control Statement

The OPTN control statement is used to initiate error statistics.

The OPTN control statement has the following format:

Name	Operation	Operand
//	OPTN	TES

OPTN OPTN statement identification.

TES Tape error statistics are to be initiated.

Note: Error statistics will only be listed on the printer when a LOG statement is issued and the Job Control program is reloaded after execution of the problem program.

TPLAB Statement

One TPLAB statement is required for each input and each output tape file on which standard file label processing is to be done. It must immediately follow the appropriate VOL statement and must contain the file label associated with the symbolic device described in that VOL statement. This statement contains an image of a portion -- fields 3 to 10 -- of the IBM standard tape file label. The format of this label is presented in Appendix B. Label fields 3-10 are always punched just as they appear in the label. These are the only fields processed.

The format of the TPLAB statement is:

Name	Operation	Operand
//	TPLAB	'IBM standard file label fields 3-10'

TPLAB	TPLAB statement identification
'(apostrophe) IBM standard tape file label fields '(apostrophe)	Start of tape file label (See Appendix B, Figure 28, for details of tape file label fields). End of tape file label

Example:

```
//  VOL  SYSIPT,UIN
//  TPLAB  '1967bSALESbbbbbbbbABCDEF000000
    00000701b67140b68031'
```

would be used to check that the tapes mounted on the input devices contain a file labeled "1967 SALES" that had been generated on May 20, 1967 and would not expire until January 31, 1968.

UPSI Statement

The UPSI control statement is used to set switch indicators in the problem program. The format is as follows:

Name	Operation	Operand
//	UPSI	nnnnnnnn

Each n represents one bit of the UPSI byte in the Monitor communication region. (For details on the UPSI byte refer to the SRL publication IBM System/360 Model 20 Disk Programming System, Control and Service Programs, Form C24-9006.)

VOL Statement

One VOL statement is required for each input and each output tape file on which standard file label processing is to be performed and for each disk input and output file. The statement contains the symbolic device name associated with the file (see ASSGN statement), and a logical file name (left-aligned). The statement format is

Name	Operation	Operand
//	VOL	SYSxxx,fffff

VOL	VOL statement identification
SYSxxx	The symbolic device name
fffff	File name (left-aligned) as specified in the individual program.

XTENT Statement

An XTENT statement defines an area, or extent, of a disk file, on a disk pack. An extent limit can only refer to a track, not to a sector within that track.

XTENT statements cannot reference the alternate track area (cylinders 1-3 and cylinder 0 track 1). When the program encounters a defective track (within the extent) for which an alternate track has been assigned, the disk utility program automatically references the appropriate alternate track and then continues at the next track within the extent.

One or more XTENT statements must follow each DLAB statement. Figure 26 defines the operands which are used with the XTENT statement. The XTENT statement has the following format:

Name	Operation	Operand
//	XTENT	type,sequence,lower limit, upper limit,'volume serial no.',SYSxxx

Example:

```
//  XTENT 1,000,0010000,0012009,'124356',
    SYSIPT
```

would be used to define an extent that occupies the thirty tracks on cylinders 10, 11, and 12 of a disk input file.

The program checks to determine whether the extents specified in XTENT statements for an input file coincide with those in the file label. If they do, processing continues normally. If the XTENT statement specifications extend beyond the file limits, the program halts.

If an extent specified for an output file overlaps with any unexpired file on the disk pack, a programmed halt occurs.

To process a multi-volume file, submit XTENT statements to define the extents of each intermediate volume (See Appendix J).

Operand	Number of Columns	Explanation
type	1	Extent Type - contains a 1 (data extent)
sequence	3	Extent Sequence Number Contains a 3-digit number from 000 to 255, indicating the sequence number of this extent within a multi-extent file.
lower limit	7	Contains the lowest address of the extent in the decimal form ccccchhh, where cccc= cylinder number (0000 to 0202 for Model 11, or 0000 - 0102 for Model 12). hhh = head number (000 to 009)
upper limit	7	Contains the highest address of the extent, in the same form as the lower limit. (Cannot be less than lower limit).
'volume serial no.'	6	Six-column alphameric field, punched within apostrophes. The number is the same as the number in the volume label on the "SYSxxx" unit. (See Appendix B).
SYSxxx		Symbolic address of the Disk Storage Drive. (See ASSGN Statement).

Figure 26. Operand Values for the XTENT Statement

Appendix B. Volume and File Labels

The IBM System/360 Model 20 DPS Utility programs process Model 20 standard disk labels and standard IBM System/360 tape labels.

There are two kinds of standard labels: volume labels which identify the physical disk or reel of tape, and file labels which identify the data file. A disk file label is stored in an area reserved for file labels and contains the track boundaries (or extents) of the file. A tape file label, however, consists of two parts: the header which precedes the file and the trailer which follows the file. With multi-volume tape files, a header and a trailer must precede and follow each volume.

Non-standard tape labels are not processed by Model 20 utility programs. They will be bypassed. Standard disk labels must be present.

Volume Labels -- Disk

IBM standard disk volume labels are written by the Initialize Disk Utility program. The Initialize Disk program provides for one IBM standard volume label per disk pack. This label must appear on all disk packs to be processed by the Disk Utility programs. The label format is described in Figure 27. Other disk volume labels are not allowed.

IBM standard disk volume labels on all input and output packs are always checked by the utility programs to ensure that the correct volume for a job is used. A halt occurs if an incorrect volume is found. If no IBM standard disk volume label is found, the job is terminated.

Volume Labels -- Tape

The 80-byte IBM standard tape volume label (Figure 28) is written by the Initialize Tape Utility program. See the publication IBM System/360 Model 20, Disk and Tape Programming Systems, Tape Utility Programs, Form C26-3808. The programs also allow the presence of up to seven additional tape volume labels. These additional labels are also 80 bytes long. Field 1 contains VOL and field 2 contains a value representing the position of the volume label relative to the IBM standard tape volume label. The remaining 76 bytes of the label are unformatted and can be filled in by yourself.

In programs using tape input where you specify standard label checking, the IBM standard tape volume label is checked on each input reel to determine whether it is the correct volume for the operation. If

Field	Bytes	Contents	Explanation
1	1- 3	VOL	Label identifier
2	4	1	Indicates standard volume label
3	5-10	Alphameric	Volume serial number. Unique identifier for each volume .
4	11	0 or 1	Volume security (not used).
5*	12-21	binary	The first five bytes contain the starting address (cchhr) of the VTOC. The r is always 0. The last 5 bytes are blank.
6-7	22-41	reserved	To be left blank
8	42-51	Alphameric	Installation code Owner's name-and-address code.
9	52	Volume protection byte X'40' or X'0F'	Protection for printer-keyboard support: 40 = volume not protected 0F = volume protected
10	53-54	End pointer (2 bytes, binary)	Contains the disk address (hr) of the last permanent label in the LIA.
11	55-56	Label count (2 bytes, binary)	Number of permanent label blocks in the LIA.
12	80	Extent indicator (1 byte)	Contains number of LIA tracks (1 or 2).

*Note: Fields 5 and 9-12 are left blank in the Volume-Label Statement.

Figure 27. IBM Standard Disk Volume Label Format

no standard volume label is found, the job is terminated. If an incorrect volume is found, an error indication is given with an option to change the reel and recheck, or to continue. If additional (user) tape volume labels (see above) are present, they are bypassed and not checked.

Field	Bytes	Contents	Explanation
1	1-3	VOL	Label identifier
2	4	1	Indicates first label within the volume label set
3	5-10	alphameric	Volume serial number. Unique ID for each volume
4	11	0 or 1	Volume security not used
5			
6	12-41	reserved	To be left blank
7			
8	42-51	alphameric	Owner's name-and-address code. Unique ID for installation
9	52-80	reserved	To be left blank

Figure 28. IBM Standard Tape Volume Label

File Labels -- Disk

IBM Model 20 standard disk file labels must also be used. Standard file labels are checked and written from information on the DLAB and XTENT control statements. No additional file labels or user labels are allowed.

A standard file label or set of file labels identifies a particular logical file, gives its location(s) on the disk pack, and contains information to prevent premature destruction of current files. The number and format of labels required for any one file depends on the number of separate areas of the pack (extents) used by the file.

Volume Table of Contents (VTOC)

All standard disk file labels are grouped together and stored in a specified area of the pack. Because each file label contains file limits, the group of labels on a pack is essentially a directory of all data records on the pack (or volume). Therefore, it is called the Volume Table of Contents (VTOC). The VTOC itself is a file of records (one or more standard label records per logical file) and is defined as such with its own file label. The label of the VTOC is the first record in the VTOC. This label identifies the file as the VTOC file and gives the file limits of the VTOC.

All labels belonging to one file on a disk pack must form a contiguous area within the VTOC, with the Format-1 label at the beginning of a sector. The gap produced by the deletion of a file label from the VTOC will be eliminated by shifting all labels which have disk addresses higher than the gap. Thus the available space within any VTOC will always form a contiguous area.

The VTOC is preformatted by the Initialize Disk program. When first preparing a disk pack you specify its location and length. It can be placed anywhere within the pack with the following restrictions.

1. It must be on cylinder 0, tracks 2-9, or within cylinders 4-202 for Model 11; or on cylinder 0, tracks 2-9, or within cylinders 4-102 for Model 12. (Cylinders 1-3 are used as the alternate track area.)

2. It must be one or more full tracks.

3. It cannot be more than 10 tracks.

The Initialize Disk Utility program preformats the entire VTOC by writing the label for the VTOC itself in the first record location.

Formats of Standard Disk File Label

All standard disk file labels are written in the VTOC. The field types within the labels written for data files follow four standard formats. The format of a label is identified by the value in byte 45 (Field 2). This is an EBCDIC value from 1 to 4 indicating label format 1 to 4.

Format 1. (See Figure 29.) This format is used for all logical files. It is always the first of the series of labels when a file requires more than one label on a disk pack.

The Format-1 label identifies the logical file (by a file name which you assign in the DLAB statement), and it contains file and data-record specifications, including the file expiration date. The extents onto which output files are to be written, are checked for expiration. If the label indicates that the file has expired, a new label is written from data in the appropriate job-control statements. If it has not expired, a message is printed, with the option to change the pack and check again, or to continue processing.

Field	Bytes	Contents	Explanation
1	1-44	alphameric	File name (must be identical to file name in DLAB statement)
2	45	1	Format identifier (Format 1)
3	46-51	alphameric	File serial number (must be identical to field 3 of volume label of first volume of data file)
4	52-53	binary	Volume sequence number (1-9999)*
5	54-56	binary	Creation date byte 54 year (0-99)* bytes 55+56 - day (1-366)*
6	57-59	binary	Expiration date byte 57 year (0-99)* bytes 58+59 - day (1-366)*
7a	60	binary	Extent count: Count of number of extents for this file on this volume
7b	61-62	reserved	To be left blank
8	63-75	alphameric	System code (optional)
9	76-82	reserved	To be left blank
10	83-84	xxx (hexadecimal)	File type (sequential organization)
11**	85	binary	Record format bits 0-1=10 indicates fixed-length bit 3=0 unblocked 3=1 blocked (Other bits not used by program)
12**	86	binary	File created using write-disk check. bit 0=1: yes 0=0: no (Other bits not used by program)
13	87-88	binary	Block length
14	89-90	binary	Record length
15**	91	binary	Key length (not used)
16**	92-93	binary	Key location (not used)
17	94	reserved	Data set indicators (not used)
18	95-98	binary	Secondary allocation (not used)
19	99-103	binary (cchhr)	Last used track and sector (not used) cc=cylinder number hh=head or track number. r=sector number
20	104-105	reserved	to be left blank

*Decimal equivalent
**Note: These fields are created for output files. The information is not used on input files.

Figure 29. Model 20 Disk File Label, Format 1 (Part 1 of 2)

Field	Bytes	Contents	Explanation	
21	106	hexadecimal	Extent type indicator 00 - next 3 fields do not indicate any extent 01 - extent containing your own data records	
22	107	binary	Extent sequence number (ascending sequence)	
23	108-111	binary (cchh)	Lower limit cc = cylinder number hh = head or track number	Extent Description
24	112-115	binary (cchh)	Upper limit cc = cylinder number hh = head or track number	
25-28	116-125	See fields 21-24	Additional extent	
29-32	126-135	See fields 21-24	Additional extent	

Figure 29. Model 20 Disk File Label, Format 1 (Part 2 of 2)

This label also provides the addresses for three separate disk areas (extents) for the file. (See fields 21-32 in Figure 25). If the file is scattered over more than three separate areas in one pack, one or more Format-3 labels are also required. In this case, the Format-3 labels follow the Format-1 label. If a logical file is recorded on more than one disk pack, the Format-1 label is always the first label for the file in the VTOC on each pack.

Format 2. This format is required for any file that is organized by the Indexed Sequential File Management System. Since the six file-to-file utility programs do not support indexed-sequential files, format 2 is not written, checked or used by the Model 20 DPS Utility programs.

Format 3. (See Figure 30.) If a logical file uses more than three extents on any pack, this format is used to specify the addresses of the additional extents. It is used only for extent information, and the 125 available bytes provide for as many as 12 extents. Several Format-3 labels may be used (when more than 15 extents of a file are located on one disk pack).

The Format-3 label is preceded by the Format-1 label for the logical file or by another Format-3 label. It is included as required on the first pack, or on additional packs if the logical file is recorded on two or more packs.

Format 4. (See Figure 31) The Format-4 label is used to define the VTOC itself.

This is always the first label in the VTOC. This label is also used to provide the location and number of available tracks in the alternate track area.

File Labels -- Tape

Fields 3-10 of IBM standard tape file labels are processed when the VOL and TPLAB job-control statements are present. These are the only fields processed. Only the first IBM standard tape file label per reel is processed; any others are bypassed. Tape files are checked and, if the existing label has expired, a new label is written from the information on the TPLAB statement. Tape input files are checked to determine whether the reel containing the correct file has been mounted. If errors occur, messages are printed, with the option to change the reel and check again, or to continue processing. The format of IBM standard tape file labels is shown in Figure 32.

In addition to IBM standard tape file labels, one to eight user standard labels may be placed on the tapes. You have to provide routines to process these labels. Exits available to you are given in Appendix D. The format of your standard tape file labels is shown in Figure 33.

Field	Bytes	Contents	Explanation
1	1-4	hexadecimal	'03030303' (identification)
2-17	5-44	See fields 21-24 of Format 1	Four groups of fields for four additional extents
18	45	3	Format identifier (Format 3)
19-50	46-125	See fields 21-24 of Format 1	Eight groups of fields for eight additional extents
51-54	126-135	Binary zeros	Not used

Figure 30. Model 20 Disk File Label, Format 3

Field	Bytes	Contents	Explanation
1	1-44	hexadecimal	'0404...0404' (identification)
2	45	4	Format identification (Format 4)
3	46-50	reserved	To be left blank
4	51-52	reserved	To be left blank
5	53-56	binary (cchh)	Address of next available alternate track cc = cylinder number hh = head or track number
6	57-58	binary	Number of alternate tracks still available
7	59	reserved	VTOC indicators (not used)
8	60	extent counter	Hexadecimal 2
9	61-62	reserved	To be left blank
10	63-64	binary (102 or 202 decimal equivalent)	Last cylinder initialized
	65-71	reserved	To be left blank
	72	bit 4 indicates whether initialized with ERASE option	Bit 4=0 ERASE not used Bit 4=1 ERASE was used
	73-105	reserved	To be left blank
11-14	106-115	See fields 21-24 of Format 1	Extent of VTOC
15	116-125	See fields 21-24 of Format 1	Extent of Label Information Area
16	116-135	reserved	To be left blank

Figure 31. Model 20 Disk File Label, Format 4

Field	Bytes	Contents	Explanation
1	1-3	HDR EOF EOV	Header End of file End of volume
2	4	1-8	File label number. Relative position of this file label in group of 1-8 tape file labels
3	5-21	alphameric	User-designated file identification
4	22-27	alphameric	File serial number. Identical to volume serial number in first or only volume
5	28-31	0001-9999	Volume sequence number. Used only with multi-volume files.
6	32-35	0001-9999	File sequence number. Used only with multi-file volumes.
7	36-39	0001-9999	Generation number. Unique identification of file edition
8	40-41	01-99	Version number of generation
9	42-47	b00001 to b99366	File creation date b is standard. 00 to 99 are the year, 001 to 366 are the day of the year
10	48-53	same as field 9	File expiration date
11*	54	0 or 1	File security (not used)
12*	55-60	numeric	Block count. Used in trailer labels only
13*	61-73	alphameric	System code
14*	74-80	reserved	To be left blank

*Not processed by System/360 Model 20 DPS or TPS programs.

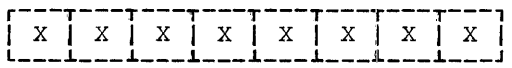
Figure 32. IBM Standard Tape File Label

Field	Bytes	Contents	Explanation
1	1-3	UHL UTL	User header label User trailer label
2	4	1-8	Tape file label number
3	5-80	alphameric	Contains any information you desire

Figure 33. User Standard Tape File Label

Appendix C. Data Formats

Data is stored in System/360 in bytes. Every byte consists of 8 bits, each having a value of zero or one. Coded combinations of bits within a byte can represent alphameric, numeric, binary, or logical data.



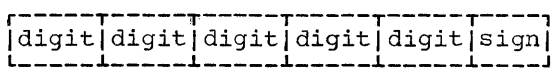
Bit Position: 0 1 2 3 4 5 6 7

one byte = 8 bits

A decimal number may be in either of two forms: packed decimal or zoned decimal. In both formats, all numbers are signed either positive or negative.

Packed Decimal

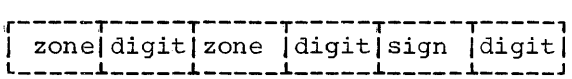
This format allows one 8-bit byte to contain two numeric digits, except in the rightmost byte of the field, which has a sign in the right half-byte.



byte byte byte

Zoned Decimal

This format contains one digit in the right most four positions of each byte.



byte byte byte

The left four bits of each byte are zone bits. The rightmost byte has a sign in the zone position.

Appendix D. Exits to User-Prepared Routines

If you want to process your own standard tape file labels , if Sterling-currency conversion routines are to be used, or if you wish to specify that tape files contain nonstandard labels, you must supply you own routine(s).

To inform the utility program that such a routine is present, a UPSI statement must be submitted with the job-control statements for the run. The UPSI statement must be as follows:

//bUPSIb1nnnnnnn

The digit corresponding to bit 0 must be a 1. The digits corresponding to bits 1-7 may be used as desired.

Two switches must be initialized (Figure 34) which the utility programs will use to determine the type of processing to be performed by the user coding. These switches must be set by assembling values into these locations with DC constants. The utility program loads your routine from SYSRDR (the designated card device), and does not transfer control to it immediately. Thus you have no opportunity to initialize these values before actual record or tape label processing begins.

LOC must be initialized at main storage location

SYSMON3+1 for Disk-to-Tape and Tape-to-Disk
SYSMON1+1 for Disk-to-Card, Card-to-Disk, and Disk-to-Printer.

SYSMON3 represents the end address of the generated Monitor including the Tape Error Recovery and Tape Error Statistics routines, while SYSMON1 represents the Monitor end address without the Tape Error Recovery and Tape Error Statistics routines. This end address is printed out after generation of the Monitor. After the Monitor has been loaded into main storage, the hexadecimal value of the Monitor end address may be obtained by displaying main storage locations X'E4-E5' (SYSMON3) or X'DE-DF' (SYSMON1), respectively.

If SYSMON3+1 results in an odd address, one must be added to provide an even address (e.g., SYSMON3+2). This also applies for SYSMON1+1. For checking purposes, the decimal address of LOC is printed out by the respective utility program when your routine is present.

The address for return from your routine is supplied in LOC+16-17 (i.e., in the two-byte field starting at LOC+16). The address of the I/O area for label processing or record processing is contained in LOC+14-15.

If you include a label processing routine , the one-byte switch in LOC+18 must also be initialized. Bits 0, 1, 2 and 3 of this byte are used during program execution to indicate the following.

- Bit 0 Exit to your input header-label routine
- Bit 1 Exit to your input trailer-label routine
- Bit 2 Exit to your output header-label routine
- Bit 3 Exit to your output trailer-label routine

Location	Setting (Hex.)	Meaning
LOC	00	User standard label routines are not to be processed. If user labels are present on input, they are to be bypassed. The file does not contain nonstandard labels (default value).
	0F	A user routine is present and user file labels are to be processed.
	F0	The file contains nonstandard labels which are to be bypassed.
LOC+1	00	User Sterling-currency conversion routines are not present (default value).
	FF	User Sterling-currency conversion routines are present and the utility program will branch to this routine for each record. This setting is valid only for Disk-to-Card, Card-to-Disk, and Disk-to-Printer Utility programs.

Figure 34. User Initialization Switches

When you do not wish to check or create further labels, you must set the appropriate bit to 0. The program label-routine bypasses additional user labels until the tapemark is sensed. If additional volumes are to be processed as part of the same job, the utility program will set the bit to 1 after the tapemark is sensed on each but the last volume.

PROGRAMMING RESTRICTIONS

Your own routines for a job are all assembled as a single program. The following restrictions must be observed.

- The program must set up the initialization switches and the routine reference points (Figure 35).
- The start location (origin) must correspond to the appropriate one given in Figure 36. These locations are the same regardless of the main-storage size of the machine.
- All general registers that are used must first be saved and finally restored to their initial values. The only exception to this requirement is register 14 which can be used freely.
- The I/O area specified for you is 80 bytes long. For label processing, no modification to main-storage contents should be made outside this area.
- The output area specified for Sterling-currency conversion is equal in size to the output record length specified in the Utility-Modifier statement. No modification to main-storage contents should be made outside this area.

Two-Byte Address	Use
LOC+2-3	Address of the last byte + 1 of your own routines
LOC+4-5	Address of your input-header routine
LOC+6-7	Address of your output-header routine
LOC+8-9	Address of your input-trailer routine
LOC+10-11	Address of your output-trailer routine
LOC+12-13	Address of your Sterling-currency conversion routine

Figure 35. User Routines Reference Points

EXAMPLE OF USER PROGRAM

A short sample user routine is shown in Figure 37. This routine examines the first two user standard header labels and looks for the labels "UHL1 LABEL" and "UHL2 LABEL". It also examines a single user standard trailer label and looks for the label "UTL1 LABEL". If the wrong label is encountered, the program halts. The program can be continued by pressing START on the CPU.

Utility	Name	Starting Location
Tape-to-Disk	TAPDSK	LOC+6020
Disk-to-Tape	DSKTAP	LOC+4750
Disk-to-Card	DSKCAR	LOC+4012
Card-to-Disk	CARDSK	LOC+5012
Disk-to-Printer	DSKPRT	LOC+4894

Figure 36. Starting Locations for User Routines

The routine is written with several symbolic names. Register 14 is assigned the name "R" and the user communications area is assigned the name LOC. The routine is assembled at the location specified in Figure 36.

The initialization DC statements set LOC to X'0F' and LOC+18 to X'C0'. The former setting indicates that user file labels are to be processed and the latter setting indicates that exits are required for a user input header-label routine and for a user input trailer-label routine. The remaining addresses are also defined in DC statements. DS statements are used to skip over halfwords indicating exits which are not to be taken. These values must not be changed so long as the exits are not to be used.

The input trailer-label checking routine obtains the location of the trailer record from bytes LOC+14-15 and compares the record with the expected label. If the label does not match, the computer halts with '0F21' displayed in the E-S-T-R registers. Before the comparison, bit 1 of LOC+18 is set to 0 so that the user exit will not be taken a second time.

The logic of the input header-label checking routine parallels that of the input trailer-label routine. An extra test is inserted to determine whether the label being checked is the first or second label. '0F11' and '0F12' will be displayed if the first user header label or the second user header label, respectively, is in error. When the routine ascertains that the second label is being checked, bit 0 of LOC+18 is set to 0. This ensures that the user routine will not be entered a third time.

```
SOURCE STATEMENT
                AOPTN NOESD, NORLD, CROSSREF
                ICTL  25
              *
              *TAPE-TO-DISK USER ROUTINE
              *
              MNTR    START 0
                      USING *,0
                      USING MNTR+4096,1
                      USING MNTR+8192,2
              *
              R       EQU  14
              EQ      EQU  8
              UNC     EQU  15
              *
                      ORG  MNTR+3600
              LOC     DC   X'0F00'
                      DC   Y(ENDU)          LOC+2
                      DC   Y(IHR)           LOC+4
                      DS   1H
                      DC   Y(ITR)           LOC+8
                      DS   4H
                      DC   X'C0'
              *
                      ORG  LOC+6020
              *
              *INPUT HEADER ROUTINE
              *
              IHR     LH   R,LOC+14         POINTER TO RECORD
                      CLI  P,C'1'           IS IT 1ST RECORD
                      BC   EQ,IHR1          YES
                      NI   LOC+18,X'7F'     NO, MASK RETURN OUT
                      CLC  0(10,R),HNAM     CHECK 2ND LABEL
                      BC   EQ,EXIT          IF LABEL OK
                      HPR  X'F12',0         ERROR 2ND LABEL
                      BC   UNC,EXIT
              IHR1    CLC  0(10,R),HNAM     CHECK 1ST LABEL
                      BC   EQ,IHR2          IF LABEL OK
                      HPR  X'F11',0         ERROR 1ST LABEL
              IHR2    LH   R,HNAM+2         MODIFY TEST LABEL
                      AH   R,B1
                      STH  R,HNAM+2
              *
              EXIT    LH   R,LOC+16         GET RETURN
                      BCR  UNC,R
              *
              *INPUT TRAILER ROUTINE
              *
              ITR     LH   R,LOC+14         POINTER TO RECORD
                      NI   LOC+18,X'8F'     MASK OUT RETURN
                      CLC  0(10,R),TNAM     CHECK TRAILER LABEL
                      BC   EQ,EXIT          IF LABEL OK
                      HPR  X'F21',0         ERROR TRAILER
                      BC   UNC,EXIT
              *
              *CONSTANTS AND STORAGE AREAS
              *
              B1      DC   H'1'
              HNAM    DC   C'UHL'
              P       DC   C'1 LABEL'
              TNAM    DC   C'UTL1 LABEL'
              ENDU    EQU  *
                      END
```

Figure 37. Sample User's Program

Appendix E. IBM 1316 Disk Pack Description

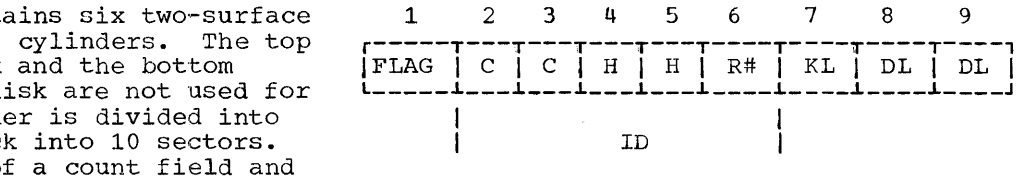
The 1316 Disk Pack contains six two-surface disks consisting of 203 cylinders. The top surface of the top disk and the bottom surface of the bottom disk are not used for recording. Each cylinder is divided into 10 tracks and each track into 10 sectors. Every sector consists of a count field and a record (data) field.

The schematic representation of a track is shown in Figure 38, in which

I = Index marker -- indicates the physical beginning of each track. There is one index marker per track.
G = Gap separating the different areas of the track.
C1-C10 = Count fields -- 9 bytes of data plus two cyclic check bytes. The data bytes include the flag byte, location identifier, and field-length information. There are 10 count fields per track.
R1-R10 = Record fields -- 270 bytes of user-controlled data plus two cyclic check bytes. There are 10 record fields per track.
S = Sector pulse -- defines the beginning of a sector on the track. There are 10 sector pulses per track including the index marker which also serves as a sector pulse.

The count field information is recorded in binary notation. This area contains the flag byte, which is 1 byte long; the location identifier (ID), which is 5 bytes long; the key length (KL), which is 1 byte long and is always equal to zero; the data length (DL), which is 2 bytes long and is always the binary equivalent of decimal 270; and the cyclic check, which is 2 bytes long. The cyclic check bytes are used for validity checking and are written automatically by the system.

The first 9 bytes of a count field on the disk have the following format.



Byte Location

1 Flag -- This byte indicates the condition of the track in the following manner.

00000000 - original good track
00000010 - defective track
00000001 - alternate track

2-3 CC = Cylinder number -- May contain any binary number equivalent to decimal 0-202; identifies the cylinder on which information is stored.
4-5 HH = Head number -- May contain any binary equivalent to decimal 0-9; identifies the location of the head (or the track) on the access mechanism.
6 R = Sector (record) number -- May contain any binary number equivalent to decimal 0-9.
7 KL = Key length -- Is always 0.
8-9 DL = Data length -- Is always the binary equivalent of decimal 270.

A sector field consists of 270 bytes available for your own data, followed by two cyclic check bytes. The cyclic check bytes are used for validity checking and are written automatically by the system. There is no key-length area recorded with the Model 20; therefore, the key length (KL) byte of the count field is always set to zero.

A block consists of one or more logical records. (Unblocked records are merely the special case of storing one record per block). The block stored on disk is equivalent to the block or physical record stored on tape (see Appendix F).

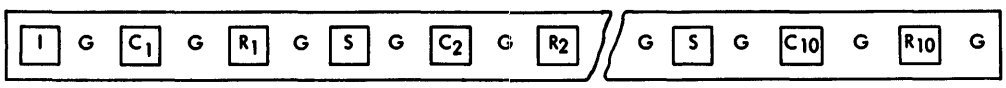


Figure 38. Schematic Representation of a Disk Track

When storing data files on disk, each block must begin in a new sector. Two blocks may not be contained in the same sector although one block may occupy several sectors. The number of sectors occupied by a block is the block length plus a number of bytes added to make the total number of bytes divisible by 270.

A block or a record may be contained in one sector or may overlap into several successive sectors. The 1-269 bytes at the end of a sector that may remain unused when a block is not a multiple of 270 bytes are filled with binary zeros. Ideally, therefore, a block should be designed to be a multiple of 270 bytes. In this way, no space in the data area is wasted, as the example below indicates. I/O time for reading or writing is also reduced to a minimum.

Block Length (bytes)	Sectors Occupied	Binary-0 Characters at End of Last Sector of Block
100	1	170
200	1	70
270	1	0
271	2	269
280	2	260
540	2	0
541	3	269
810	3	0

A block cannot extend from one cylinder to another. Normally you need not concern yourself with this limitation, since the program will automatically begin a block in the next cylinder within the extent when there is insufficient space to complete it in the former cylinder.

Appendix F. Record and Block Formats

Data Blocking

Only fixed-length records are handled by the DPS Disk Utility programs. If records are blocked, all blocks (except blocks of the input file in the Tape-to-Disk program) must contain the same number of records. Each data record or -- if records are blocked -- each block of records is called a physical record. (On tape, a physical record consists of the area between two successive inter-block gaps).

When reading from magnetic tape or disk, or writing on it, the unit of transfer is always a physical record.

A physical record on tape has a minimum length of 18 bytes and a maximum length of 4095 bytes. A physical record on disk has a minimum size of one sector (270 bytes), although the data record or block may require only a single byte.

The program manipulates records internally in the central processing unit (CPU) in units of logical records. A logical record consists of a data record, rather than a block of records. If records are unblocked, the physical and logical records are identical; if they are blocked, each physical record contains two or more logical records, although a single logical record may occur in a block.

The efficiency of most program runs is highest when records are stored in blocked format.

Figure 39 illustrates fixed-length blocked and unblocked tape records. The terms "data record", "logical record", and "physical record" have similar significance for disks, but inter-block gaps do not exist as such. (Refer also to Appendix E).

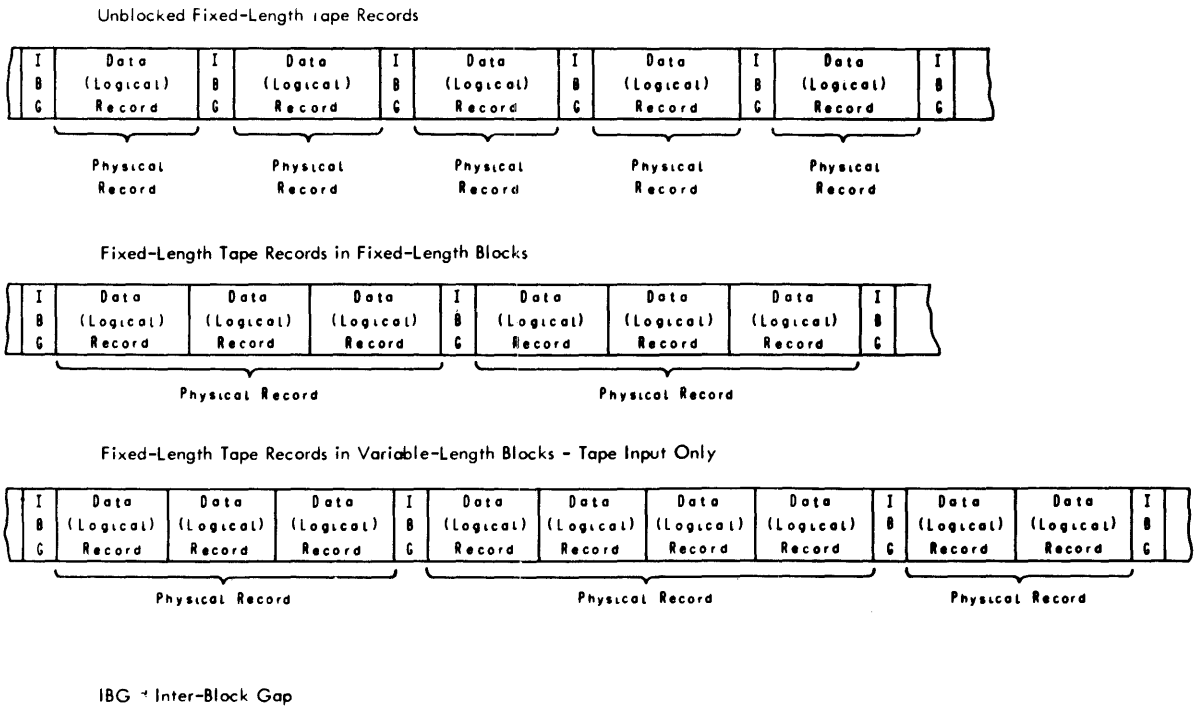


Figure 39. Tape Record Formats

Appendix G. Control Statements for Utility Program

Program / Statement	Disk to Disk	Disk to Tape	Tape to Disk	Disk to Card	Card to Disk	Disk to Printer	Initialize Disk	Clear Disk	Alternate-Track Assignment	Disk Dump
JOB	DKSDSK	DSKTAP	TAPDSK	DSKCAR	CARDSK	DSKPRT	INTDSK	CLRDSK	ATASGN	DDUMP
VOL (File Name)	UIN UOUT	UIN UOUT	UIN UOUT	UIN	UOUT	UIN	Not Used	UOUT	Not Used	Not Used
ASSGN Primary Input	SYSIPT	SYSIPT	SYSIPT	SYSIPT	SYSIPT	SYSIPT	SYSIPT	Not Used	Not Used	Not Used
ASSGN Primary Output	SYSOPT	SYSOPT	SYSOPT	SYSOPT	SYSOPT	SYSLST	SYSLST SYSOPT	SYSOPT	SYSOPT	SYSOPT
ASSGN Alternate Tape Input	Not Used	Not Used	SYS000	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
ASSGN Alternate Tape Output	Not Used	SYS001	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
ASSGN Alternate Disk Input	SYS002 SYS003 SYS004	SYS002 SYS003 SYS004	SYS002 Not Used	SYS002 SYS003 SYS004	SYS002 Not Used	SYS002 SYS003 SYS004	SYS002 SYS003 SYS004	SYS002 Not Used	Not Used	Not Used
ASSGN Logging	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG	SYSLOG
ASSGN Alternate Disk Output	SYS005 SYS006 SYS007	Not Used	SYS002 SYS003 SYS004	Not Used	SYS002 SYS003 SYS004	Not Used	SYS002 SYS003 SYS004	SYS002 SYS003 SYS004	Not Used	Not Used

Figure 40. Summary of Job-Control Statements (Fields Unique to Utility Programs)

Program / Operand	Disk to Disk	Disk to Tape	Tape to Disk	Disk to Card	Card to Disk	Disk to Printer	Initialize Disk	Clear Disk	Alternate-Track Assignment
Utility-Modifier Statement ID	//bU	//bU	//bU	//bU	//bU	//bU	//bU	//bU	//bU
Program ID xx	DD	DT	TD	DC	CD	DP	IN	CL	AT
Function Specification Tt	TC TF TR TRF	TC TF TR TRF	TC TF TR TRF	TC TF TR TRF	TC TF TR TRF	TD TL TLF	TPI TSI		
Record Type Ff	FF	FF	FF	FF	FF	FF			
Input Record and Block Size A=(input)	A = (n,m)	A = (n,m)	A = (n,m) (m:max block size)	A = (n,m)	A = (n,n)	A = (n,m)			
Output Record and Block Size B=(output)	B = (a,c)	B = (a,c)	B = (a,c)	B = (a,a)	B = (a,c)	B = (P)			
Input Tape Handling or Data Format Ix			IN IR IU IM		I1 I2				
Output Tape Handling, Data Format, Write-Disk Check, or Print Data Ox	OY ON	ON OR OU	OY ON		OY ON	OX OC		OY ON	OY ON

Figure 41. Utility-Modifier Statement Summary

Program / Operand	Disk to Disk	Disk to Tape	Tape to Disk	Disk to Card	Card to Disk	Disk to Printer	Initialize Disk	Clear Disk	Alternate-Track Assignment
Spacing Control Sx						S1 S2			
Page Numbering Px						PY PN			
First Record Printed Rx						Rx			
Sequence Checking or Numbering Q=(x,y)				Q=(x,y)	Q=(x,y)				
Clear Disk Fill								C'a' X'aa'	
Cylinder and Head Location R=(ccccchhh)									R=(cccc hhh)
Listing VTOC							TL		

Figure 42. Utility-Modifier Statement Summary

```
DECIMAL ADDRESS OF LOC ..................... xxxx

RUN TYPE ................................... COPY
                                             REBLOCK
                                             DATA DISPLAY
                                             LIST
                                             FIELD-SELECT
                                             REBLOCK AND FIELD-SELECT
                                             LIST AND FIELD-SELECT

RECORD FORMAT .............................. FIXED LENGTH
                                             VARIABLE LENGTH

INPUT RECORD LENGTH ........................ xxxx

INPUT BLOCK LENGTH ......................... xxxx

PRINT WIDTH ................................ xxx

OUTPUT RECORD LENGTH ....................... xxxx

OUTPUT BLOCK LENGTH ........................ xxxx

INPUT OPTION ............................... EBCDIC CARDS
                                             BINARY CARDS
                                             DO NOT REWIND TAPE
                                             REWIND TAPE
                                             REWIND AND UNLOAD TAPE
                                             MULTIPLE TAPES

OUTPUT OPTION .............................. HEXADECIMAL
                                             ALPHANUMERIC
                                             DO NOT REWIND TAPE
                                             REWIND TAPE
                                             REWIND AND UNLOAD TAPE
                                             WRITE-DISK CHECK
                                             DO NOT WRITE-DISK CHECK

SPACING OPTION ............................. SINGLE SPACE
                                             DOUBLE SPACE
                                             TRIPLE SPACE

PAGE NUMBERS ............................... YES
                                             NO

STARTING SEQUENCE NUMBER ................... xxxx

STARTING SEQUENCE COLUMN ................... xx

SEQUENCE LENGTH ............................ xx

xxxxxxxINPUT BLOCKS PROCESSED (Disk and tape input only)

yyyyyyyINPUT BLOCKS BYPASSED (Disk and tape input only)

zzzzzzzOUTPUT BLOCKS WRITTEN (Disk and tape output only)
```

Figure 43. Logging Messages for File-to-File Utility Programs

```
ALTERNATE   CYL   XXX   TRK    X    DEFECTIVE

ALTERNATE   CYL   XXX   TRK    X    TEMPORARY DEFECT

CYL   XXX   TRK   X     DEFECTIVE ALTERNATE

CYL   XXX   TRK   X     DEFECTIVE, ASSIGNED ALTERNATE   CYL   XXX   TRK   X

CYL   XXX   TRK   X     SCT    X    CONTAINS

CYL   XXX   TRK   X     TEMPORARY DEFECT

Disk pack has been initialized
```

Figure 44. Logging Messages

Message	Reason
DATA LOSS IN FS xxxx	The output field length of Field-Select entry # xxxx is too short for the field specified as input. Printed only for pack or unpack field-selection.
FS xxxx OVERLAYS FS yyyy	The output field of Field-Select entry # xxxx will overlap with the output field of the previous entry, yyyy.
PROGRAM ID FOR ANOTHER UTILITY	The Utility-Modifier statement contains an ID for another program. For example, //bUDD was processed for Disk-to-Printer program.
SEQUENCE ERROR xxxxxxxxxx yyyyyyyyyy	The sequence number field "y" just read is less than or equal to the field "x" read previously. The next number will be compared to "y".
SEQUENCE OVERFLOW	The output sequence number just exceeded the maximum decimal number allowed in the specified sequencing field length. Sequence numbers are restarted from zero.
SEQUENCING OVERLAPS FS xxxx	The output sequence number field to be generated will overlap with the output field of Field-Select entry # xxxx.

Figure 45. Warning Messages for all Utility Programs

Message	Reason
CYLNDR = FORMAT NOT (N)	
CYLNDR OPERAND ERROR	Operand not 102 or 202.
DATA ERROR - PRESS START TO BEGIN SURFACE ANALYSIS	
DISK VOLUME LABEL MISSING	Program unable to read disk volume label.
DISK VTOC MISSING	
DUPLICATE OPERAND	
END STATEMENT INCORRECT OR MISSING	The last-card condition was detected on SYSRDR, or a card with a 12-2-9 punch in column 1 was read.
EXTENT = FORMAT NOT (N)	
EXTENT OPERAND ERROR	Operand not between 1 and 10.
FIELD-SELECT MUST BE SPECIFIED	The input and output record lengths are unequal and a TC or TR option was specified.
IF 'DATA LOSS' OR 'OVERLAP' WARNINGS CAN BE IGNORED, PRESS START KEY TO CONTINUE	This is to allow the operator to decide whether to continue or terminate the job.
x ILLEGAL FORMAT	Format specifications were not followed: C = Operands extend beyond end of card; M = Multiple optional operand definitions; N = Illegal numeric field (value lies outside the range 1-27000); O = Card order or usage is incorrect; H2 but no H1 read; FS read and not required, or required but not read. U = Undefined operand or format error.
INPUT BLOCK xxxxx DE	Input block xxxxx on the current tape reel (volume) was read with an irrecoverable data error.
INPUT BLOCK xxxxx WLR	Input block xxxxx on the current tape reel (volume) either exceeds the specified input block size, or is not a multiple of the input record length. There is no data error.
INPUT BLOCK xxxxx DE+WLR	A combination of the two conditions above.
INPUT TAPE LABEL REQUIRED	No input tape label for UIN found, and a. User routine specifying label processing is present; or b. Multiple input tapes specified.
INSUFFICIENT EXTENT	Not enough tracks specified on XTENT control statement for the output file to contain the entire input file. (It may be possible that all the data was transferred but only the /* record could not be written).
INVALID ALTERNATE ASSIGNMENT	a. SYS000 was assigned the same physical device as SYSIPT. b. SYS001 was assigned the same physical device as SYSOPT.

Figure 46. Error Messages, Part 1 of 4

Message	Reason
INVALID CONTROL CARD	A //b card was read which was not UM, FS or END; or not H1 or H2 for Disk-to-Printer program.
INVALID DELIMITER	Program expects comma or blank or UM or VTOC statement.
INVALID FILL CHARACTER	Fill character missing or invalid format.
INVALID INPUT BLOCK OR RECORD LENGTH	a. The input block length exceeds specified limits: for tape input, 18 to 4095 bytes; for card input, 1 to 80 (EBCDIC) or 1 to 160 (binary); for disk, 1 to 27,000 bytes. b. The input block length is not a multiple of the input record length. c. The input record length is less than 18. (Tape input only)
INVALID INPUT RECORD LENGTH	a. Card input - the input record length does not equal the input block length. b. Tape input - the input record length exceeds the input block length. c. Disk input - the input record length exceeds the input block length.
INVALID OPERAND	Invalid keyword in utility-modifier or VTOC statement.
INVALID OUTPUT BLOCK OR RECORD LENGTH	a. The output block length exceeds specified limits: for tape output, 18 to 4095 bytes; for card output, 1 to 80 columns; for printer output, 1 to 144 (list) or 20 to 144 (display) positions; for disk, 3 to 27,000 bytes. b. The output block length is not a multiple of the output record length. c. The output record length is less than 18. (Tape output only)
INVALID OUTPUT RECORD LENGTH	a. Card output - the output record length does not equal the output block length. b. Tape output - the output record length exceeds the output block length. c. Disk output - the output record length exceeds the output block length.
INVALID SECONDARY INITIALIZATION	Not a model 11 Disk Storage Drive.
INVALID SEQUENCING REQUEST	The sequencing field length is greater than 10, or the rightmost sequencing column is beyond column 80.
INVALID SYSIPT ASSIGNMENT	a. SYSIPT was not assigned. b. An incorrect type of input device was assigned c. An illegal card punch was assigned for a utility program requiring card input.
INVALID SYSLST ASSIGNMENT	a. SYSLST was not assigned. b. The print width specified exceeds that allowed on the printer assigned to SYSLST.

Figure 46. Error Messages, Part 2 of 4

Message	Reason
INVALID SYSOPT ASSIGNMENT	a. SYSOPT was not assigned. b. An incorrect type of output device was assigned. c. An illegal card punch was assigned for a utility program requiring card output.
INVALID TT OPERAND	a. TT operand not TPI, TSI or TL; b. TL operand on second initialization.
I/O AREA CANNOT BE ASSIGNED	Insufficient storage space is available to assign the necessary I/O areas.
MULTI-VOLUME TAPE PROHIBITED	Insufficient space is available to perform label processing on more than one volume.
NO MORE ALTERNATE TRACKS	Defective track found and no alternate track available.
NO OPERAND ON VTOC STATEMENT	
NOT UM STATEMENT OR END STATEMENT	Invalid utility control statement.
OUTPUT TAPE LABEL REQUIRED	No output tape label for UOUT found and user routines are present specifying user output label processing to be done.
PROGRAM AND DATA OVERLAP	The number of field-selects (and header statements for Disk-to-Printer program) specified exceeds the available storage capacity; or the field-select code necessary to perform the requested field selection exceeds the available storage capacity.
R = FORMAT NOT (CCCCHHH)	Invalid format.
R = OPERAND MISSING	No defective track specified; or incorrect format.
R = OPERAND ERROR	a) Assign an alternate track; b) Not within disk-pack size.
/* READ	End-of-file record was encountered on disk input file.
REBLOCKING MUST BE SPECIFIED	The input and output block lengths are unequal and a TC or TF option was specified.
RECORD CAPACITY EXCEEDED BY FS xxxx	The input or output fields of Field-Select xxxx are in part or altogether beyond the specified input or output record length.
STRTADR = FORMAT NOT (CCCCHHH)	
STRTADR OPERAND ERROR	a. Alternate track specified; b. Higher than maximum cylinder; c. Track number greater than 9; d. Not in area of initialization.

Figure 46. Error Messages, Part 3 of 4

Message	Reason
SYSIPT SYSOPT NOT A 2311 SYS002 DISK DRIVE	Incorrect specifications in job-control statements.
SYSIPT SYSOPT SYS002 NOT ASSIGNED SYSLST	Incorrect specifications in job-control statements.
UM STATEMENT MISSING	Program expects utility-modifier statement.
UNEXPIRED FILE XXXXXX	An unexpired file (xxxxxx) has been found.
UNSUCCESSFUL DATA TRANSFER	Data is read from the defective track and written on the alternate track, but cannot be verified.
UNSUCCESSFUL INITIALIZATION	Incorrect count fields have been written on the disk pack because of a malfunction of the disk drive during initialization.
UNSUCCESSFUL READ COUNT	Unable to read the count field in all 10 sectors of a track.
USER ROUTINE OVERLAYS MONITOR	LOC address too low.
USER ROUTINE OVERLAYS UTILITY	Origin address too low.
USER ROUTINE TOO LARGE	User's last address exceeds available core storage
VERIFY = FORMAT NOT (N)	Invalid VERIFY operand format.
VERIFY OPERAND ERROR	Operand not between 1 and 256.
VOL1 OR END STATEMENT MISSING	
WRITE CHECK CYL XXX TRK X	CYL xxx TRK x data cannot be properly written on track.
/* WRITTEN IN BLOCK STARTING AT CYLINDER XXX TRACK X SECTOR X ON DISK PACK XXXXXX	End-of-file reached on input file and disk output file terminated by padding last block with EOF records. Part or all of the block contains EOF records.

Figure 46. Error Message, Part 4 of 4

Appendix I. Use of the UPSI Statement to Override the End-of-File Record

The Disk-to-Disk utility program is used to transfer the contents of one disk pack to another. Since a pack will normally contain several files, each concluded by the end-of-file indicator, /*, the program must be instructed to treat this two-character sequence as a piece of data; otherwise processing will terminate after the first file is read. An UPSI statement with the following format is included with the job-control statements for the run:

```
//bUPSIb01nnnnnn
```

The digit corresponding to bit 1 must be 1. The digits corresponding to bits 0 and 2-7 may be used as desired.

This feature of the UPSI statement is also available in the Disk-to-Tape, Disk-to-Card, and Disk-to-Printer utility programs.

To copy all the files on a disk pack, the XTENT statement should specify the entire disk, excluding the volume label area and the alternate track area.

When copying the contents of an entire disk pack you must be aware of the fact that, with the VTOC, the number of alternate tracks, the address of the next available alternate track, and the extent of the Label Information Area (LIA) are copied. This means that, prior to copying, the new disk pack to be prepared (either directly or via magnetic tape or cards) must contain the same number of alternate tracks, the same address of the next available alternate track, and the same LIA extent as the source pack. Further, since the volume label which contains the address of the VTOC, is not copied, the location of the VTOC must be identical on both packs.

Appendix J. Processing Multi-Volume Files on Disk

To process multi-volume files on disk, at least two disk packs with different volume serial numbers are needed. The maximum number of disk packs is not limited. Since the file-to-file utility programs process only sequential disk files, the disk packs may be processed on-line (disk packs are mounted simultaneously on different disk drives and processed one after the other) or off-line (after a disk pack has been processed a halt occurs, the old pack is removed, and the new one is mounted on the same drive), or both on-line and off-line.

As a rule, off-line processing is not possible with the disk drive to which the symbolic device address SYSRES has been assigned, since the utility program needs the label information stored on SYSRES via the job-control program.

If you want to process multi-volume files on one disk drive, the label information on each disk pack has to be identical. The job can be run by means of the card-resident system; if you use the disk-resident system, at least the first and the last disk pack must be a system disk pack.

Whenever you want to process multi-volume files, you only have to provide the label information via XTENT statements supplied with the job-control statements (See Appendix A).

On-line processing

Figure 47 shows several examples of how to prepare the job-control statements for multi-volume processing when using four disk drives on-line. The first job (CLRDSK) clears the tracks of four disk packs with the volume serial numbers PAYRL1 through PAYRL4 (as specified in the XTENT statements). Only the volume serial number of the first volume to be processed has to be stated in the DLAB statement, since the volume sequence number will automatically be increased by the utility program when the disk packs are changed.

With the second job (CARDSK) first all specified extents on SYSOPT (disk drive 1) are filled with data, then those on SYS002 (disk drive 2), and so on until all data is stored on disk or the specified extents are completely filled.

The third job (DSKDSK) transfers data on four disk packs to different extents on the same packs. Here again data is processed in the sequence of the specified XTENT statements.

The symbolic device addresses SYS002-SYS004 need only be assigned if the disk drives are actually being used. However, it is not possible to have an extent on SYS004 without having assigned SYS002 and SYS003 to disk drives. The same applies to SYS005-SYS007 if the DSKDSK program is used. You can process the disk packs in any order, i.e., you may specify an extent on the disk pack mounted on SYS003 and afterwards another on SYSOPT or SYSIPT, respectively.

Off-line processing

If it is necessary to process a multi-volume volume, sequentially organized file consisting of more disk packs than disk drives are attached to the system, the disk packs must be changed during processing. Figure 48 gives an example of how to fill in the job-control statements for this problem. Here only one disk drive is necessary; each time the program reaches the end of all extents of the appropriate disk pack it tries to open the extents on the next disk pack of the same drive. Since the volume serial number does not agree, the program comes to a halt indicating that the volume serial number does not match with the volume serial number specified in the appropriate XTENT statement. Now the next volume can be mounted on the disk drive and the job will be continued.

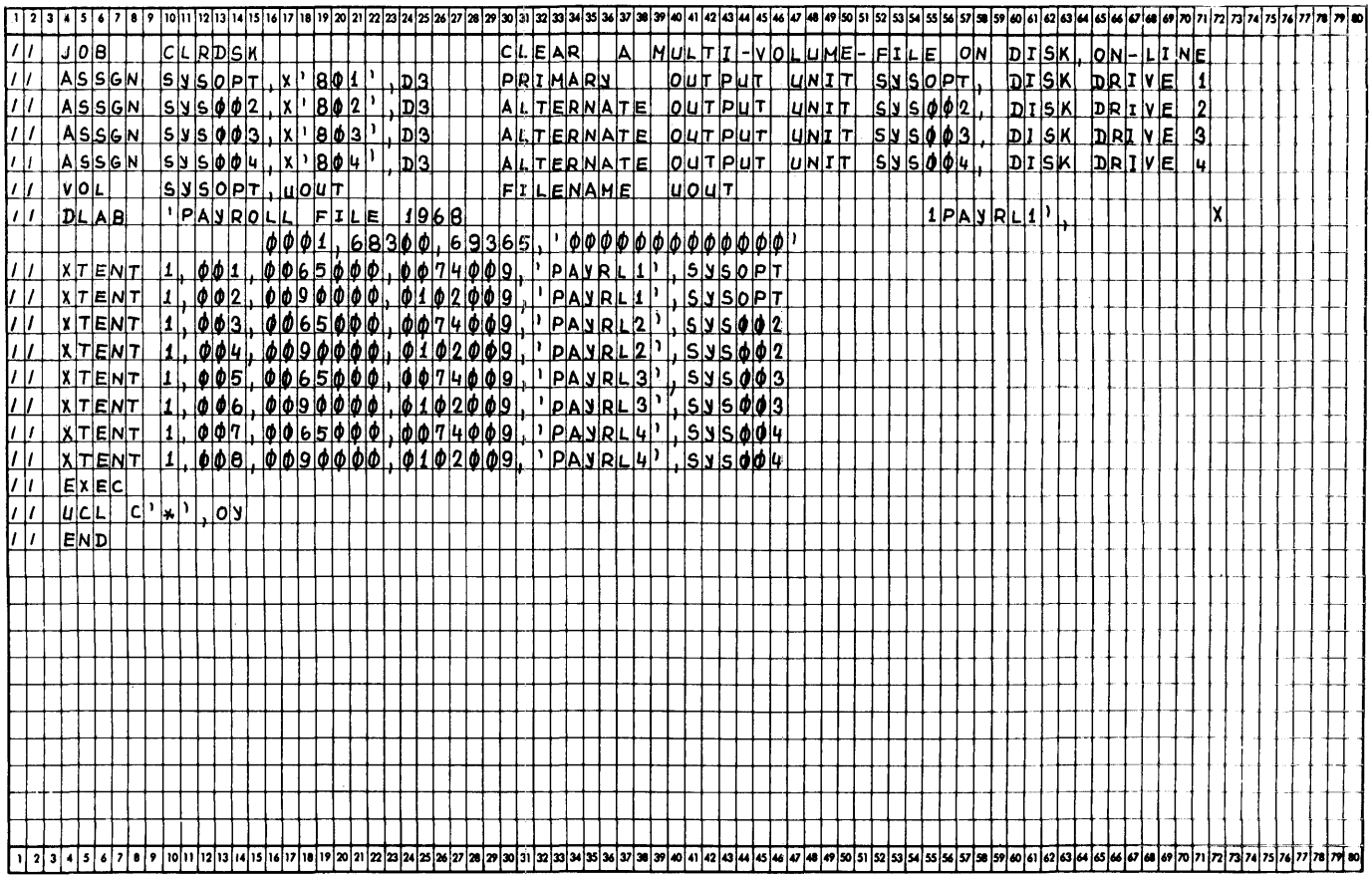


Figure 47. On-line CLRDSK Processing (Part 1 of 3)

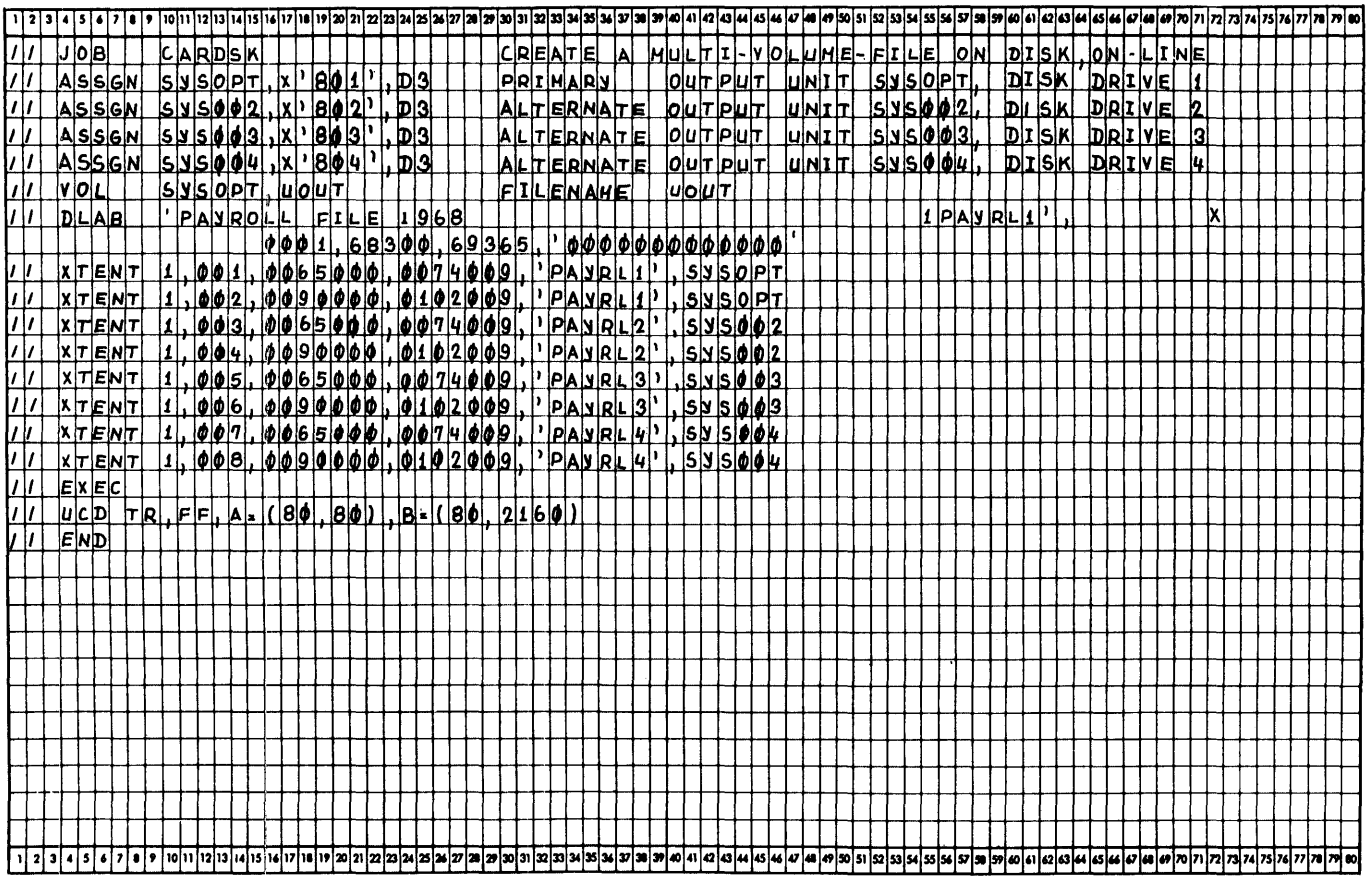


Figure 47. On-line CARDSK Processing (Part 2 of 3)

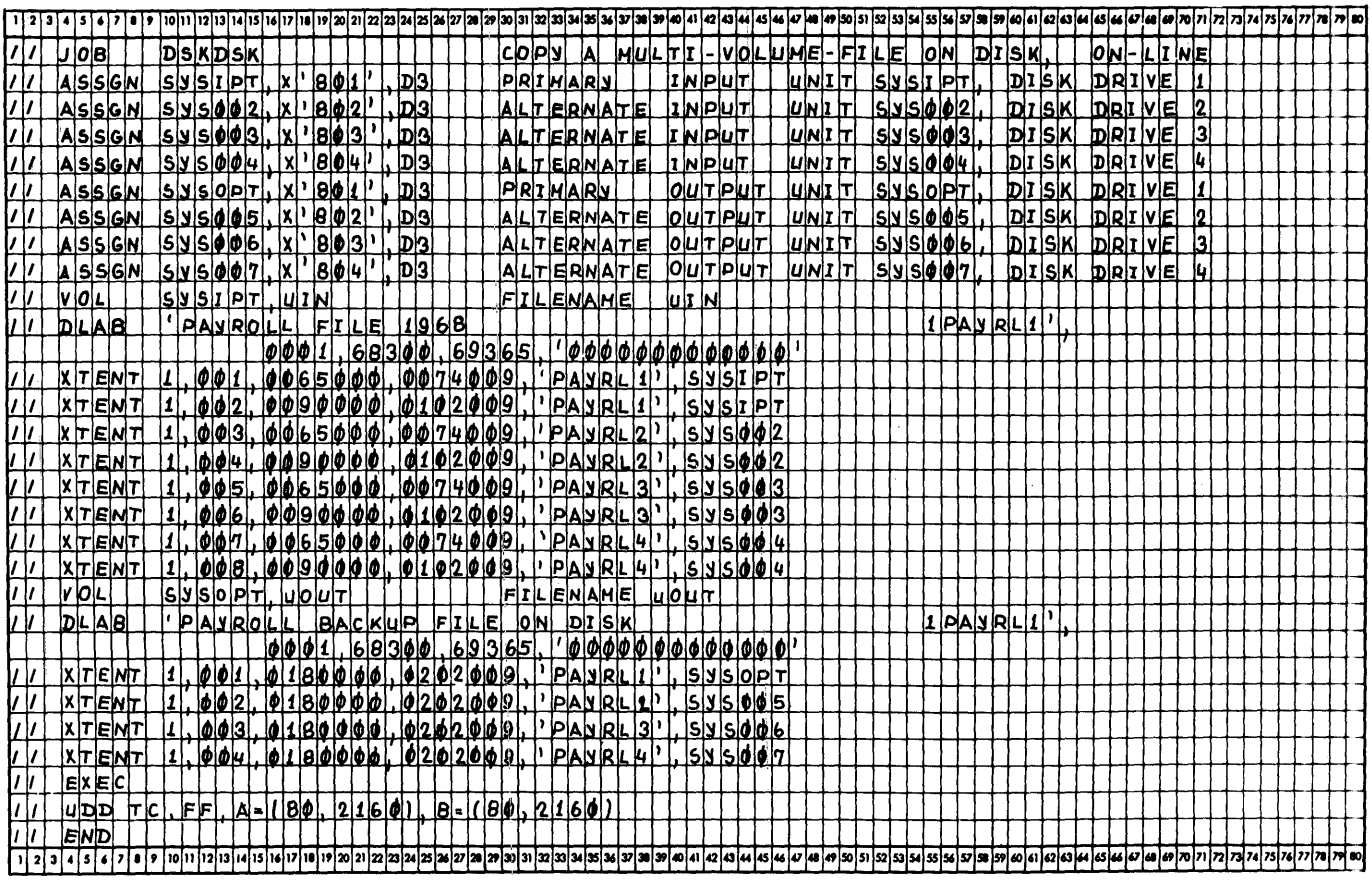


Figure 47. On-line DSKDSK Processing (Part 3 of 3)

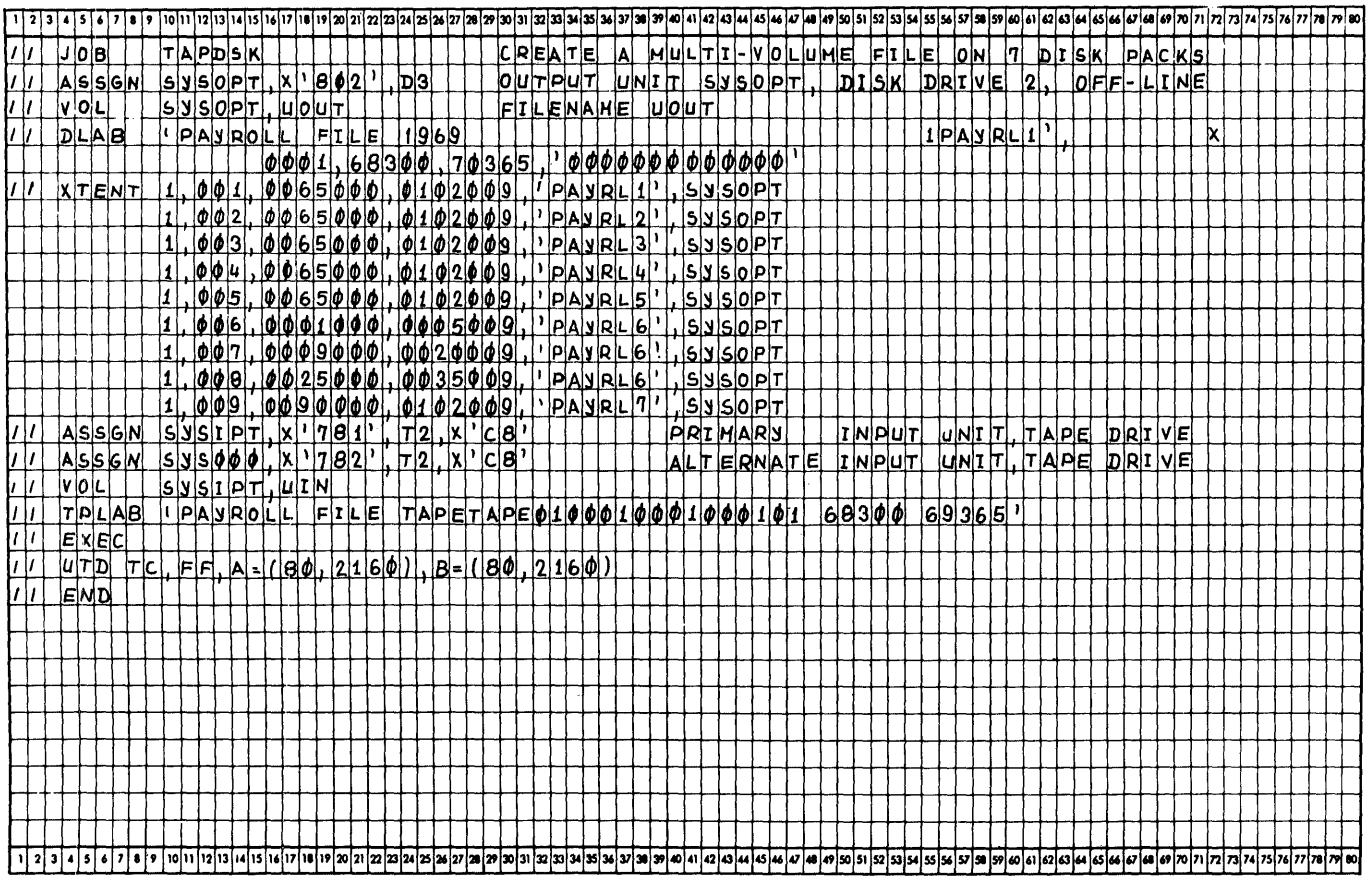


Figure 48. Off-line Processing

Alternate Track Area. An area of three cylinders on the disk pack in which tracks may be used as alternatives to defective tracks, occurring elsewhere on the disk pack.

b. The symbol for a blank space.

Bit. The smallest unit of information in System/360. It can have either of the two binary values: zero or one.

Block.

1. To group records physically for the purpose of conserving storage space or increasing the efficiency of access or processing.

2. A physical record on tape or disk.

Byte. The smallest addressable unit of information in System/360. Every byte consists of 8 bits, each having a value of zero or one. (See Bit).

Column Binary. A method of storage representation on punched cards. Every punch position represents a bit (=1 if hole, =0 if no hole). Sets of bits are read by card column, two 6-bit sets per column. The first set is the 12-11-0-1-2-3 positions and the second set the 4-5-6-7-8-9 positions.

Conversion. The act of packing or unpacking.

Data Display. Printed byte-for-byte representation of a logical record or block. Printed information will extend over as many lines as necessary. File description is also printed.

Data File. A collection of related data records organized in a specific manner.

Data List. Printed representation of a logical record that can be edited by the user. No more than one line of printed output can be used per record. No file description is printed.

Data Record. See Logical Record.

Default Value. The operand value assumed by a program when the value is omitted from a control statement.

EBCDIC. (Extended Binary Coded Decimal Interchange Code). A specific set of 8-bit codes standard throughout System/360.

Field. A contiguous set of digits or characters that form a meaningful entity.

File. See Data File.

Fixed-Length Record. A record having the same length as all other records with which it is logically or physically associated.

Format. The general makeup of data or of a control statement, record, or a file.

Graphic. The visual representation of a character or symbol.

Half-Byte. The leftmost or rightmost four bits of a byte. Can contain representation of a digit or the sign of a number.

Hexadecimal. A character representation for a set of four bits. The values 0-15 are represented by the digits 0-9 and the alphabetic characters A-F.

Input. Data to be processed by a computer program. See also Output.

Inter-Block Gap. A blank space on magnetic tape that separates physical records.

I/O Area. An area (portion) of main storage into which data is read or from which data is written. I/O means Input/Output.

Job Control Program. A program that is called into main storage to prepare each job to be run. Some of its functions are to assign I/O devices to certain symbolic names, set switches for program use, log (or print) job-control statements, and identify the requested program phase.

Job-Control Statement. Any one of the control statements in the input stream that identifies a job or defines its requirements and options.

Label. A physical identification record on magnetic tape or disk.

Label Information Area (LIA). An area on the system disk pack into which disk file label information, as contained in the VOL, DLAB, and XTENT statements, is placed by the Job Control program.

Logical File. Usually used to describe a file (see Data File) that shares a reel of tape or a disk pack with other files.

Logical Record. A record identified from the standpoint of its content, function, and use rather than its physical attributes. It is meaningful with respect to the information it contains. (Contrasted with Physical Record).

Mnemonic. A contraction or abbreviation whose characters are suggestive of the full expression.

Operand. The representation of a value that must be supplied to define a selective function to the program.

Output. The resultant data produced by a computer program. See also Input.

Pack. A storage technique whereby two digits or one digit and sign are stored per byte.

Packed Decimal. See Pack.

Physical Record. A record identified from the standpoint of the manner or form in which it is stored and retrieved; that is, one that is meaningful with respect to access. (Contrasted with Logical Record).

Reblock. To change the format of a file so that a different number of logical records comprises one physical record. See Block.

Record. A general term for any unit of data that is distinct from all others when considered in a particular context.

System Disk Pack. The disk pack on which your disk-resident system is stored.

Tapemark. A special symbol that can be read from or written on magnetic tape. Used to recognize the end of a file or file segment, and to segregate the labels from data.

Unblock. To change the format of a file so that a physical record comprises only one logical record. See Block.

Unpack. Storage technique whereby one digit, with or without sign, is stored per byte.

User Routine. A routine supplied by the user and incorporated into a utility program as a modification.

Utility Control Statement. A record that contains information to tailor a utility program to the requirements of the problem program.

Volume. That portion of a single unit of storage medium that is accessible to a single read/write mechanism. For example, a reel of magnetic tape for an IBM 2415 magnetic tape drive.

Volume Table of Contents (VTOC). A number of records on a disk pack, composed of disk file labels, specifying extents of, and identifying all files on the pack.

Zoned Decimal. See Unpack.

Index

READER'S COMMENT FORM

IBM System/360 Model 20
Disk Programming System
Disk Utility Programs

Form C26-3810-3

- How did you use this publication?

  As a reference source ..... ☐
  As a classroom text ..... ☐
  As a self-study text ..... ☐

- Based on your own experience, rate this publication . . .

	.....	.....	.....	.....	.....
As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE . . .

This SRL manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation

112 East Post Road

White Plains, N. Y. 10601

Attention: Department 813 BP

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

CUT ALONG THIS LINE

IBM System/360 Printed in U.S.A. C26-3810-3

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM System/360 Printed in U.S.A. C-3810-3