UNIVAC
OS/4
Data Management System

Programmer Reference

SPERRY UNIVAC COMPUTER SYSTEMS

UP-7629 Rev. 2

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

Other trademarks of the Sperry Rand Corporation include:

FASTRAND

UNISCOPE

UNISERVO

PAGEWRITER

MATED-FILM

PRINTED IN U.S.A.

PUBLICATIONS UPDATE

UNIVAC OS/4

Data Management System

Programmer Reference

UP-7629 Rev. 2–A.

This UNIVAC OS/4 Library Memo announces the release and availability of Updating Package A to "UNIVAC OS/4 Data Management System Programmer Reference," UP-7629 Rev. 2.

Updating Package A should be utilized as specified on the Updating Summary Sheet. Upon completion of manual updating, file the Updating Summary Sheet after the Contents Section so that a record of updating is maintained.

Section 2 contains new information on processing 7-track tape with the data conversion feature.

Information on the UNIVAC 8424 Disc Subsystem has been added to Sections 3 and 4. In Section 3, new information on the DTFSD and the DTFCD macro instructions has been added and a change has been made to the TRUNC macro instruction.

Section 4 contains a change to the example for the LBRET macro instruction. A correction has been made to the list of DTFMT I/O modules in Appendix C and additions have been made to the table of error codes in Appendix E. Appendix J contains new information on the object module of equates and two new sample control streams.

Note that the title of the manual has been changed to reflect the software operating system instead of the hardware system. Note also that throughout the manual, a large number of references to the UNIVAC 9400 System have been changed to reflect the UNIVAC OS/4 Operating System. Because of the volume of these references, arrows in the margin of the text have been omitted for these changes. All other technical changes in the manual are denoted by arrows.

Copies of Updating Package A are now available for requisitioning. Either the updating package alone, or the complete manual with the updating package may be requisitioned by your local Sperry Univac Representative. To receive the updating package alone, order UP-7629 Rev. 2–A. To receive the complete manual, order UP-7629 Rev. 2.

LIBRARY MAILING LISTS | PAPER PRINTOUT MAILING LISTS | PUBLICATIONS

Mailing Lists 217, 630 and 692

Mailing Lists 60, 61, 65 and 66
(Package A to UP-7629 Rev. 2, cover and 141 pages plus Memo and 2 Updating Summary Sheets.)

Library Memo for UP-7629 Rev. 2–A.

RELEASE DATE:

December 1973

UPDATING PACKAGE A
File pages as specified below

SECTION	DESTROY FORMER PAGES NUMBERED	FILE NEW PAGES NUMBERED
Front Cover & Disclaimer	†	†
Page Status Summary	PSS-1	PSS-1-A
Contents	1 and 2	1A and 2
	5 and 6	5 and 6A
Section 1	1 and 2	1A and 2
Section 2	1 and 2	1A and 2A
	3 and 4	3 and 4A
	9 and 10	9 and 10A
	N.A.	12aA**
	13 and 14	13A and 14A
	15 and 16	15A and 16
	17 and 18	17A and 18
	19 and 20	19A and 20
	21 thru 24	21A thru 24A
	25 and 26	25A and 26
	27 and 28	27A and 28
	29 and 30	29A and 30
	31 and 32	31 and 32A
	35 and 36	35 and 36A
Section 3	17 and 18	17A and 18A
	23 and 24	23A and 24
	25 and 26	25A and 26A
	29 and 30	29A and 30
	35 and 36	35A and 36
	39 and 40	39 and 40A
	77 and 78	77A and 78
	85 and 86	85 and 86A
Section 4	1 and 2	1 and 2A
	3 and 4	3A and 4
	5 and 6	5A and 6
	15 and 16	15A and 16A
	19 and 20	19A and 20
	27 and 28	27 and 28A
	29 and 30	29A and 30
	31 and 32	31A and 32
	33 and 34	33A and 34
	35 and 36	35 and 36A

UNIVAC OS/4
Data Management System
Programmer Reference
UP-7629 Rev. 2

December 1973

UPDATING PACKAGE A
File pages as specified below

SECTION	DESTROY FORMER PAGES NUMBERED	FILE NEW PAGES NUMBERED
Section 4	39 and 40	39A and 40
	43 and 44	43A and 44
	45 and 46	45A and 46
	49 and 50	49 and 50A
Section 5	1 and 2	1A and 2A
	3 and 4	3 and 4A
	5 and 6	5A and 6
	7 and 8	7A and 8
Appendix A	5 and 6	5 and 6A
Appendix B	1 thru 6	1A thru 6A
Appendix C	1 and 2	1A and 2A
	3 and 4	3 and 4A
Appendix E	1 thru 3	1A thru 3A
Appendix G	1 thru 4	1A thru 4A
	5 and 6	5A and 6
	7 and 8	7A and 8
	9 and 10	9A and 10
Appendix H	1 and 2	1A and 2A
	3 and 4	3A and 4
Appendix J	1 and 2	1 and 2A
	5 and 6	5A and 6
	19 and 20	19 and 20A
	21 and 22	21 and 22A
	N.A.	26aA,26bA,26cA and 26dA**
	27 and 28	27A and 28A
	N.A.	28aA and 28bA**
Index	3 and 4	3 and 4A
	13 and 14	13 and 14A
	19 and 20	19 and 20A

All the technical changes in an update are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

PAGE STATUS SUMMARY

ISSUE: Update A to UP-7629 Rev. 2

Section	Page Number	Update Level
Cover/Disclaimer		A
PSS	1	A
Contents	1 2 thru 5 6 7 thru 9	A Orig. A Orig.
1	1 2 thru 6	A Orig.
2	1, 2 3 4 5 thru 9 10 11, 12 12a 13 thru 15 16 17 18 19 20 21 thru 25 26 27 28 29 30, 31 32 33 thru 35 36 37	A Orig. A Orig. A Orig. A* A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig.
3	1 thru 16 17, 18 19 thru 22 23 24 25, 26 27, 28 29 30 thru 34 35 36 thru 39 40 41 thru 76 77 78 thru 85 86	Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A
4	1 2, 3 4	Orig. A Orig.

Section	Page Number	Update Level
4 (cont.)	5 6 thru 14 15, 16 17, 18 19 20 thru 27 28, 29 30 31 32 33 34, 35 36 37, 38 39 40 thru 42 43 44 45 46 thru 49 50 51 thru 64	A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig. A Orig.
5	1, 2 3 4, 5 6 7 8 thru 11	A Orig. A Orig. A Orig.
Appendix A	1 thru 5 6 7 thru 13	Orig. A Orig.
Appendix B	1 thru 6	A
Appendix C	1, 2 3 4	A Orig. A
Appendix D	1 thru 3	Orig.
Appendix E	1 thru 3	A
Appendix F	1 thru 7	Orig.
Appendix G	1 thru 5 6 7 8 9 10, 11	A Orig. A Orig. A Orig.
Appendix H	1 thru 3 4 thru 9	A Orig.
Appendix I	1	Orig.

Section	Page Number	Update Level
Appendix J	1 2 3, 4 5 6 thru 19 20 21 22 23 thru 26 26a 26b 26c 26d 27, 28 28a 28b 29	Orig. A Orig. A Orig. A Orig. A Orig. A* A* A* A* A A* A* Orig.
Index	1 thru 3 4 5 thru 13 14 15 thru 19 20 21	Orig. A Orig. A Orig. A Orig.
User Comment Sheet		
Total 334 pages including covers		

CONTENTS

APPENDIXES

FIGURES

TABLES

1. INTRODUCTION

1.1. GENERAL

This manual describes the Data Management facilities provided in the UNIVAC OS/4 Operating System. It includes descriptions of the file conventions accepted by Data Management and the macro instructions used in sequential and nonsequential methods of accessing files. A knowledge of the *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version) and *UNIVAC 9400 System Assembler/Central Processor Unit Programmer Reference, UP-7600* (current version), is helpful in using this manual.

Data Management is part of the software that provides a convenient interface between problem programs and the hardware-oriented I/O portions of the Supervisor. Data Management facilities provide organizational benefits such as record blocking and deblocking, buffering, data validation, and label processing.

Data Management facilities consist of logical Input/Output Control System (IOCS) modules, transient routines, and declarative and imperative macro instructions. The logical IOCS modules consist of re-entrant common code subroutines. The transient routines provide functions which are infrequently used and therefore do not need to be available during the complete file processing. A macro instruction is similar in form to a source code instruction; it may have a label, but it must have an operation code and an operand field containing one or more parameters.

The parameters used with the declarative macro instructions describe all aspects of the file to be processed, whereas the parameters used with the imperative macro instructions point to the file described by a declarative macro and sometimes add additional details specifying processing action to be taken.

1.1.1. Logical IOCS Modules

Logical IOCS modules consist of re-entrant common code subroutines that can become part of the operating system, can be linked into the problem program, or a combination of both.

At system generation time the user can select those logical modules or parts of modules that are needed to process the files that are used most often. The result is a saving in storage space since a subroutine can be shared by all files rather than repeating file processor coding in each problem program that is loaded. Any module so chosen becomes a part of the operating environment and is replaced in this form in the systems library by abbreviated modules that contain only special symbol definitions. Thus, references to these modules within a problem program produce a special output rather than the entire module. This enables logical IOCS to establish linkage to the code chosen as part of the operating system. This option is particularly beneficial when the user is operating in a multiprogramming environment.

Any module that is not chosen at system generation time to become part of the operating system can remain as part of the system library. When logical IOCS references these modules, the code for the entire module is linked directly into the problem program. The modules are then shared among the files within the problem program. This option is particularly beneficial when the user is operating in a single program environment.

1.1.2. Declarative Macro Instructions

A problem program must inform the system of the parameters, special conditions, current status and options pertaining to a file. This is accomplished by including a declarative (file definition) macro instruction for each file required by the problem program. As implied by the term declarative, these macro instructions generate nonexecutable code, such as constants and storage areas for variables. Therefore, these macro instructions should be separated from the inline file processing coding. The declarative macro instruction and the selected *keyword* parameters in the operand define the file. The first three characters of the operation code must be DTF, meaning *D*efine *T*he *F*ile. The last two characters usually indicate the type of device or method of accessing. A keyword parameter consists of a word or code immediately followed by an equals (=) sign which is in turn followed by one specification.

The format of the declarative macro instruction is:

LABEL	ƀ OPERATION ƀ	OPERAND
filename	DTFcc	keyword-1=x,keyword-2=y,...,keyword-n=z

The symbolic name of the file *must* appear in the label field. It can have a maximum of *seven* characters and must begin with an alphabetic character. The appropriate DTF designation must appear in the operation field. The keyword parameters can be written in any order in the operand field and must be separated by commas. Appropriate assembler rules regarding macro instructions apply to blank columns and continuation statements. Register numbers are specified to the Data Management declarative macro instructions (DTF's) by enclosing the number in parentheses.

1.1.3. Imperative Macro Instructions

A problem program must be able to communicate with logical IOCS in order to accomplish the processing of files that have been defined by declarative macro instructions. This is accomplished by including imperative (file processing) macro instructions in the problem program, which in turn communicate with the transient routines and modules of logical IOCS coding. The imperative macro instructions are expanded as inline executable code. Not all macro instructions are available for use on all devices. Some are specifically input type macro instructions and cannot be used for a device that is exclusively used for output; the opposite is true, also.

The format of the imperative macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	xxxx	yyyy,...,zzzz

A symbolic name can appear in the label field. It can have a maximum of eight characters and must begin with an alphabetic character. The appropriate verb or code must appear in the operation field. The positional parameters (as signified by the name) *must* be written in the specified order in the operand field and be separated by commas. When a positional parameter is omitted, the comma must be retained to indicate the omission except in the case of omitted trailing parameters. Appropriate assembler rules regarding macro instructions apply to blank columns and continuation statements.

1.2. STATEMENT CONVENTIONS

The conventions used to illustrate statements in the manual are as follows:

- Capital letters and punctuation marks (except braces, brackets, and ellipses) are information that must be coded exactly as shown;
- **Lowercase letters and terms represent information that must be supplied by the programmer;**
- Information contained within braces { } represents necessary entries of which one must be chosen;
- Information contained within brackets represents optional entries that (depending on program requirements) are included or omitted. Braces within brackets signify that one of the entries must be chosen if that operand is included.

An ellipsis (a series of three periods) indicates the presence of a variable number of entries.

Commas are required after each parameter except after the last parameter specified. When a positional parameter is omitted from within a series of parameters, the comma must be retained to indicate the omission.

1.3. SPECIAL REGISTER NOTATION

The user can preload parameter registers 0 and/or 1 prior to executing an imperative macro instruction. When the register option is selected, the actual designations (0) and/or (1) are coded signifying that the register(s) are loaded with the necessary parameter(s) as specified by logical IOCS. This is known as special register notation.

1.4. DATA MANAGEMENT REGISTER CONVENTIONS

Logical IOCS assumes that registers 0 and 1 are volatile. These two registers usually contain positional parameters. Register 13 is assumed to have been preloaded by the user with the address of a 72-byte save area (aligned on a full word boundary), and it is used to access that area. Registers 14 and 15 are considered volatile and available for use by logical IOCS. For macro instructions that do not generate a call to a transient routine, register 14 is used to transfer control to the user following an imperative macro instruction, while register 15 is used to pass control from the macro instruction to the processing modules. For macro instructions that do generate a transient call, the contents of registers 14 and 15 are preserved by the transient scheduler.

1.5. DATA MANAGEMENT DUMMY CONTROL SECTION

A special macro instruction (DTFD) is used to generate a dummy control section which can be shared by any or all of the logical IOCS modules included in the user's system. The dummy control section (DSECT) permits the user to define symbolic addresses without any associated storage allocation; that is, it allows the mapping or remapping of a storage area already defined within the module or elsewhere without any additional allocation of space. Displacement and base relative addresses are calculated by the assembler for each symbol defined by the DSECT, but they do not become part of the object program.

Only the unprotected portions of the DTF's are affected by the DTFD macro call. Keyword parameters associated with this instruction permit the user to specify any or all of the DTF expansions he wants covered by the generated dummy control section. The resulting DSECT code will consist of a common section (common to all DTF's), followed by individual sections that correspond to the number of parameters selected in the DTFD macro instruction. The DTFD macro may be called several times within a program with the assurance that no symbol will be generated more than once.

NOTE: In order to provide for I/O storage protection, certain critical portions of each DTF macro instruction are separated from the rest of the program unit and placed in a protected area. When a DTF routine forms an address to be used in an input operation, for example, it checks that the operation refers only to those storage locations allocated to the routine and does not violate the I/O protect area.

1.5.1. Instruction Format

The DTFD macro instruction contains a maximum of six optional keyword parameters. The instruction format is:

LABEL	♭ OPERATION ♭	OPERAND
unused	DTFD	[MT=YES] [SD=YES] [CD=YES] [PR=YES] [PT=YES] [OR=YES] [DA=YES] [IS=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in operand field apply when writing this macro instruction; see 1.2.

where: MT provides a map of the unprotected DTF storage area generated by a call to DTFMT (magnetic tape file).

SD provides a map of the unprotected DTF storage area generated by a call to DTFSD (sequential device file).

CD provides a map of the unprotected DTF storage area generated by a call to DTFCD (card device file).

PR provides a map of the unprotected DTF storage area generated by a call to DTFPR (printer file).

PT **provides a map of the unprotected DTF storage area generated by a call to DTFPT (paper tape file).**

OR **provides a map of the unprotected DTF storage area generated by a call to DTFOR (optical document reader).**

DA provides a map of the unprotected DTF storage area generated by a call to DTFDA (direct access file).

IS provides a map of the unprotected DTF storage area generated by a call to DTFIS (indexed sequential file).

NOTES: If no parameters are specified in the operand field, a DSECT that includes all options will be generated.

This macro instruction may be used to request mapping of more than one type of DTF; if so, operands must be separated by commas.

The format of the generated DSECT includes a common unprotected section (containing labels of the form DC$xxx) which is common to all DTF types, and which is concluded by the definition of an overlay point (DC$OVL). Following the overlay point, any additional sections are generated according to the DTF's requested in the DTFD operand field.

1.5.2. DSECT Usage

In order to reference standard symbols within the DSECT, the user must supply a USING statement referencing label DC$SCT (the DSECT starting point). The register specified in the USING statement must be the register containing the address of the DTF macro instruction to be manipulated.

For example:

```
USING DC$SCT,2
```

This specifies that all DSECT-generated symbols would be covered by register 2.

Prior to use of the DSECT symbols, register 2 must be loaded with the address of the required DTF macro instruction; that is,

```
L  2,=A(filename)
```

where filename is the symbol appearing in the label field of the actual DTF.

2. FILE CONVENTIONS

2.1. GENERAL

A set of conventions has been adopted to facilitate the handling, locating, and processing of data files by the UNIVAC OS/4 Operating System (OS/4) Data Management System. The form of the files may vary somewhat from one device type to another, yet there are large areas where their conventions overlap. In the section that follows, explicit formats show these conventions (which are compatible within the industry).

As a means of identification, files are assumed to have labels in a standard format. Files may be combined to span a volume. A volume is a reel of tape or a disc pack cartridge. Every volume in a system can be numbered and have a volume label recorded on it. On direct access storage devices, files may be divided into extents. An extent is any contiguous space on the device. Therefore, files may consist of one extent, or, if the file is fragmented, it may consist of several extents. On direct access storage devices, files may be stored in any available area on the device and an index to the various fragments of the file is maintained. This index or directory is called the volume table of contents and is provided to record the allocation of space for the files on the volume.

2.2. MAGNETIC TAPE CONVENTIONS

Magnetic tapes may or may not be labeled and a labeled tape may have either standard or nonstandard labels. The Data Management System assumes tapes have standard labels but permits other label conventions. Tape volumes have formats which may be specified as standard, nonstandard, or unlabeled. Record types and the form for checkpoint blocks are also specified. The following paragraphs describe the magnetic tape conventions handled by the UNIVAC OS/4 Data Management System.

2.2.1. Standard Tape Labels

All standard tape labels are in blocks of 80 bytes. Tape labels are always recorded at the same density as the data. There are five types of tape label groups (three are required and two are optional):

- Volume label group
- File header group
- User header group (optional)
- File trailer group
- User trailer group (optional)

2.2.1.1. Volume Label Group

A volume label group is composed of an initial volume label (VOL1) followed by up to seven volume labels (VOL2 ... VOL8). The VOL1 label identifies the tape reel and its owner and is used to check that the proper reel is mounted. When a tape is first used at an installation, the serial number and other volume information as shown in Figure 2–1 are specified by parameter cards supplied to a standard utility routine which writes the label. The serial number is also written on the exterior of the reel for visual identification.

At job preparation time, the serial number of the volume to be used may be supplied to the Job Control program along with other parameters. If supplied, the Job Control program checks the serial number from the VOL1 label to ensure that the proper reel is mounted for input or output. The tape is then rewound without interlock. See *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version).

When an OPEN macro instruction is issued for an output tape, the VOL1 label is read and the serial number is saved for use in the file header labels. The tape is positioned so that the volume labels are not destroyed and no further processing is performed.

When an OPEN macro instruction is issued for an input tape, the volume serial number equal to that in the VOL1 label is used to check the file serial number in the file header label. The tape is positioned past the last volume label and no further volume label processing is performed.

When any volume label is encountered during the processing of a CLOSE macro instruction for an input tape, and the READ parameter is set to BACK, the label is simply bypassed without processing.

The formats of the volume labels are shown in Figures 2–1 and 2–2.

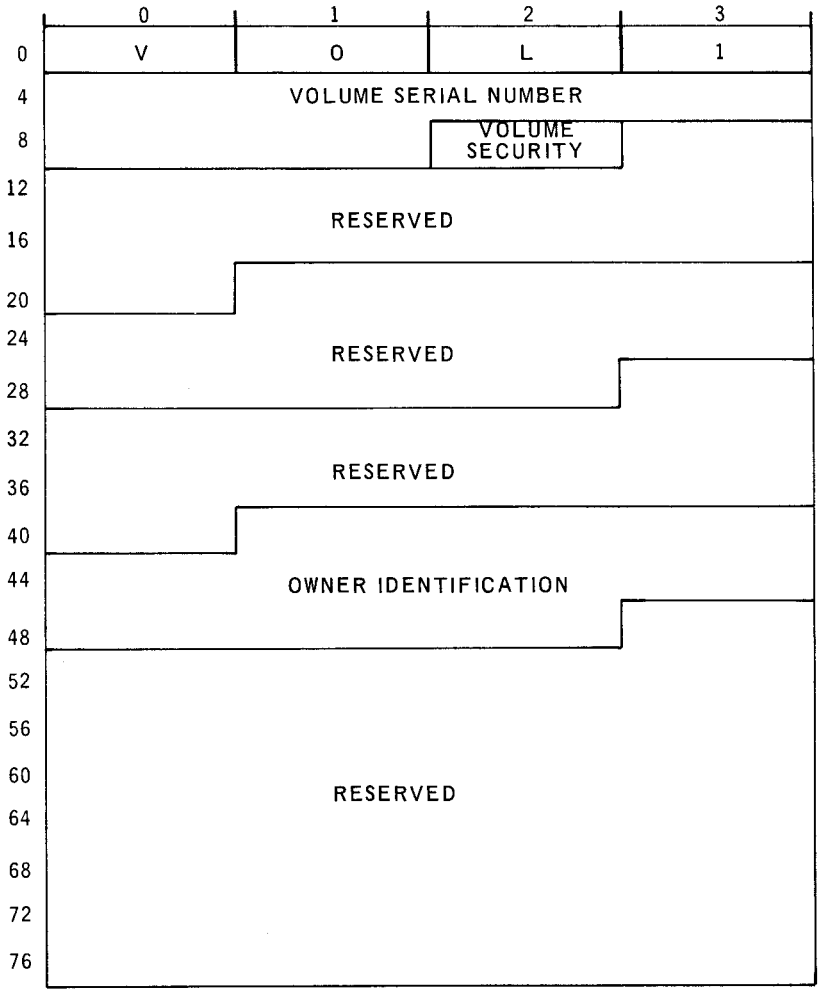


Figure 2–1. Tape Volume 1 Label Format

Field	Code	
■ Bytes 0–2	EBCDIC	Label Identifier – Contains VOL to indicate that this is a volume label.
■ Byte 3	EBCDIC	Label Number – Always 1, for the initial volume label.
■ Bytes 4–9	EBCDIC	Volume Serial Number – Unique identification assigned to a reel when it enters the system. This number also appears on the outside of the reel. It is usually numeric but may be any six alphanumeric characters.
■ Byte 10	EBCDIC	Volume Security – Reserved for future use by installations requiring security at the reel level; this is currently blank.
■ Bytes 11–20	EBCDIC	Reserved – Contains blanks.
■ Bytes 21–30	EBCDIC	Reserved – Contains blanks.
■ Bytes 31–40	EBCDIC	Reserved – Contains blanks.
■ Bytes 41–50	EBCDIC	Owner Identification – Unique identification of the owner of the reel. It may be any combination of alphanumeric characters.
■ Bytes 51–79	EBCDIC	Reserved – Contains blanks.

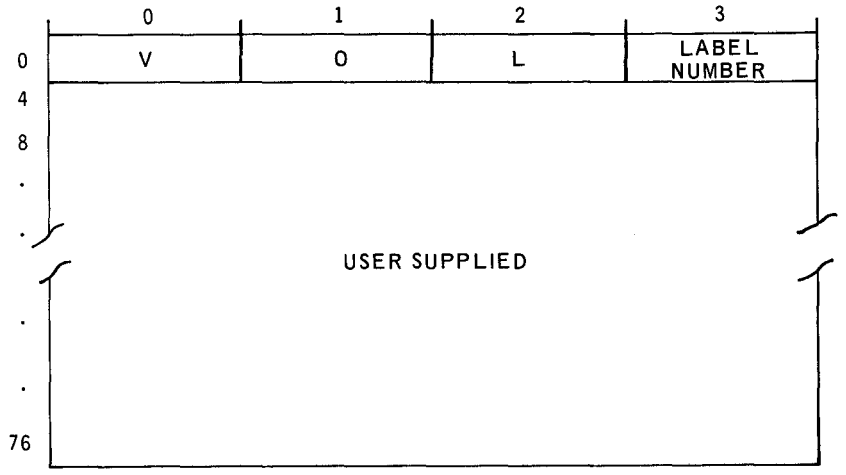


Figure 2–2. Additional Volume Labels Format

Field	Code	
■ Bytes 0–2	EBCDIC	Label Identifier – contains VOL to indicate a volume label.
■ Byte 3	EBCDIC	Label Number – 2 through 8, indicating the position of the volume label within the group.
■ Bytes 4–79	EBCDIC	User supplied.

2.2.1.2. File Header Label Group

A file header label, HDR1, is written at the beginning of each file. This label is required and has a fixed format. The HDR1 label identifies the file. The format is shown in Figure 2–3.

On an input file, all fields up to and including the creation date in the HDR1 label are checked against values specified in the LBL Job Control statement. Only those fields for which values have been supplied are checked; however, when the READ parameter is set to BACK, the HDR1 label is bypassed without processing.

For an output file, the expiration date in the HDR1 label is checked against the current or actual calendar date to determine if the associated file has expired. If it has expired, the tape is positioned so that the old HDR1 label is written over. The new HDR1 block is set up from values specified by the LBL Job Control statement and is written on the tape. All of the fields in the HDR1 label may be supplied in the job stream. See the LBL statement described in *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version), and Appendix A.1 in this manual.

The user also has the option of using HDR2 ... HDR8 labels, which are reserved for compatibility with other systems. However, Data Management does not create these labels for output files, but the user may write them on output or check them on input in a user label routine.

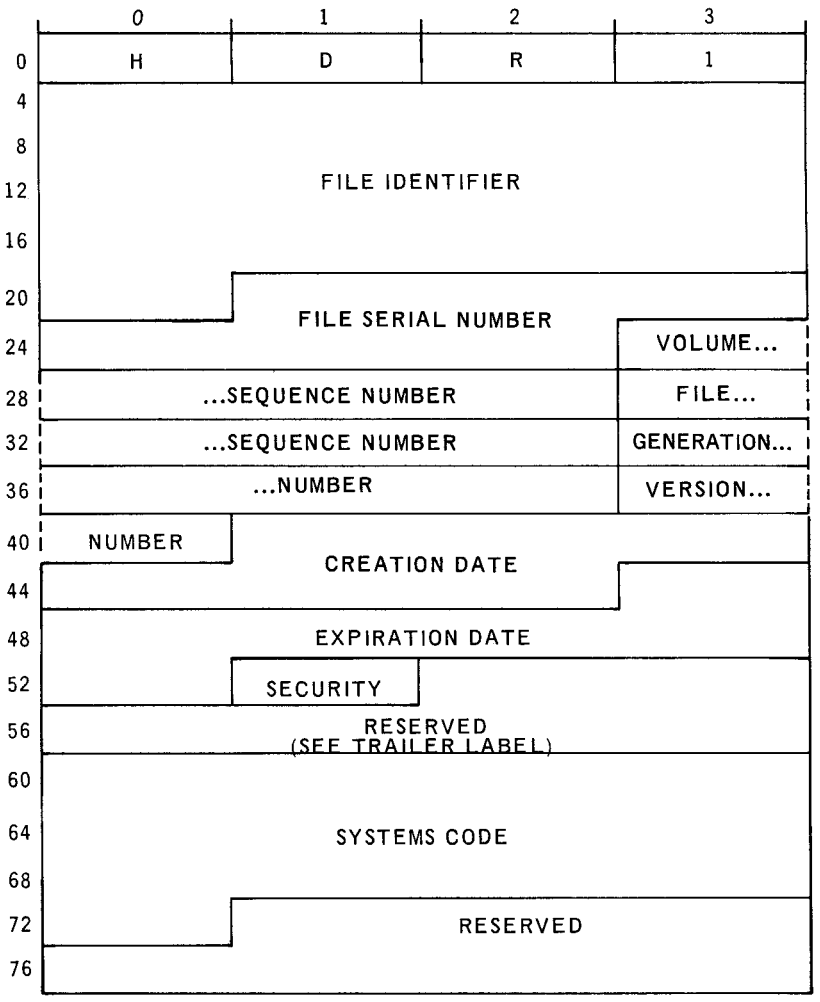


Figure 2–3. Header 1 Label Format

Field	Code	
■ Bytes 0–2	EBCDIC	Label Identifier – Contains HDR to indicate a file header label.
■ Byte 3	EBCDIC	Label Number – Always 1.
■ Bytes 4–20	EBCDIC	File Identifier – A 17-byte configuration that uniquely identifies the file. It may not contain embedded blanks and is left justified in the field if less than 17 bytes are specified.
■ Bytes 21–26	EBCDIC	File Serial Number – The serial number of the VOL1 label for the first reel of a file or a group of multifile reels.
■ Bytes 27–30	EBCDIC	Volume Sequence Number – The position of the current reel with respect to the first reel on which the file begins. This is used with multireel files.
■ Bytes 31–34	EBCDIC	File Sequence Number – The position of this file with respect to the first file in the group.
■ Bytes 35–38	EBCDIC	Generation Number – The generation number of the file (0000–9999).

Field	Code	
■ Bytes 39–40	EBCDIC	Version Number of Generation – The version number of a particular generation of the file.
■ Bytes 41–46	EBCDIC	Creation Date – The date on which the file was created. The date is expressed in the form yyddd and is right justified. The leftmost position is a blank.
■ Bytes 47–52	EBCDIC	Expiration Date – The date on which the file may be written over or used as scratch. The date is in the same form as the creation date.
■ Byte 53	EBCDIC	Reserved for File Security Indicator – This byte indicates whether additional qualifications must be met before a problem program can have access to the file. 0 = No additional qualifications are required. 1 = Additional qualifications are required.
■ Bytes 54–59	EBCDIC	Unused – Contains zeros.
■ Bytes 60–72	EBCDIC	Reserved for System Code – The unique identification of the operating system that produced the file.
■ Bytes 73–79	EBCDIC	Reserved – Contains blanks.

2.2.1.3. User Header Label Group

Up to eight user header labels can follow the file header label group. For output files, the labels are written by Data Management as directed by the problem program recording the file.

When the file is retrieved, the user header label group is made available to the problem program for processing. Conventions describing this procedure can be found in the description of the OPEN and LBRET macro instructions (see 3.3.1 and 3.3.7, respectively). The format of the user header label is shown in Figure 2–4.

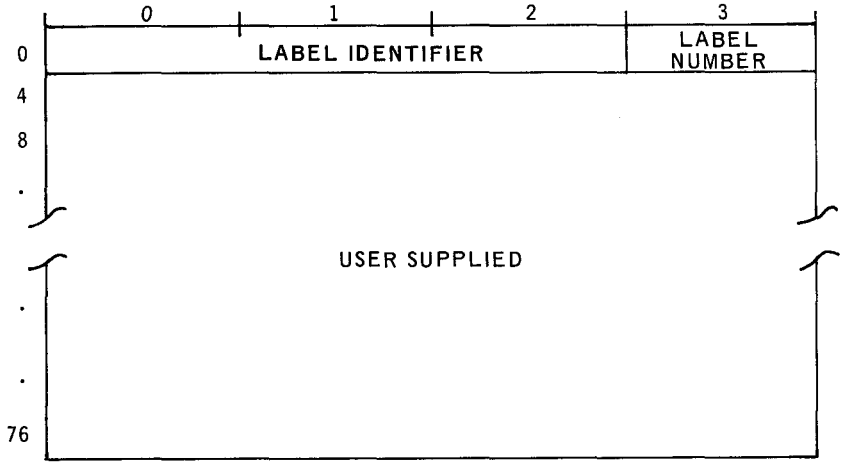


Figure 2–4. User Header Label Format

Field	Code	
■ Bytes 0–2	EBCDIC	Label Identifier – Contains UHL to indicate a user header label.
■ Byte 3	EBCDIC	Label Number – This field gives the relative position (1 to 8) of the label within the group.
■ Bytes 4–79	EBCDIC	User supplied.

2.2.1.4. File Trailer Label Group

The file trailer label group consists of either of two labels written in a format identical to the file header labels, shown in Figure 2–3, except for the following fields:

■ Bytes 0–2	EBCDIC	Label Identifier – The contents indicate that this is a file trailer label and may be either EOF for an end-of-file label or EOV for an end-of-volume label.
■ Byte 3	EBCDIC	Label Number – Indicates whether the label is the first or second (optional) file trailer label.
■ Bytes 54–59	EBCDIC	Block Count – In the first file trailer label, this field indicates the number of data blocks either in the file of a multifile reel or on the current reel of a multireel file. Logical IOCS checks the block count for input files, and writes the count for output files.

When an OPEN macro instruction is issued on an input file with READ = BACK specified, the fields in an EOF1 label are checked against values specified in the LBL Job Control statement. This is similar to processing of the HDR1 label as described in 2.2.1.2.

2.2.1.5. User Trailer Label Group

Up to eight user trailer labels may follow the file trailer label group. User trailer labels are processed in the same manner as user header labels. They have the same format as well, except that the first field contains UTL instead of UHL. Figure 2–4 and the notes which follow also illustrate and describe the format of the user trailer label.

2.2.2. Reel and File Organization

The macro instruction that defines the file for magnetic tapes must specify whether the tape volumes are standard, nonstandard, or unlabeled. The following three subsections describe the organization of and requirements for files and reels with respect to these three conventions.

2.2.2.1. Standard Tape Volume Organization

A standard volume has standard labels, requires tape marks, and is capable of being processed by logical IOCS. Figure 2–5 illustrates the reel organization for a standard volume with either an end-of-file or end-of-volume condition. Logical IOCS assumes that the labels appear in the order shown. The labels that are shaded are optional. Any optional header or trailer labels must be processed by the user's special label handling routine. If the user does not specify such a routine, the optional labels are simply bypassed.

On input operations, the additional volume labels are not checked by logical IOCS. If any exist, they are bypassed. On output operations, no provision is made in logical IOCS for the user to write additional volume labels. Logical IOCS also has no provisions for handling a multifile reel or a group of associated multireel files.

A volume processed by logical IOCS will have either an end-of-file label or an end-of-volume label followed by two tape marks. The second tape mark signifies that no valid information follows.

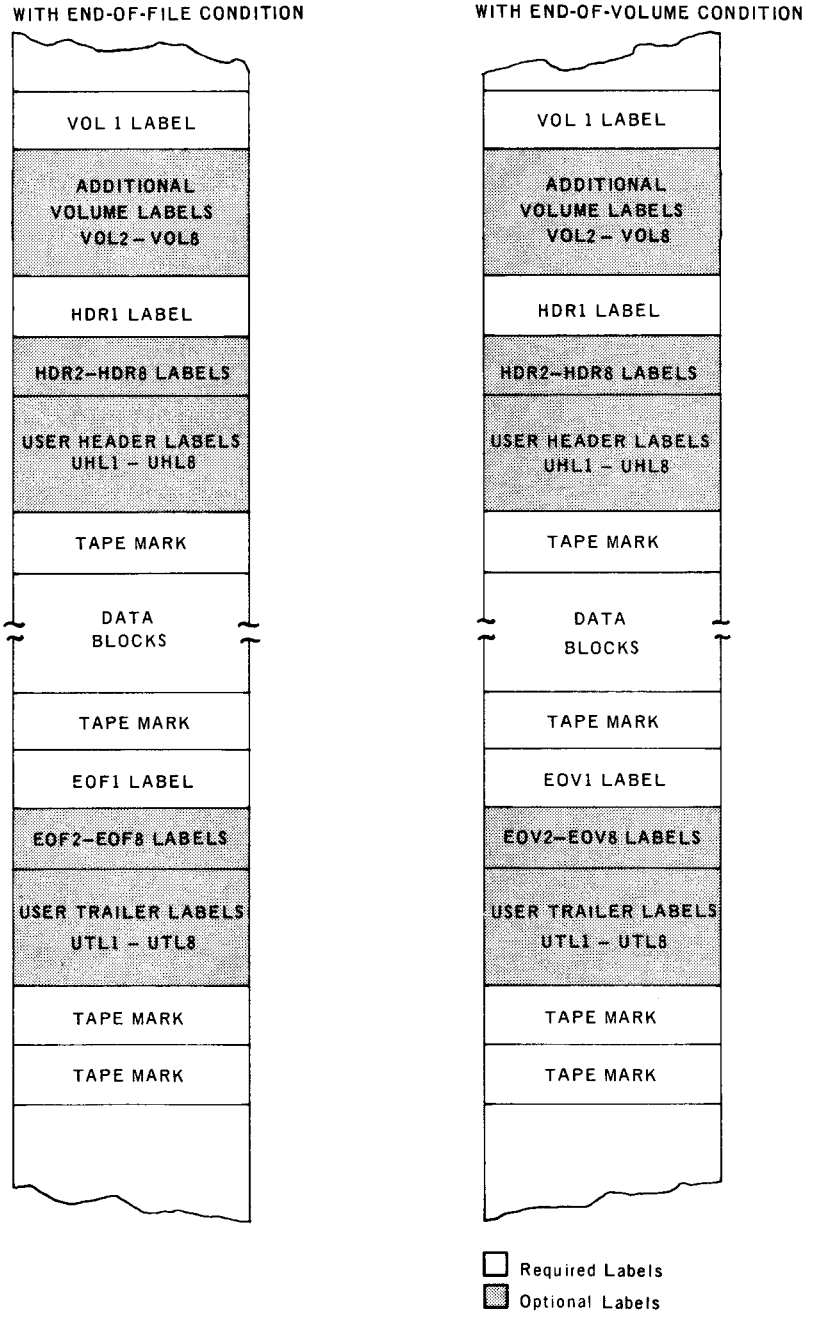


Figure 2–5. Reel Organization for a Standard Volume

2.2.2.2. Nonstandard Volume Organization

A nonstandard volume is any volume that has nonstandard labels and is capable of being processed by logical IOCS. The reel organization for a nonstandard volume is shown in Figure 2–6.

The address of a user's label handling routine to process nonstandard labels is usually specified; in which case the tape mark following the header labels may be omitted. If nonstandard header labels appear on an input file but are not to be checked when the file is read, the address of the user's label handling routine is omitted. In this case, the tape mark must appear.

The header and trailer labels are optional and may be of any format length and number since they are handled by the user's label routine. The tape mark following the header label is required only if label checking is omitted or a read backward operation is specified. The tape marks following the data blocks and the trailer label are required and are written by logical IOCS. The second tape mark following the trailer label is written if the tape is to be rewound with or without interlock on output operations.

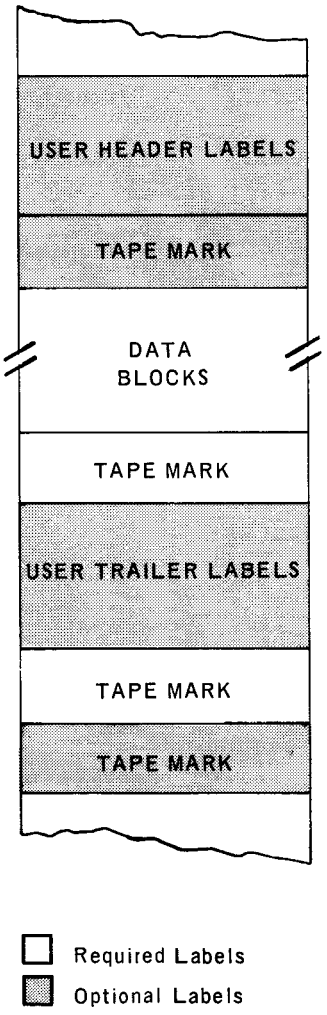


Figure 2–6. Reel Organization for a Nonstandard Volume

2.2.2.3. Unlabeled Volume Organization

An unlabeled volume is any volume capable of being processed by logical IOCS. The fact that the volume has no labels and may or may not have a tape mark preceding the data blocks must be specified in the declarative macro instruction. The reel organization for a volume with no labels is illustrated in Figure 2–7. It should be noted that on read backward operations, the tape mark preceding the data blocks must be present.

The tape mark following the data blocks is required and is supplied by logical IOCS on output operations. An additional tape mark is written if the tape is to be rewound with or without interlock on output operations.

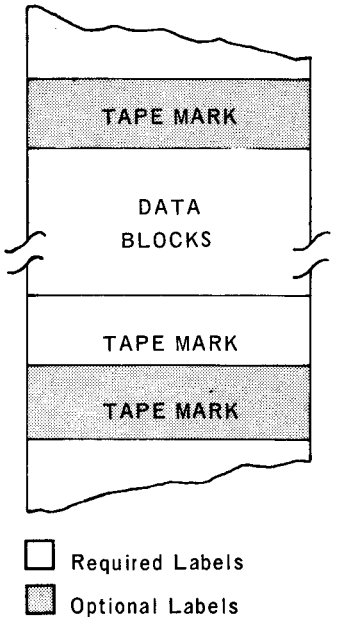


Figure 2–7. Reel Organization of a Volume Without Labels

2.2.3. Checkpoint Blocks

The first and last blocks of a checkpoint dump begin with the following configuration:

→ bbb///bCHKPTb//xxyynnnn

where:

→ bbb **is an optional block number.**

b **represents a blank.**

→ xx **is a two-byte binary count of storage dump records between the header and the trailer records.**

→ yy **is the two-byte total number of words following the header unpacked decimal counter.**

nnnn **is the checkpoint number assigned.**

Checkpoint blocks are bypassed by logical IOCS on input operations when the proper keyword parameter is specified in the macro instruction defining the file.

Checkpoint blocks are not reflected by the total in the block count field in either the EOF1 or the EOV1 standard trailer labels.

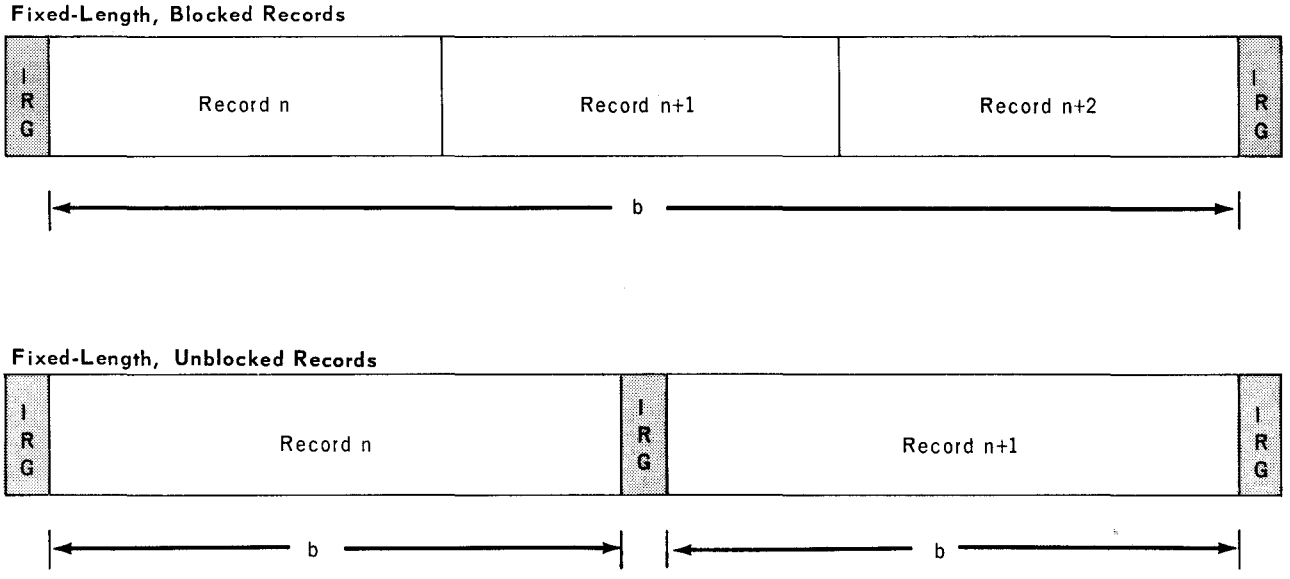
2.2.4. Record Formats

Records within a file may be grouped into blocks when more than one logical record equals a physical record. Records may be unblocked if only one logical record equals a physical record.

The format of individual records may be fixed length, variable length, or undefined. If the record format is undefined, the records are also assumed to be unblocked and any deblocking must be supplied by the user.

2.2.4.1. Fixed-Length Records

If the size of a record in a file is a fixed length, the size of a block is usually a fixed length. The block size is always an integral multiple of the record size. A TRUNC macro instruction may cause a short block to be written; the block written is still a multiple of the record size. An illustration of fixed-length records is given in Figure 2–8.



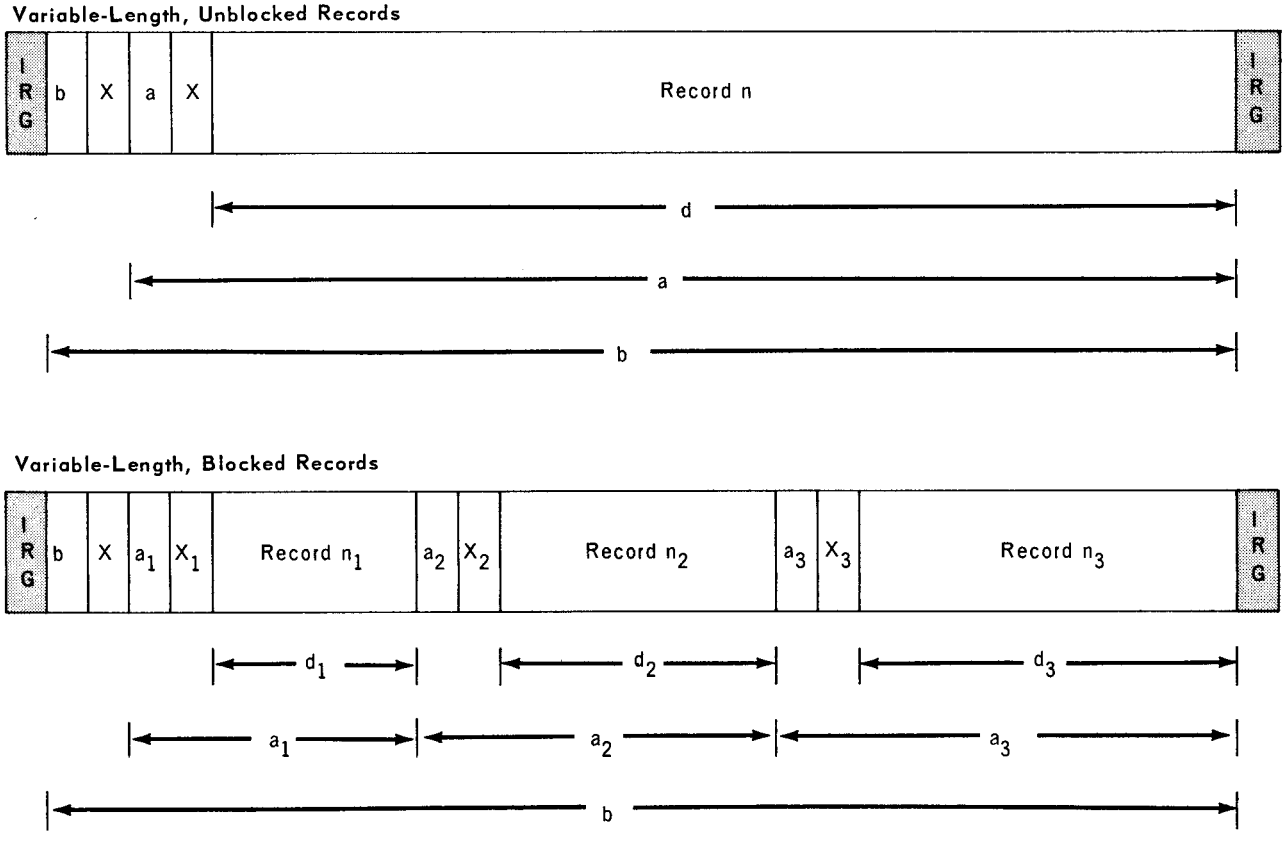
LEGEND:

IRG – Interrecord gap
n – Any data record in the file
b – Block size

Figure 2–8. Magnetic Tape Fixed-Length Records

2.2.4.2. Variable-Length Records

Files with variable-length records may have either blocked or unblocked records. The length of the block must be specified and must precede the record or records in the block. The number of bytes in a variable-length record must be specified and must precede the data in the record. Figure 2–9 illustrates the format of both blocked and unblocked records.



LEGEND:

b – Block length (two bytes)
a – Record length (two bytes)
X – Reserved for system use (two bytes)
d – Data field in the record
IRG – Interrecord gap

Figure 2–9. Magnetic Tape Variable-Length Records

2.2.5. Processing 7-Track Tape with the Data Conversion Feature

When the data conversion feature is being used for writing, three 8-bit data bytes are converted to four 6-bit tape characters. Conversion is accomplished in multiples of three data bytes. If the last group contains less than three data bytes, the group is zero filled by the hardware to make up the equivalent of three bytes. In effect, the hardware always writes blocks in multiples of three.

Only those tapes which have been converted to four 6-bit tape characters can be read by the data conversion feature. Tapes must be written in a format that will enable them to be read. These are the only restrictions imposed on writing tape. When reading tape, Data Management compares the specified block size with the number of bytes actually read in. If the hardware has added any bytes to make the block size a multiple of 3, Data Management will detect a block size error and either cancel the job or exit to a user-supplied error routine. This can be avoided by ensuring that each block written is a multiple of 3.

For reading tape forward, the following precautions should be taken:

- If the record format is fixed unblocked, the block size must be a multiple of 3.
- If the record format is fixed blocked, the record size must be a multiple of 3.
- If the record format is variable unblocked or blocked, the total block length must be a multiple of 3. For unblocked records, this block length is equal to the length of the data field, plus 8. For blocked records, this block length is equal to the sum of the length of the data fields, plus 4 times the number of records, plus 4 (Figure 2—9).
- If the record format is undefined, there are no restrictions. The user is cautioned, however, that if the block length is not a multiple of 3, the last one or two bytes will contain 0's, and these bytes will be counted in the record length passed to the user by Data Management.

For reading tape backward, the same restrictions apply. In addition, if the record format is undefined, each block must be a multiple of 3.

2.3. DIRECT ACCESS STORAGE DEVICE CONVENTIONS

Conventions for files on the direct access storage subsystem are similar to those for files on magnetic tapes. However, there are some differences.

The method of addressing volumes, files, and records on the disc uses a combination of the logical unit number of the device (n = 1—8), the cylinder number (cc = 0—202 for the UNIVAC 8411 and 8414 Disc Subsystems, or cc = 0—405 for the UNIVAC 8424 Disc Subsystem), the read/write head number (hh = 0—9 for the UNIVAC 8411 Disc Subsystem, or hh = 0—19 for the UNIVAC 8414 and 8424 Disc Subsystems), and the sequential number of the record (r) on the track. The value of the record number may range from 0 through 255; however, track capacity is considerably less depending on the size of the records. The last three cylinders of a volume are reserved for alternate tracks. (In the address of a disc record, the value 00 is supplied for Bb.)

Because the files within a volume may be stored in various locations on a disc, a directory listing the addresses of the fragments of the files is required. This directory is called the volume table of contents. The volume table of contents and files within a volume on a disc require various standard labels in predefined formats to describe the files.

The standard disc labels include the volume label and five types of format labels. The volume label includes the address of the volume table of contents which is normally the next group of records. The user also has the option of specifying additional volume, header, and trailer labels for a disc volume. Format labels contain such information as upper and lower limits, identifiers, indicators, lengths, counters, addresses, and pointers. The labels are illustrated and described in the paragraphs that follow.

The volume table of contents starts at the beginning of a track if it does not follow the volume label group and must terminate at the end of a track. It occupies one extent, which is specified when the device is initialized.

The first record in the volume table of contents is a format 4 label which describes the table of contents itself. The second record is a format 5 label, which maintains control of the available space in the volume. These two labels are written when the volume is initialized and the table of contents is set up. Following the format 4 and 5 labels are various combinations of format 1, 2, 3, and other format 5 labels. Every file and every extent of a file must be accounted for by these labels.

Refer to *UNIVAC 9400 System Supervisor Programmer Reference, UP-7689* (current version) for a complete listing of the standard equate labels (STDEQU's) which appear in the following tabulations.

2.3.1. Disc Volume Labels

Any disc volume may have one or more volume labels. A four-byte key precedes each volume label and is the same as the first four bytes of the volume label itself. These first four bytes in the volume label contain the identifier VOLn, where n is a number from 1 through 8. The VOL1 label is the standard volume label in the system. Others, if present, are user volume labels. The volume labels for discs are used in much the same way as tape volume labels (see 2.2.1). The formats of the disc volume labels are shown in Figures 2—10 and 2—11.

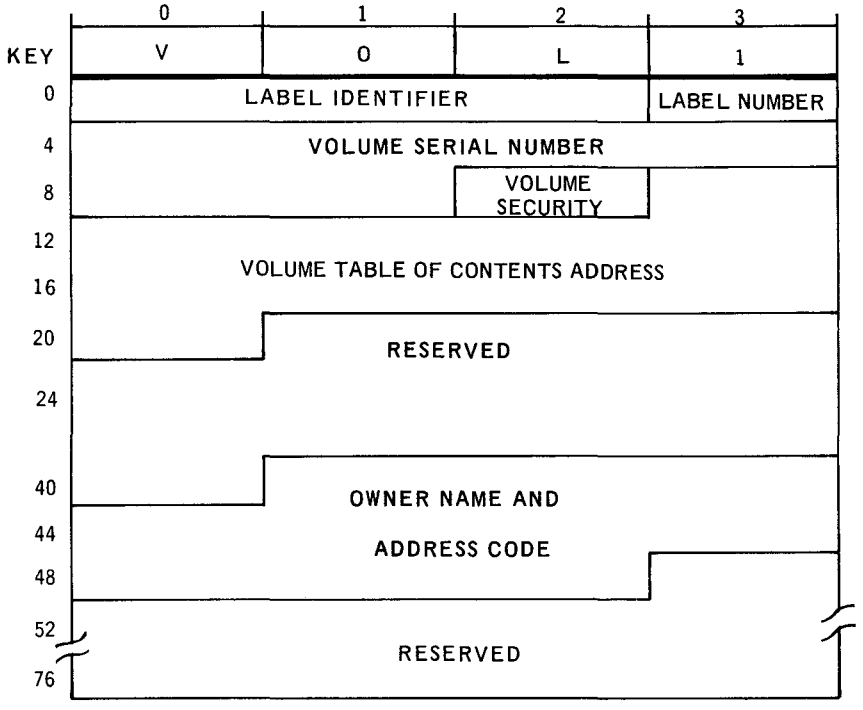


Figure 2–10. Disc Volume 1 Label Format

STDEQU Label	Field	Code	
■ DL$VL	Key		The key contains VOL1.
■ DL$VL1	Bytes 0–2	EBCDIC	Label Identifier – Contains VOL.
■ *	Byte 3	EBCDIC	Label Number – Always 1, for initial volume label.
■ DL$VSN	Bytes 4–9	EBCDIC	Volume Serial Number – A unique code assigned to a disc pack when it enters the system. The same code should appear visually on the disc pack for operator identification.
■ DL$VSB	Byte 10	Binary	Volume Security – 0 = No further identification is required for each file on the volume. 1 = Further identification is required.
■ DL$VTC	Bytes 11–20	Discontinuous Binary	Volume Table of Contents Address – This field is used on disc volumes to point to the format 4 label which starts the table of contents of that volume. In bytes 11 through 15, the address is of the form cchhr (where cc is the cylinder number, hh is the head number, and r is the record number). Bytes 16 through 20 are 0.

*No STDEQU label applicable.

STDEQU Label	Field	Code	
■ *	Bytes 21–40		Reserved.
■ DL$ONR	Bytes 41–50	Optional	Owner Name and Address Code – A user supplied, installation identifier.
■ *	Bytes 51–79		Reserved.

*No STDEQU label applicable.

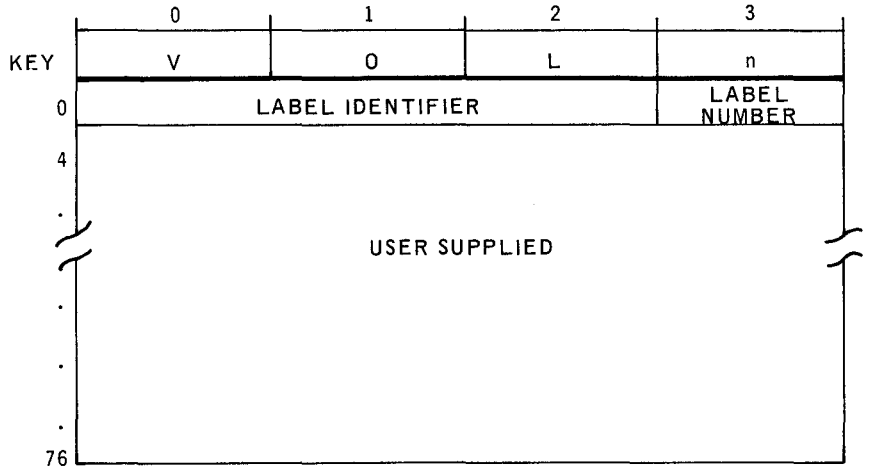


Figure 2–11. Disc User Volume Label Format

Field	Code	
■ Key		The key contains VOLn, where n may be 2 through 8.
■ Bytes 0–2	EBCDIC	Label Identifier – Contains VOL to indicate a volume label.
■ Byte 3	EBCDIC	Label Number – 2 through 8, indicating the position of the volume label within the group.
■ Bytes 4–79		User supplied.

2.3.2. Disc Format 1 Labels

A format 1 label exists for each file in a volume. It is equivalent to the HDR1 label on tape and includes similar fields. As many as three extents of the file may be described in the format 1 label. The format 1 label contains a pointer which, if used, indicates the address of either a format 2 or a format 3 label. The schematic form of the label is shown in Figure 2–12. A description of the fields within the label follows the figure.

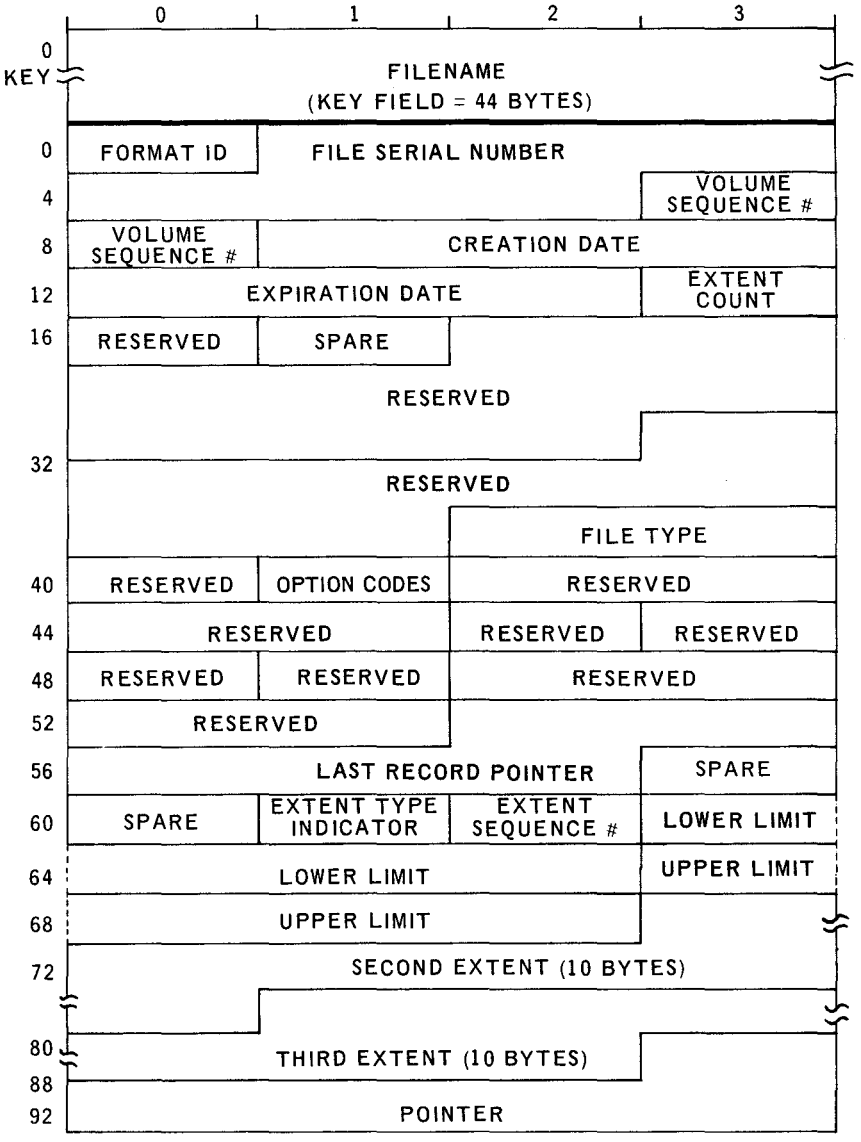


Figure 2–12. Disc Format 1 Label

STDEQU Label	Field	Code	
■ *	Key	EBCDIC	File Identifier – Each file must have a unique 1- to 44-byte name in this key field. A search of the Volume Table of Contents is made on this name.
■ DL$ID1	Byte 0	EBCDIC	Format Identifier – Always 1, for format 1 label.
■ DL$FS1	Bytes 1–6	EBCDIC	File Serial Number – Identifies the volume on which the file starts, is a six-digit alphanumeric number, and is the same as the volume serial number of the volume on which the file starts.

STDEQU label	Field	Code	
■ DL$VS1	Bytes 7–8	Binary	Volume Sequence Number – Indicates the number of this volume relative to the first volume in the file.
■ DL$CD1	Bytes 9–11	Discontinuous Binary	Creation Date – Format is ydd (year-day-day) where y is 0 to 99, and dd is 1 to 366.
■ DL$ED1	Bytes 12–14	Discontinuous Binary	Expiration Date – The date when the file may be deleted. Format is the same as the creation date.
■ DL$XC1	Byte 15	Binary	Extent Count – Specifies the number of extents currently constituting the file, or portions of it, on this volume.
■ DL$LD1	Byte 16	Binary	Reserved for Directory Block – Used only in partitioned files. Indicates the number of bytes used in the last directory block. Partitioned files are not processed by the Data Management System.
■ *	Byte 17		Spare.
■ *	Bytes 18–30	EBCDIC	Reserved for Programming System Identifier – May consist of any combination of digits (0–9), characters (A–Z), and blanks.
■ *	Bytes 31–37		Reserved.
■ DL$FT1	Bytes 38–39	Hexadecimal	File Type –

Hexadecimal Code	Meaning
4000	Sequential
2000	Direct Access
8000	Indexed Sequential
0200	Partitioned
0000	Undefined

■ DL$RF1 Byte 40 Binary Reserved for Record Format –

Bit	Content	Meaning
0,1	01	Variable-length records
	10	Fixed-length records
	11	Undefined format
2	0	No track overflow
	1	Uses track overflow
3	0	Unblocked records
	1	Blocked records

STDEQU label	Field	Code	
■ DL$RF1	Byte 40	Binary	Reserved for Record Format – (cont.)

Bit	Content	Meaning
4	0	No truncated records
	1	Truncated records in file
5,6	01	Control character ASCII code
	10	Control character machine code
	11	Control character undefined
7	0	Records have no keys
	1	Records have keys

■ DL$OC1 Byte 41 Binary — Option Codes –

Bit	Content	Meaning
0	1	File was created using write validity check
1	1	Allocation by cylinder
2	1	New file
3–7		Unused

■ DL$BL1 Bytes 42–43 Binary — Reserved for Block Length – Size of fixed-length blocks or maximum size of variable-length blocks.

■ DL$RL1 Bytes 44–45 Binary — Reserved for Record Length – Size of fixed-length records or maximum size of variable-length records.

■ DL$KY1 Byte 46 Binary — Reserved for Key Length.

■ DL$KL1 Bytes 47–48 Binary — Reserved for Key Location – High order position of key field within each data record.

■ DL$DS1 Byte 49 Binary — Reserved for Data Set Indicators –

Bit	Meaning (when bit = 1)
0	The last volume of this file
1	File is not relocatable
2	Block length is a multiple of eight bytes
3	File is security protected – password is necessary
4	File may not be read
5	File may not be read by more than one program
6	File may not be written into
7	File may not be written into by more than one program

STDEQU label	Field	Code	
■ DL$SA1	Bytes 50–53	Binary	Reserved for Secondary Allocation – The amount of storage to be requested at the end of an extent.
■ DL$LR1	Bytes 54–58	Discontinuous Binary	Last Record Pointer – Indicates the address of the last record written in a sequential or partitioned file. The format is ttrll, where tt is the relative track number, r is the record number, and ll is the number of bytes remaining on the track. These bytes are generated for an output volume by the FEOV macro instruction and are used to control the last record location on input.
■ *	Bytes 59–60		Spare.
■ DL$XT1	Byte 61	Hexadecimal	Extent Type Indicator –

Hexadecimal Code	Meaning
00	No valid extent described
01	Prime data area
02	Overflow area for indexed sequential file
04	High level index area of indexed sequential file
40	User label track area
8n	Reserved for shared cylinder indicator, where n = 1, 2, or 4

■ DL$XS1	Byte 62	Binary	Extent Sequence Number – Relative number of extent in multiple extent volume.
■ DL$XL1	Bytes 63–66	Discontinuous Binary	Lower Limit – The address specifying the start of the extent, in the form cchh.
■ DL$XU1	Bytes 67–70	Discontinuous Binary	Upper Limit – The address specifying the end of the extent, in the form cchh.
■ *	Bytes 71–80		Second Extent – Same format as described for bytes 61 through 70.
■ *	Bytes 81–90		Third Extent – Same format as second extent.
■ DL$CP1	Bytes 91–95	Discontinuous Binary	Continuation Pointer – The address of a continuation label, either a format 2 label for indexed sequential files or format 3 for extra extent descriptors. The address is in the form cchhr. Binary 0 if label does not exist.

*No STDEQU label applicable.

2.3.3. Disc Format 2 Labels

Format 2 labels are used only with indexed sequential files. For a particular file, this label appears in the VTOC of the first (or only) volume containing the file, and is addressed by the continuation pointer (bytes 92 through 95) of the format 1 label for the file. The schematic form of the label is shown in Figure 2–13. A description of the fields within the label follows the figure.

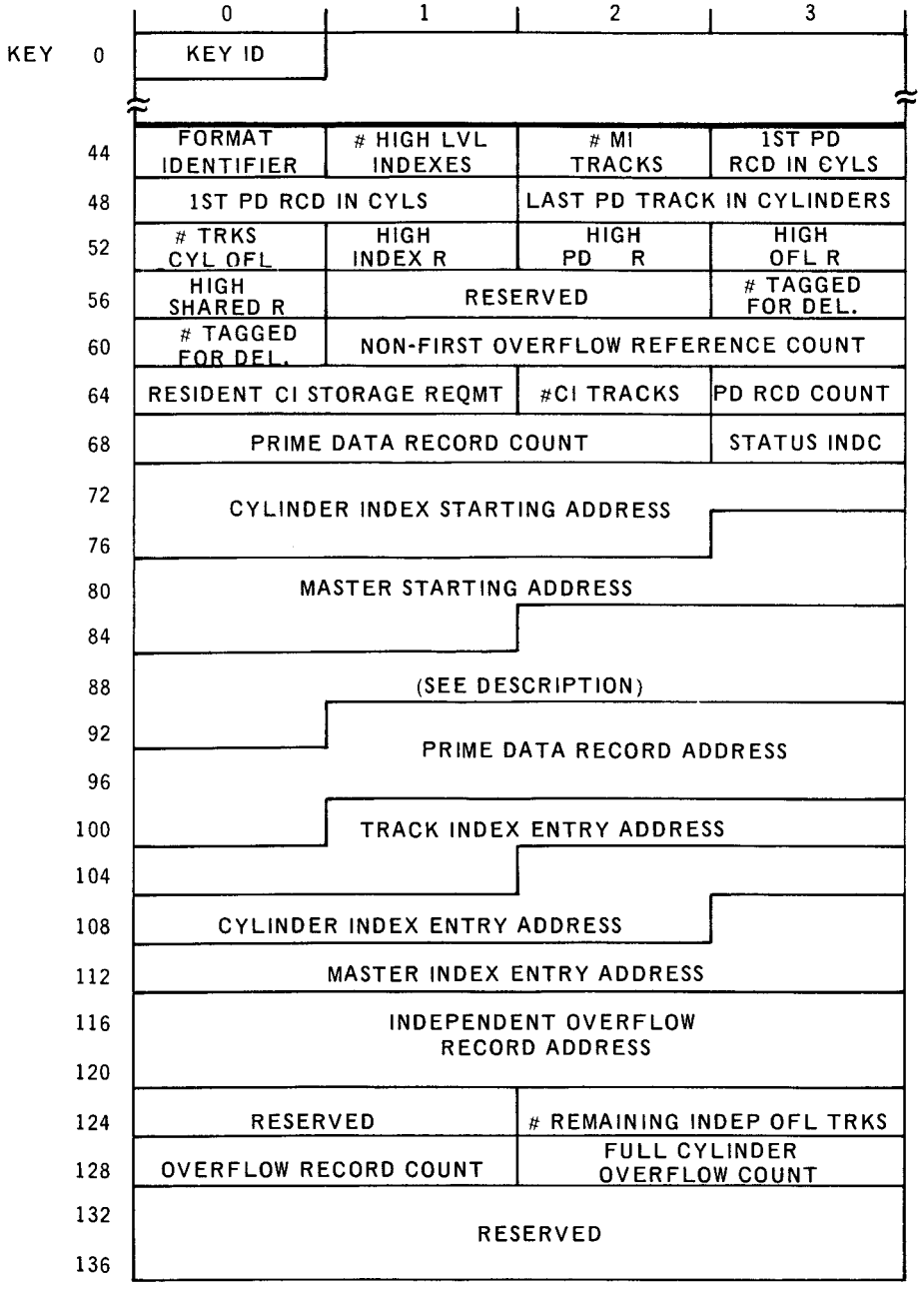


Figure 2–13. Disc Format 2 Label (Indexed Sequential Files)

STDEQU Label	Field	Code	
■ DL$ID2	Key		Key ID – Always X'02' – remaining 43 bytes are reserved.
■ DL$FI2	Byte 44	EBCDIC	Format Identifier – Always 2, for format 2 label.
■ DL$IL2	Byte 45	Binary	Number of Higher Level Indexes – Equals 2 if both master and cylinder indexes are present; equals 1 if cylinder index only.
■ DL$HL2	Byte 46	Binary	Number of Master Index Tracks.
■ DL$RC2	Bytes 47–49	Discontinuous Binary	First Prime Data Record in Cylinders – Value of hhr for first data record in each prime data cylinder. ←
■ DL$LC2	Bytes 50–51	Discontinuous Binary	Last Prime Data Track in Cylinders – Value of hh for last prime data track in each prime data cylinder. ←
■ DL$TO2	Byte 52	Binary	Number of Tracks for Cylinder Overflow.
■ DL$HH2	Byte 53	Binary	High Index Record – Value of r for last possible record on nonshared index tracks. ←
■ DL$HP2	Byte 54	Binary	High Prime Data Record – Value of r for last possible record on nonshared prime data tracks. ←
■ DL$HO2	Byte 55	Binary	High Overflow Record – Value of r for last possible record on overflow tracks. ←
■ DL$LR2	Byte 56	Binary	High Shared Prime Data Record – Value r for last possible record on shared prime data tracks. ←
■ *	Bytes 57–58		Reserved.
■ DL$TD2	Bytes 59–60	Binary	Number of Records Tagged for Deletion – Number of records in the file tagged for deletion by user. Corresponds to filenameT; refer to Table 4–7.
■ DL$NF2	Bytes 61–63	Binary	Nonfirst Overflow Reference Count – Number of times in random retrieval reference was made to an overflow record not the first in its chain. Corresponds to filenameR; refer to Table 4–7.
■ DL$HB2	Bytes 64–65	Binary	Resident Cylinder Index Storage Requirement – Number of bytes required to hold entire cylinder index in main storage. Corresponds to filenameS. (Refer to INDAREA keyword for DTFIS, and to Table 4–7.)

STDEQU Label	Field	Code	
■ DL$HT2	Byte 66	Binary	Number of Cylinder Index Tracks.
■ DL$PR2	Bytes 67–70	Binary	Prime Data Record Count – Total number of logical records in prime data areas of the file. Corresponds to filename P; refer to Table 4–7.
■ DL$SI2	Byte 71	Binary	Status Indicators

Bit	Meaning
0	Last prime data block written is full.
1	Last prime data track written is full.
2	Last prime data cylinder written is full.
3	Last cylinder-index track written is full.
4–6	Reserved.
7	First record indicator.

↓	■ DL$CI2	Bytes 72–78 Discontinuous Binary	Cylinder Index Starting Address – Value of mbbcchh, where m is the volume number, bb is two bytes of zeroes, cc is the cylinder number, and hh is the head number.
↑ →	■ DL$LM2	Bytes 79–85 Discontinuous Binary	Master Index Starting Address – Value of mbbcchh.
	■ DL$HM2	Bytes 86–92 Discontinuous Binary	Identical with preceding field (bytes 79–85); maintained for compatibility.
→	■ DL$LP2	Bytes 93–100 Discontinuous Binary	Prime Data Record Address – Value of mbbcchhr, which is the disc address to which the last prime data record (r) was written, or following which the next prime data record will be written.
→	■ DL$LT2	Bytes 101–105 Discontinuous Binary	Track Index Entry Address – Value of cchhr, which is the disc address to which the last track index entry was written, or following which the next track index entry will be written.
→	■ DL$LE2	Bytes 106–110 Discontinuous Binary	Cylinder Index Entry Address – Value of cchhr, which is the disc address to which the last cylinder index entry was written, or following which the next cylinder index entry will be written.

STDEQU label	Field	Code	
■ DL$MI2	Bytes 111–115	Discontinuous Binary	Master Index Entry Address – Value of cchhr, which is the disc address to which last master index entry was written, or following which the next master index entry will be written. ←
■ DL$LI2	Bytes 116–123	Discontinuous Binary	Independent Overflow Record Address – Value of mbbcchhr, which is the disc address to which last record in the independent overflow area was written, or following which the next record will be written. ←
■ DL$BR2	Bytes 124–125	Binary	Reserved for the number of bytes remaining on overflow tracks.
■ DL$IO2	Bytes 126–127	Binary	Remaining Independent Overflow Tracks – Number of tracks in the independent overflow area which can still receive overflow records. Corresponds to filenamel; refer to Table 4–7.
■ DL$OR2	Bytes 128–129	Binary	Overflow Record Count – Total number of logical records in overflow areas of the file. Corresponds to filenameO; refer to Table 4–7.
■ DL$CO2	Bytes 130–131	Binary	Full Cylinder Overflow Count – Total number of prime data cylinders containing full cylinder overflow areas. Corresponds to filenameA; refer to Table 4–7.
■ *	Bytes 132–134		Reserved.
■ DL$CP2	Bytes 135–139	Discontinuous Binary	Reserved for the pointer to a format 3 label. Binary 0 if label does not exist.

*No STDEQU label applicable.

2.3.4. Disc Format 3 Labels

A format 3 label is used when there are more than three extents in a file. Each format 3 label can describe up to 13 additional extents of a file. The last field of the label is a pointer which, if used, indicates the address of the next format 3 label. However, since a limit of 16 extents per volume are supported by the UNIVAC OS/4 Data Management System, this field is always 0. The schematic form of the format 3 label is shown in Figure 2–14. A description of the fields within the label follows the figure.

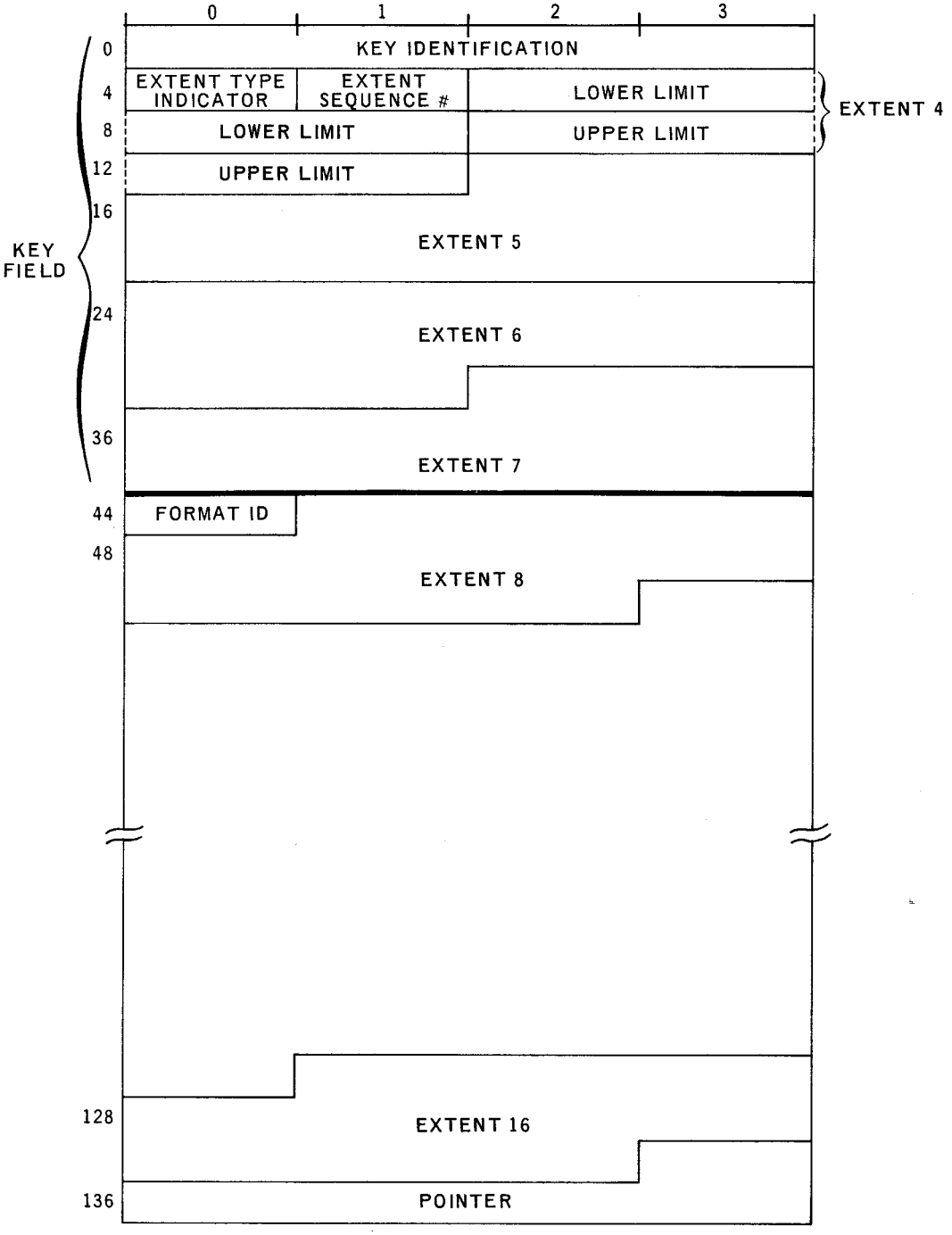


Figure 2–14. Disc Format 3 Label

STDEQU Label	Field	Code	
■ DL$ID3	Bytes 0–3		Key Identification – Each byte contains 03_{16}.
■ DL$XT3	Byte 4		Extent Type Indicator –
			Hexadecimal Code: 00 – No valid extent described; 01 – Prime data area
■ DL$SN3	Byte 5	Binary	Extent Sequence Number – relative number of extent in this volume of the file.
■ DL$XL3	Bytes 6–9	Discontinuous Binary	Lower Limit – Starting track address of the extent, in the form cchh. ←
■ DL$XU3	Bytes 10–13	Discontinuous Binary	Upper Limit – Terminating track address of the extent, in the form cchh. ←
■ *	Bytes 14–23		Extent 5 – Same format as described for bytes 4 through 13 for this extent.
■ *	Bytes 24–43		Extents 6 and 7.
■ DL$FI3	Byte 44	EBCDIC	Format Identifier – Always 3, for format 3 label.
■ DL$XS3	Bytes 45–134		Extents 8 through 16.
■ DL$CP3	Bytes 135–139	Discontinuous Binary	Pointer – Address of next format 3 label, in the form cchhr. Binary 0 if no further label. ←

*No STDEQU label applicable.

2.3.5. Disc Format 4 Labels

The format 4 label describes the volume table of contents itself and is the first record in the table of contents. An indicator in the format 4 label states whether a format 5 label contains valid information. The schematic form of the format 4 label is shown in Figure 2–15. A description of the fields within the table follows the figure.

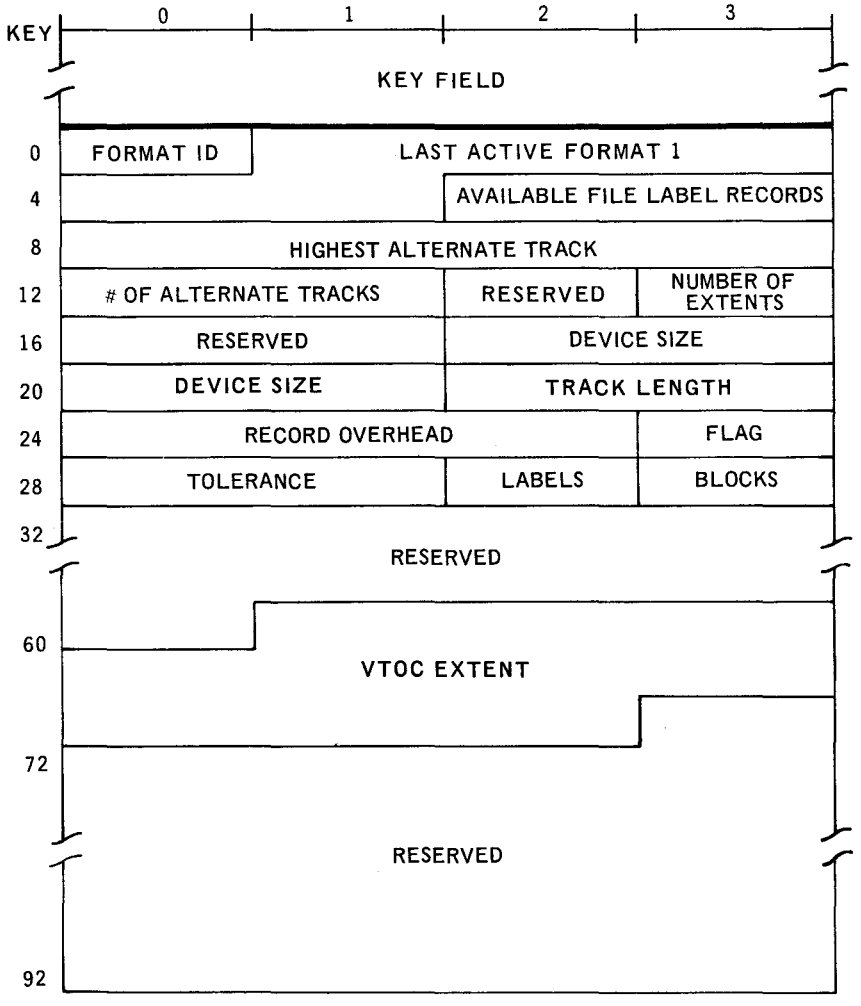


Figure 2–15. Disc Format 4 Label

STDEQU Label	Field	Code	
■ *	Key	Binary	Key Field – Each byte of this field contains 04_{16}.
■ DL$ID4	Byte 0	EBCDIC	Format ID – Always 4 for format 4 label.
■ DL$LF4	Bytes 1–5	Discontinuous Binary	Last Active Format 1 – The address, in the form cchhr, used for a search on filename. ←
■ DL$AF4	Bytes 6–7	Binary	Available File Label Records – Number of unused records in the volume table of contents.
■ DL$HA4	Bytes 8–11	Discontinuous Binary	Highest Alternate Track – Address, in the form cchh, of alternate tracks set aside in case of bad tracks. ←
■ DL$AT4	Bytes 12–13	Binary	Number of Alternate Tracks.
■ DL$VI4	Byte 14		Reserved for VTOC Indicators –

Bit	Contents	Meaning
0	1	A format 5 label, if present, contains invalid information
1–7	0	Unused

■ DL$XC4	Byte 15	Binary	Number of Extents – Contains 01_{16} to indicate the one extent in the volume table of contents.
■ *	Bytes 16–17		Reserved.
■ DL$DS4	Bytes 18–21		Device Size – Indicates the number of cylinders and the number of heads per cylinder on the device, in the form cchh. ←
■ DL$TL4	Bytes 22–23		Track Length – Number of available bytes on a track exclusive of home address and Record 0.
■ DL$RO4	Bytes 24–26		Record Overhead – ILK describes overhead bytes on track, where I is for keyed record which is not the last on track, L is for keyed record which is the last on track, and K is a decrement applied to records which have no key.
■ DL$FG4	Byte 27		Flag –

Bit	Meaning
0–5	Reserved
6,7	Device-dependent characteristics

■ DL$TO4	Bytes 28–29		Tolerance – A device-dependent factor which is used to calculate effective record lengths for that device.
■ DL$LT4	Byte 30		Labels per Track – A device-dependent factor specifying the number of 140-byte labels possible in a volume table of contents track.

■ DL$BT4	Byte 31	Blocks per Track – A device-dependent factor specifying the number of directory blocks of a partitioned file which can be written on a track.
■ *	Bytes 32–60	Reserved.
■ DL$VX4	Bytes 61–70	VTOC Extent – Describes the extent occupied by the VTOC itself. The format of this field is identical to the fields describing the extent in the format 1 and 3 labels.
	Bytes 71–95	Reserved.

*No STDEQU label applicable.

2.3.6. Disc Format 5 Labels

The format 5 label is the second record in the volume table of contents and is used to maintain control of the available extents in the volume at any time. If several format 5 labels are valid, they are linked to each other. The schematic form of the format 5 label is shown in Figure 2–16. A description of the fields within the table follows the figure.

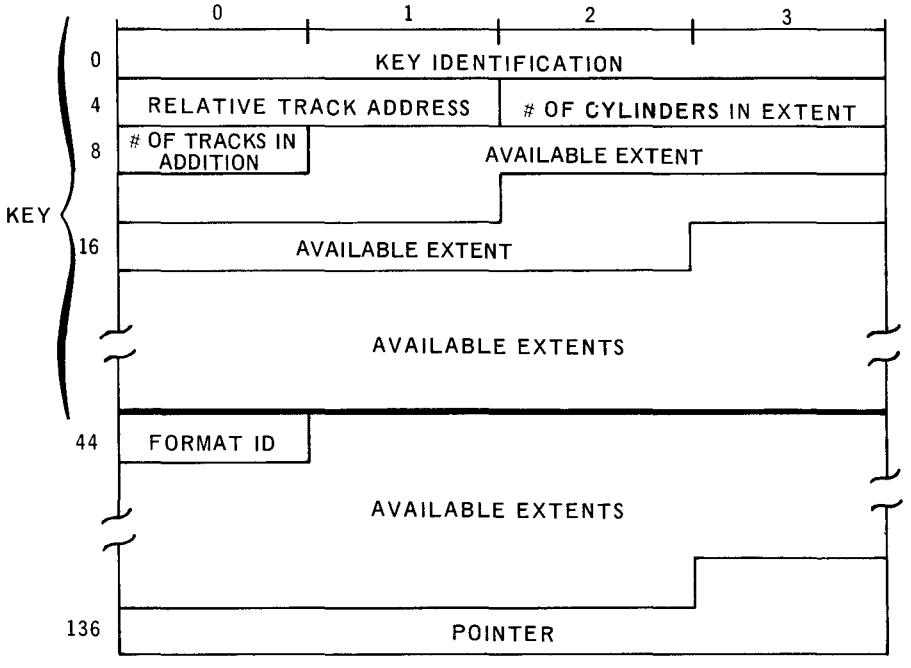


Figure 2–16. Disc Format 5 Label

STDEQU Label	Field	Code	
■ DL$ID5	Bytes 0–3		Key Identification – Each byte of this field contains 05_{16}.
■ DL$XT5	Bytes 4–5	Discontinuous Binary	Relative Track Address – Start of extent.
■ DL$XC5	Bytes 6–7	Binary	Number of Cylinders in Extent.
■ DL$XE5	Byte 8	Binary	Number of tracks in extent in addition to the cylinders.
■ *	Bytes 9–13		Available Extent – Describes another extent in fields with the same format as bytes 4 through 8 above.
■ *	Bytes 14–43		Six more available extents.
■ DL$FI5	Byte 44	EBCDIC	Format ID – Always 5, for format 5 label.
■ DL$XS5	Bytes 45–134		Eighteen more available extents.
■ DL$CP5	Bytes 135–139	Discontinuous Binary	Pointer – Indicates the address of another format 5 label, in the form cchhr. Binary 0 if no further label.

*No STDEQU label applicable

2.3.7. Disc User Header/Trailer Labels

Each volume of a file may have up to eight user header and eight user trailer labels. A four-byte key precedes each label. For header labels it contains UHL1 through UHL8; for trailer labels it contains UTL0 through UTL7. A schematic form of the label is shown in Figure 2–17.

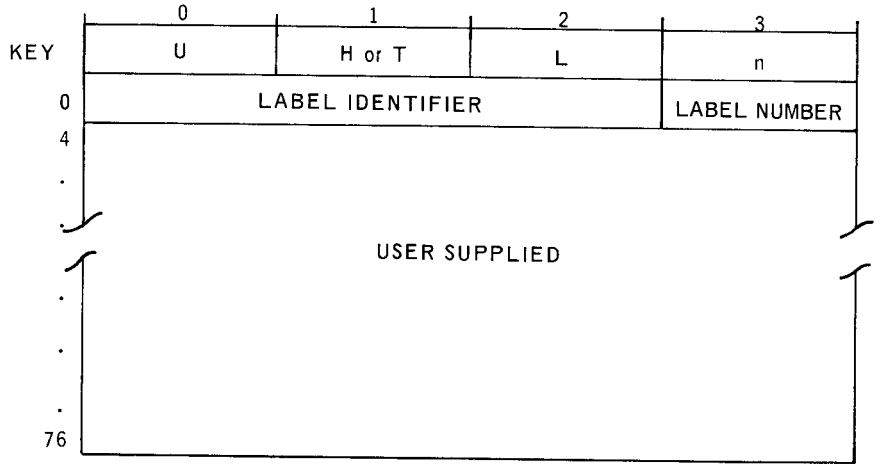


Figure 2–17. Disc User Header/Trailer Label

■ Key	The key contains UHLn for header labels or UTLn for trailer labels. The range of values for n may be 1 through 8 for header labels and 0 through 7 for trailer labels.
■ Bytes 0–2	Label Identifier – For disc header labels these bytes contain UHL; for trailer labels, UTL.
■ Byte 3	Label Number – The number of a user disc header or trailer label ranges from 1 to 8.
■ Bytes 4–79	User supplied.

2.3.8. Track Organization for User Header/Trailer Labels

Standard user labels may be processed by both the sequential and direct access methods. The sequential access method processes both header and trailer labels; the direct access method processes header labels only.

The user labels must be in the first track of the first extent for each volume of the file. There may be a maximum of eight header labels and eight trailer labels for each volume of a file.

The labels are each 80 bytes long and the four bytes preceding the label contain a key. For header labels, the key is UHL1...UHL8. The key for trailer labels is UTL0...UTL7.

A diagram of the relational position of the header and trailer labels on a track for a file accessed by the sequential access method and a file accessed by the direct access method is shown in Figures 2–18 and 2–19. An explanation of the user labels follows each figure.

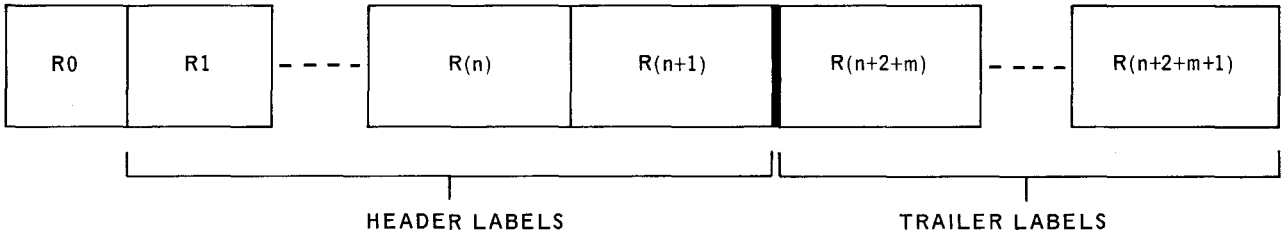


Figure 2–18. Sequential Access Method Track User Label Format

The range of values for n is 1 through 8 and for m is 0 through 7. An explanation of the content of each record follows.

- R0 — Record 0 has no key. Both the length of the data and the contents are standard.
- R1 — The key is UHL1. The length of the label is 80 bytes. The first four bytes of the label are UHL1; the remaining 76 bytes are user supplied.

  .
  .
  .
- R(n) — The key is UHL(n). The length of the label is 80 bytes. The first four bytes of the label are UHL(n); the remaining 76 bytes are user supplied.
- R(n+1) — The key is UHL(n+1). The length of the label is 0. This is a file mark recognized by logical IOCS.
- R(n+2+m) — The key is UTL(m). The length of the label is 80 bytes. The first four bytes of the label are UTL(m+1); the remaining 76 bytes are user supplied.

  .
  .
  .
- R(n+2+m+1) — The key is UTL(m+1). The length of the label is 0. This is a file mark recognized by logical IOCS.

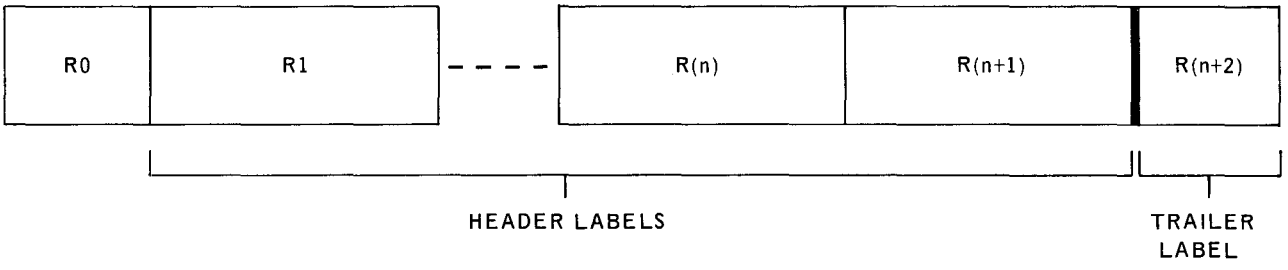


Figure 2–19. Direct Access Method Track User Label Format

The range of values for n is 1 through 8. An explanation of the content of the records follows.

- R0 — Record 0 has no key. Both the length of the data and the contents are standard.
- R1 — The key is UHL1. The length of the label is 80 bytes. The first four bytes of the label are UHL1; the remaining 76 bytes are user specified.

  .
  .
  .
- R(n) — The key is UHL(n). The length of the label is 80 bytes. The first four bytes of the label are UHL(n); the remaining 76 bytes are user specified.
- R(n+1) — The key is UHL(n+1). The length of the label is 0. This is a file mark recognized by logical IOCS.
- R(n+2) — The key is UTL0. The length of the label is 0. This is a file mark recognized by logical IOCS.

2.3.9. Record Types

There are two main record types on direct access storage devices: with a key field or without a key field. These are illustrated in Figure 2–20.

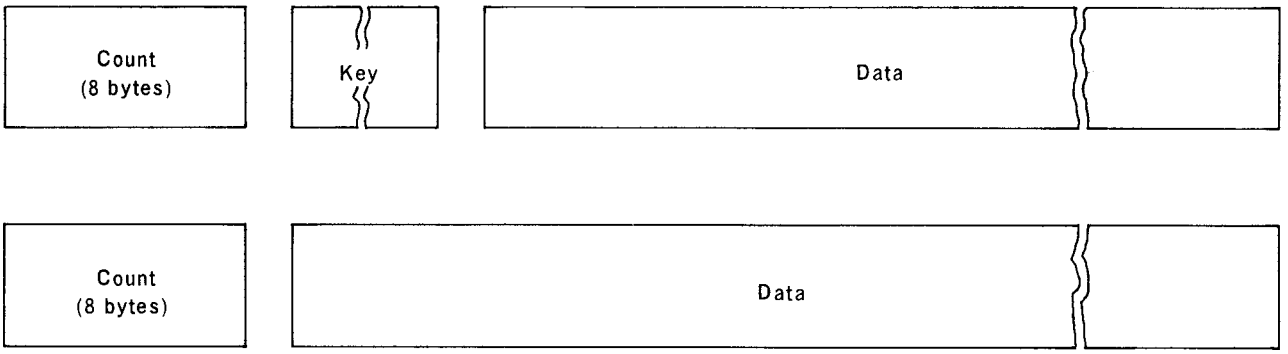


Figure 2–20. Disc Record Types

The count field is required by the hardware and is illustrated in detail in Figure 2–21. This field is used to describe the physical location of the record and the length of the key and data fields. The size of the count field is always 11 bytes, although only 8 bytes are transmitted into and out of the processor. The remaining 3 bytes are generated by the control unit and are not available to the user. Allowance must be made for the count field (8 bytes) when specifying the size of output areas as described in later sections.

The key field is optional and is used only when the data is located by means of a search on a key. The key field can vary in size from 3 to 255 bytes. However, within one file, all keys must be the same length.

The data field may be any length up to a maximum of 3625 bytes for the UNIVAC 8411 Disc Subsystem or 7294 bytes for the UNIVAC 8414 Disc Subsystem, and the UNIVAC 8424 Disc Subsystem.

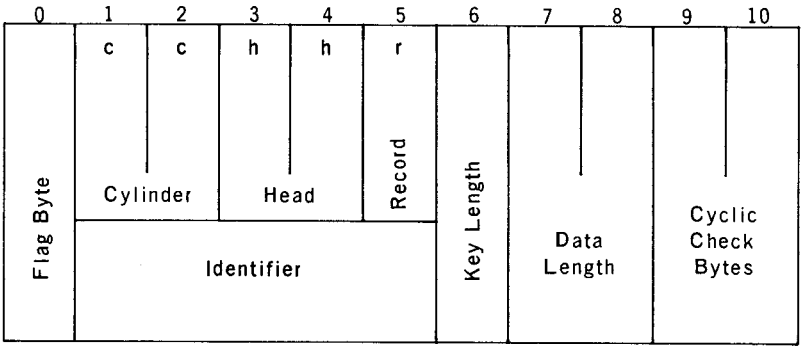


Figure 2–21. Count Area on Disc

- Flag Byte

  This byte is automatically supplied by the control unit as records are written. However, the byte is not available to the user. The significance of the bits in the flag byte is as follows:

Bit	Function
0	0 = Even numbered records 1 = Odd numbered records
1	0 = Nonoverflow (normal) records 1 = All segments, except the last, of overflow records
2–5	Always 0
6	0 = Operative track 1 = Defective track
7	0 = Primary track 1 = Alternate track

- Identifier

  The identifier is normally the address of the count area itself and is composed of the cylinder, head, and record numbers.

  – Cylinder Number

    These bytes indicate the cylinder number of the track on which the data is stored.

  – Head Number

    These bytes indicate the read/write head number of the track on which the data is stored.

  – Record Number

    This byte indicates the sequential number of this record on the track. A maximum value of 255 is allowed by this eight-bit binary number.

- Key Length

  This field specifies the number of bytes in the key area of the record. Lengths from 3 to 255 bytes are possible, or 0 if no key length specified.

- Data Length

  This field specifies the number of bytes in the data area of the record. A data length of 0 causes one byte of binary 0's and two check bytes to be written. On reading, no data is transferred to the I/O channel and an end-of-file mark is indicated.

- Cyclic Check

  These bytes permit error checking of the information read from the count area, but the bytes are not available to the user.

2.3.9.1. Sequential Disc Files

Records may be grouped into blocks when more than one logical record equals a physical record. Records also may be unblocked when only one logical record equals a physical record. The format of individual records may be fixed length, variable length, or undefined. If the record format is undefined, the records are assumed to be unblocked and any deblocking must be supplied by the user. Figures 2–22 and 2–23 illustrate the formats of fixed-length and variable-length records. Comparisons may be made with Sections 2.2.4.1 and 2.2.4.2 which describe tape record formats.

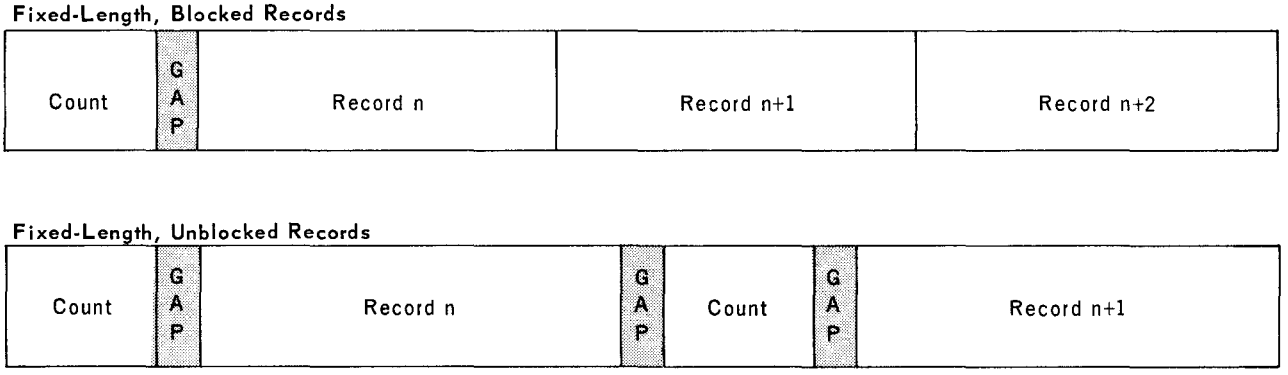
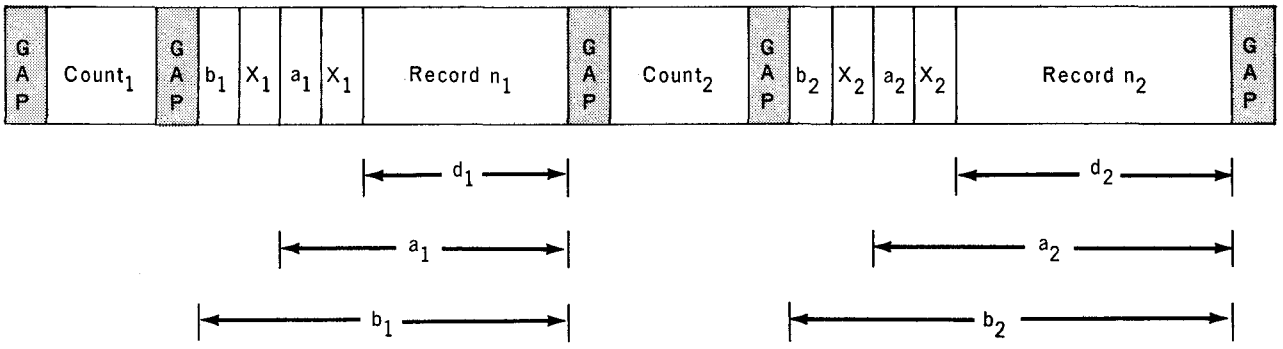


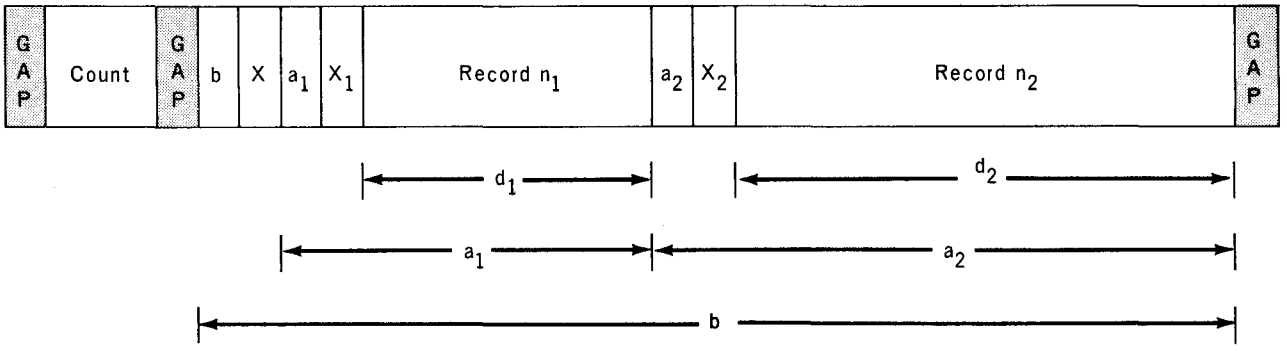
Figure 2–22. Disc Fixed-Length Records – Sequential Files

Files with variable-length records may have either blocked or unblocked records. The length of the block must be specified and must precede the record(s) in the block. The number of bytes in a variable-length record must be specified and must precede the data in the record. Figure 2–23 illustrates the format of both blocked and unblocked variable-length records.

Variable-Length, Unblocked Records



Variable-Length, Blocked Records



LEGEND:

b – Block length (2 bytes)
a – Record length (2 bytes)
X – Reserved for system use (2 bytes)
d – Data field in the record
GAP – Disc (hardware) gap

Figure 2–23. Disc Variable-Length Records – Sequential Files

2.4. CARD AND PRINTER CONVENTIONS

Data Management utilizes certain card and printer conventions established by the UNIVAC OS/4 Operating System in processing card files.

2.4.1. Start-Of-Data ($ Job Control Statement)

Data Management does not check for a start-of-data card. If the user is filing data as a Job Control option, the start-of-data (/$) card must be included in the job stream. For consistency the user may choose the /$ card convention as a card file identification. In this case the problem program should include a check for this card.

2.4.2. End-Of-Data (* Job Control Statement)

Data Management checks for an end-of-data card when the user is reading EBCDIC cards. The format of this card is identical to that required by Job Control. The first two columns contain /*. When this configuration is sensed, control is transferred to the end-of-file address specified for the file. When an output file is punched, the end-of-data card is not punched.

2.4.3. Printer/Punch – First Character Control

The user has the option of specifying in the DTFPR or DTFCD macro instruction that a control character is to be included in the data records. This character specifies line spacing or skipping when the file is printed or stacker selection when the file is punched. The character itself is never printed or punched but is a part of the record in storage. If the record is sent to a device, such as the disc, that does not recognize this control character, the system assumes that the control character is data. If the record is sent to a punch or a printer and the user has not specified that the record contains a control character, the character is handled as data. I/O areas must be large enough to include this character.

When fixed-length or undefined record formats are used, the control character is the first character in the record (see Figure 2–24 and Figure 2–25). It is the first character following the record length specification in a variable-length unblocked record format (see Figure 2–26). The block size of the output area must include the byte for the control character. If variable-length, unblocked records are to be processed, the block size must account for the initial eight characters as well as the control character (nine bytes total) in the output area. Although these characters do not appear in the output, the output area must be large enough to accommodate them. It must also be noted that when a control character is specified, every record must contain a control character.

When a PUT macro instruction is executed, the control character (for the character required, see the CTLCHR keyword parameter under the appropriate DTF macro instruction) in the data record becomes the command code (one byte) of the Channel Command Word (CCW). The first character after the control character in the output data is the first character printed or punched.

If the command parameter within a CNTRL macro instruction is to be issued for the punch or printer, the CNTRL keyword parameter is specified and CTLCHR must be omitted. First character control must not be specified when the read/punch feature of the UNIVAC 0604 Card Punch Subsystem is used for combined files.

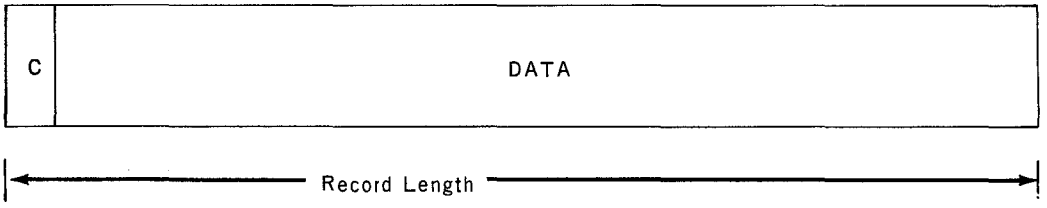


Figure 2–24. Fixed-Length Record Format

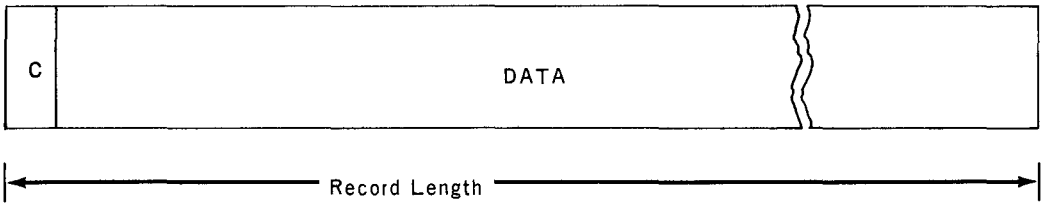
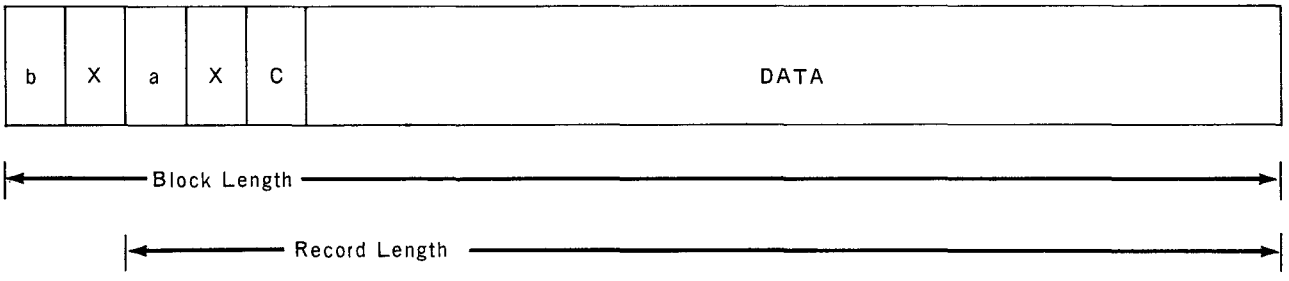


Figure 2–25. Undefined Record Format



LEGEND:

b – Block length (two bytes)
a – Record length (two bytes)
X – Reserved for system use (two bytes)
C – Control character (one byte)

Figure 2–26. Variable-Length, Unblocked Record Format

3. SEQUENTIAL FILE PROCESSING

3.1. GENERAL

All files with records that follow one another in a serial or physically adjacent manner are processed by a sequential access method. Devices which may have sequential files include magnetic tape units, card readers, printers, punches, and direct access storage units (discs). Each file for each device must be defined by a declarative macro instruction which generates the necessary constants.

The problem program handles and processes records, blocks, and files by issuing imperative macro instructions which are expanded into executable code in-line with the user's instructions.

This section describes the declarative macro instructions for each device and the imperative macro instructions for processing the files.

3.2. DECLARATIVE MACRO INSTRUCTIONS

A problem program must include the following declarative macro instructions to define the files for any devices which may be accessed in a sequential manner.

DTFMT	—	**Magnetic Tape**
DTFSD	—	**Direct Access Storage Device**
DTFCD	—	**Card Device**
DTFPR	—	**Printer**
DTFOR	—	**Optical Document Reader**
DTFPT	—	**Paper Tape**

The symbolic name for the file has a maximum of seven characters and must begin with an alphabetic character. The keyword parameters may appear in any order and must be separated by commas.

3.2.1. Magnetic Tape

The declarative macro instruction, DTFMT, is required for both input and output magnetic tape files. Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFMT macro instruction.

A description of each keyword parameter follows the listing. A summary of the keyword parameter is given in Table 3–1 following the descriptions.

LABEL	OPERATION	OPERAND
filename	DTFMT	BKSZ=n EOFA=symbol (Input Files Only) IOA1=symbol [BKNO=YES] [CKPT=YES] [CLRW={RWD / NORWD}] [ERRO={IGNORE / SKIP / symbol}] [ERROR=symbol] [FLBL={STD / NSTD / NO}] [IOA2=symbol] [IORG=(r)] [LBAD=symbol] [OPTION=YES] [OPRW=NORWD] [RCFM={FIXUNB / FIXBLK / VARUNB / VARBLK / UNDEF}] [RCSZ={n / (r)}] [READ={FORWARD / BACK}] [TPMK=NO] [TYPE={INPUT / OUTPUT / INOUT}] [VBLD=(r)] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- Block Numbering

The user may specify whether tape blocks are to be numbered by the VOL statement card in the control stream or by the following keyword parameter.

BKNO=YES

When this keyword parameter is specified, logical IOCS checks input tapes for block numbering or writes binary block numbers on output tapes. Block numbers begin with 1 and are incremented sequentially by one. All label, data, and checkpoint blocks are numbered. Tape marks are counted but cannot have a number as part of the tape mark.

Block numbers are written as part of the block, and require an additional three bytes which are the first three bytes in the block. These additional bytes are not reflected in the value of the block size parameter BKSZ, but they must be considered when the block size approaches the maximum/minimum hardware physical block size.

When this keyword parameter is specified, a four-byte storage area aligned on a fullword boundary must immediately precede the area defined by IOA1 (and IOA2, if used). The I/O area(s) must also be located on a fullword boundary. The four-byte storage area is not included in the definition of IOA1, IOA2, or BKSZ.

Following is an example of fullword alignment and storage allocation for block numbering.

LABEL	OPERATION	OPERAND
	CNOP	0,4 fullword alignment
	DC	XL4'0' four-byte binary zeros
INPUT1	DS	dCLnn I/O area 1
	CNOP	0,4
	DC	XL4'0' } I/O area 2, if used
INPUT2	DS	dCLnn

NOTE: BKSZ = d(nn) of the define storage (DS)

If IOA2 is specified, the same conventions regarding block numbering are applicable.

On a forward read with block numbering, logical IOCS reads into IOA1–3 a block of length BKSZ +3 and checks the three-byte block number.

On a backward read with block numbering, logical IOCS reads into IOA1 + (BKSZ–1) a block of size BKSZ+4. The three-byte area preceding the actual block is checked to ensure a correct block number. The additional (fourth) byte is used to ensure that the length of IOA1+3 is large enough to contain the data and block number for checking without destroying other data. If the block on the tape exceeds the specified BKSZ+3, the use of the fourth byte indicates this fact and an error condition is generated. On a backward read without block numbering, logical IOCS reads into IOA1 + (BKSZ–1) a block of size BKSZ+1.

On a write with block numbering, logical IOCS writes from IOA1–3 a block of length BKSZ+3. Logical IOCS inserts the current block number in the least significant three bytes of the reserved four-byte field immediately proceding IOA1 before writing.

On a backward read with nonstandard labels and no LBAD routine specified, any user labels may cause a BKNO error, if block number checking is desired.

When writing tape marks, the block number is updated but is not written. When reading a tape mark, the block number is updated only, since no checking can be performed.

The block number is never addressed as a data record, nor is it passed as part of a user's label. The block numbering option can be used with 9-track tapes or 7-track tapes if data conversion and no translation is specified.

- Block Size

  This keyword parameter is required for both input and output files. The format is:

  BKSZ=n

  where n (decimal) is the length or size of the block in bytes. For all record formats the minimum block size allowed is 18 bytes. For variable-length blocks, n is equal to the maximum size for blocks in this file and must include the block and record length fields. However, BKSZ must not include the additional block numbering bytes, or the extra byte required for a backward read operation; that is, when BKNO=YES and READ=BACK are specified.

  The maximum size of a block on magnetic tape is 4095 bytes on the UNISERVO VI-C, 32,767 bytes on the UNISERVO 12, and 32,767 bytes on the UNISERVO 16.

  All of the minimum and maximum block sizes, however, will be affected whenever block numbering (BKNO=YES) and read backward (READ=BACK) processing are specified.

- Bypass Checkpoint Dumps

  If an input file has checkpoint blocks interspersed with data, the checkpoint records are bypassed if the following keyword parameter is specified:

  CKPT=YES

- Rewind at CLOSE

  A tape may or may not be rewound when a file is closed. If this parameter is omitted, the tape is rewound with interlock when a file is closed.

  – CLRW=RWD

    This keyword parameter is used if a tape is to be rewound without interlock when a file is closed.

  – CLRW=NORWD

    This keyword parameter is used if a tape is not to be rewound when a file is closed.

- End of an Input File

  The address to which control is transferred when the tape mark following an end of data is sensed must be supplied by the following keyword parameter:

  EOFA=symbol

  where symbol (label) is the address of the routine which performs end-of-data processing. This keyword parameter is required for all input files.

- Unique File Errors

  A unique error for tape is a parity error from which physical IOCS has tried unsuccessfully to recover. If a unique error occurs when reading a block of an input tape, the user has the following options. He may process the block as if no error had occurred, bypass the block, enter a special routine for individually processing error blocks or terminate the job. The absence of this parameter specification will cause logical IOCS to assume that a job should be terminated if a unique error occurs. The other options are specified as follows:

– ERRO=IGNORE

This option permits the images in the I/O area to be available to the user as though no error had occurred.

– ERRO=SKIP

This option causes the block in error to be bypassed and the next error-free block to be accessed.

– ERRO=symbol

For this option, symbol (label) is the address of a user routine to process unique errors.

User tape files that specify the ERRO address option should note the following when setting up their routine at the ERRO address:

User must not change the present register save area.

Register 13 must be maintained after processing.

Data Management macros on the file should not be issued.

The ERRO record should be accessed through the address passed in register 1; address is start of data block. If PIOCS is used, another save area must be supplied.

Return to Data Management modules must be by way of address in register 14.

Data Management assumes the user return is set to skip the error. If he wishes to ignore the error and process the block, he must turn off the skip switch in the DTF (NI DT$FG2,X'EF'), and turn on the ignore switch in the DTF (OI DT$FG2,X'20'); see 1.5.

Returning to the Data Management modules with neither (or both) of the skip or ignore switches set takes the user to the ERROR address, if any, or cancels his job.

■ Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the ERROR address from a processing macro instruction (such as GET or PUT), the address of the instruction following the macro call is found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN, CLOSE, and FEOV). Register 0 contains the following information concerning the reasons for the error:

- Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary zeros.

- Byte 2 contains a one-character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.

- Byte 3 contains a one-character EBCDIC value indicating the reason for the error; refer to Appendix E.

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs.

Further access of the file is not permitted after a major file error has occurred.

■ Tape Labels

The type of labels on a tape must be specified. They may be standard, nonstandard, or undefined. The formats of the keyword parameter for the three types of labels are:

- FLBL=STD

  This is the keyword parameter to use for tapes with standard labels.

- FLBL=NSTD

  This is the keyword parameter to use for tapes with nonstandard labels. The user must provide a routine to create the labels for output files or to check the labels on input files. The address of the routine to handle user file labels is supplied by the keyword parameter LBAD. If nonstandard labels are present on input files but are not to be checked, LBAD is omitted. If the input file labels are not checked, a tape mark must precede the data.

- FLBL=NO

  This is the keyword parameter to use if the tape is unlabeled.

■ Input/Output Area

Each input or output file must have an area reserved for its individual use. The area must be defined by the following keyword parameter:

IOA1=symbol

where symbol (label) is the address of the I/O area. For variable-length records or blocks, the I/O area for the file includes the bytes of block and record length information. For backward or forward processing, if a block numbering option is desired (see BKNO=YES or the VOL Job Control statement), four bytes for storing the block number must be reserved immediately preceding the labeled I/O area. If read backward processing is specified, and no block numbering is desired, one byte must be reserved preceding the I/O area for validating block length. Logical IOCS handles all modifications necessary to write block numbers for output files and to check block numbers on input files.

NOTE: If standard labels are specified with user labels present and an LBAD routine is also specified, the I/O area must be at least 80 bytes in length to accommodate the user label.

- Secondary Input/Output Area

A secondary I/O area for standby processing improves efficiency by overlapping I/O and record processing time. The format of the keyword parameter for a secondary I/O area is:

IOA2=symbol

where symbol (label) is the address of the secondary I/O area. It is subject to the same requirements as noted in IOA1.

- Current Record Pointer

When a general register is used as an index register to reference current data, the following keyword parameter should be specified:

IORG=(r)

where (r) is the number of the general register (2–12). If a work area is not required, this keyword parameter must be specified when there are two I/O areas, or records are blocked. When read backward processing is desired and a work area is not specified, this parameter is also needed for variable length or undefined records.

For input files, logical IOCS places the address of the next available record in the specified register.

For output files, logical IOCS places the address of the next available I/O output area in the specified register.

- Special Label Handling

This keyword parameter supplies the address of a user routine which processes user header or trailer labels, either standard labels or nonstandard labels. The format is:

LBAD=symbol

where symbol (label) is the address of the user routine. When used, the LBAD routine may not retain control for more than 500 milliseconds; otherwise a TIME LIMIT error will result.

– Standard Labels

The special label handling routine must be capable of processing both header and trailer labels for files with standard labels. During the processing of the OPEN transient routine, the user receives control after the HDR1 header label has been checked on an input file or after the HDR1 has been written on an output file. For read backward processing, the user receives control after the bypassing of the first end-of-reel tape mark. He is then positioned to read the user's trailer labels. In any case, the user cannot access the HDR1 or EOF1 label. For both input and output files, logical IOCS places the EBCDIC character O, denoting OPEN, in the low order byte of register 0 and places the address of the I/O area in register 1. The input label pointed to by the address in register 1 is available for processing; labels for output files can be built in an I/O area which is pointed to by register 1. The special label handling routine must return control to logical IOCS by means of the LBRET macro instruction with the proper option specified (see 3.3.7). Logical IOCS completes the processing and prepares the file for the first GET or PUT macro instruction.

On input, after an end-of-volume (EOV1) label has been processed, for each EOV2-8 or UTL read, logical IOCS places an EBCDIC character V in register 0, loads register 1 with the I/O address of the label, and transfers control to the user's LBAD routine. All labels are processed or bypassed until a tapemark is encountered. At this time logical IOCS executes the end-of-volume procedures to close the current volume and open the next.

On input, after the end-of-file (EOF1) label has been processed, logical IOCS transfers control to the user's end-of-file (EOFA) routine. The user is expected to issue a CLOSE macro instruction, following which, if LBAD is specified, and for each EOF2-8 or UTL label read, logical IOCS places an EBCDIC character F in register 0, loads register 1 with the I/O address of the label, and transfers control to the user's LBAD routine. All labels are processed or bypassed until a tapemark is encountered. At this time the resetting of the DTF and CLOSE rewind procedures take place.

On output, after an end-of-volume (end tape) condition is encountered, the end-of-data tapemark and EOV1 label are written. Then logical IOCS places an EBCDIC character V in register 0, loads register 1 with the designated I/O area address in which the label is to be placed, and transfers control to the user's LBAD routine, if specified. All user trailer labels will be written until a LBRET 1 macro instruction is issued. At this time two tapemarks are written, the tape is rewound with interlock, a volume swap occurs, and the next output volume is opened.

On output, after an end-of-file (CLOSE) condition is encountered, the end-of-data tapemark and EOF1 label are written. Then logical IOCS places an EBCDIC character F in register 0, loads register 1 with the designated I/O area address in which the label is to be placed, and transfers control to the user's LBAD routine. All user labels will be written until a LBRET 1 macro instruction is issued, at which time two tapemarks are written and the CLOSE rewind options take place.

For read backward processing, end of volume and end of file are synonymous. This condition is assumed when logical IOCS senses the tapemark between the data blocks and the header labels. Logical IOCS loads the EBCDIC character F into register 0, places the address of the I/O area in register 1, and transfers control to the user's LBAD routine, if specified. Once any UHL's or HDR2-8 labels are processed, all further labels are bypassed until either a tapemark or load point is encountered. Logical IOCS then transfers control to the user's end-of-file routine (EOFA) in which the user issues a CLOSE macro instruction for the file.

- Nonstandard Labels

The user must process all labels by a special label handling routine when files have nonstandard labels.

On input, when such a tape file is concerned, logical IOCS executes the user's rewind option (OPRW) places an EBCDIC character O in register 0, and transfers control to the special label handling routine. If LBAD is not specified for the input file, logical IOCS positions the tape to the first record following the tape mark. If the tape mark is not present, logical IOCS cannot distinguish between labels and data. Therefore either LBAD must be specified or a tape mark must precede the data. When the special label handling routine finishes processing labels, it returns control to logical IOCS by means of the LBRET macro instruction.

Before control is transferred to the special label handling routine when logical IOCS senses the tape mark at the end of the data, an EBCDIC character E is placed in register 0, while the address of the I/O area containing the label is placed in register 1. After processing label(s), the routine returns control to logical IOCS by means of the LBRET macro instruction with the EBCDIC characters either EF or EV right justified in register 0.

When the EF characters are stored (or LBAD is not specified), logical IOCS branches to the user's end-of-file address as specified in the EOFA keyword parameter. When the EV characters are stored, logical IOCS executes the end-of-volume procedures to close the current volume and opens the next.

For read backward processing, logical IOCS assumes the tape is positioned properly for OPEN, and bypasses the tape mark, and then reads the last user trailer label. Control is then transferred to the user's LBAD routine, if specified, with register 0 containing the EBCDIC character O and register 1 containing the address of the label area. Once any user trailer labels are processed, the tape is positioned beyond the tape mark which follows the data. For each user trailer label processed by the user's LBAD routine, the user must return to logical IOCS by way of the LBRET macro instruction.

When the tape mark preceding the data is encountered, logical IOCS places an EBCDIC character E in register 0, places the address of the label area in register 1, and transfers control to the user's LBAD routine, if specified. For each user header label processed by the user's LBAD routine, the user must return to logical IOCS by means of the LBRET macro instruction (see 3.3.7), with register 0 containing the EBCDIC characters EF, right justified.

This cycle continues until all labels are either processed or bypassed and either a tape mark or load point is encountered. Control is then transferred to the user's end-of-file routine (EOFA) in which the user should issue a CLOSE macro instruction for the file.

On output, logical IOCS enables the user to write the header (or trailer) labels by loading register 0 with an EBCDIC O (or V or F), loading register 1 with the address of the I/O area in which the label is to be placed, and then transferring control to the user's LBAD routine. For each label to be written, the user must return by way of a LBRET 2 macro instruction, with register 0 containing the length of the label.

Refer to Appendix A.1 for descriptions of tape label processing in the Job Control/Logical IOCS interface.

- Optional Input File

The keyword parameter to allow the user to specify an optional input file follows:

OPTION=YES

When this keyword parameter is specified and the file has not been allocated to a device by Job Control, the OPEN transient routine ensures that logical IOCS will transfer control to the address of the user's end-of-file routine (EOFA) following the execution of the first GET macro instruction. This procedure maintains continuity in the user's program. The user is required to CLOSE the optional file.

When this keyword parameter is not specified and the file has not been allocated by Job Control, it is impossible to obtain records from the file, if one exists. Logical IOCS transfers control to the address of the user's error routine (ERROR). If a user error routine is not supplied, the job step is aborted.

- Rewind at OPEN

If the reels of a file are not to be rewound before labels are checked during the processing of the OPEN macro instruction, the following keyword parameter should be specified.

OPRW=NORWD

Otherwise reels are rewound at OPEN time. When read backward processing is specified, NORWD is assumed.

■ Record Format

The record format of a file is specified by a keyword parameter. There are five options to indicate whether the records are fixed length, variable length, blocked, unblocked, or undefined. The options are coded as follows:

- RCFM=FIXUNB Fixed-length, unblocked
- RCFM=FIXBLK Fixed-length, blocked
- RCFM=VARUNB Variable-length, unblocked
- RCFM=VARBLK Variable-length, blocked
- RCFM=UNDEF Undefined

Logical IOCS assumes a fixed-length, unblocked record format when this keyword parameter is not specified.

■ Record Size

The number of bytes in a record is made available to logical IOCS by this keyword parameter. Logical IOCS assumes that the record size is equal to the block size for fixed-length, unblocked records and that the size of the record is available in the first two bytes of each variable-length record.

- RCSZ=n

  This form of the keyword parameter should be used for fixed-length, blocked records, where n is the number of bytes in the record.

- RCSZ=(r)

  This form of the keyword parameter is required for output files with an undefined record format. The general register (2–12) specified by (r) contains the block length. The keyword parameter may also be used for input files with an undefined record format. Logical IOCS places the block length into the general register specified by (r).

■ Input File Direction

Logical IOCS assumes that input files are to be read in a forward direction. Input files may also be read in a backward direction.

- READ=FORWARD

  This form of the keyword parameter is specified, if an input file is to be read forward.

- READ=BACK

This form of the keyword parameter is required, if an input file is to be read backwards. A byte must then be allocated before each I/O area, as described in the block numbering section. Also, the user is limited to handling one volume only during backward processing.

- Tape Mark Following Header Labels

  For output files with nonstandard labels or no labels, a tape mark usually separates the labels and the data. If, however, the user wants to eliminate the use of the tape mark, the following keyword parameter should be specified:

  TPMK=NO

  It is the responsibility of the user to be able to differentiate between header labels and data.

- Type of File

  The type of file is specified by this parameter.

  – TYPE=INPUT

    This keyword parameter specifies an input file (one that is to be read). This option is assumed if the parameter is not specified.

  – TYPE=OUTPUT

    This keyword parameter specifies an output file (one that is to be written).

  – TYPE = INOUT

    This keyword parameter specifies a file which can be used either as an output or input file.

    It is assumed the user wants the file to be an output type first, and the DTF is set up accordingly. After the use of the file as an output type, the user can change to an input file by resetting the DTF File Type indicator (bit 7 of DC$FG1) before re-opening the file.

    DTFD with MT=YES must be included in the program (1.5.1).

```
LABEL    OPERATION  OPERAND
1        10    16
*  CHANGE FILE     FROM OUTPUT TO INPUT

         DTFD      MT=YES
filename DTFMT     TYPE=INOUT,..........
         .
         .
         .
         .
         .
         NI        filename+DC$FG1,X'FE'

```

If the user wants the option of READ = FORWARD or BACK when a combined file is selected, the READ = BACK parameter must be selected. When this parameter has been selected in a combined file environment, it is assumed that the first time the file is opened as an input type, the user will want the file read in a backward direction. After the use of the input file in a backward direction, the user can change to an input file in a forward direction by resetting the DTF Input File Direction indicator (bit 3 of DC$FG2) before re-opening the file.

DTFD with MT=YES must be included in the program (1.5.1).

```
LABEL      OPERATION   OPERAND                                        COMM
1          10          16
* CHANGE   INPUT       FILE FROM READ=BACK TO READ=FORWARD

           DTFD        MT=YES
filename   DTFMT       TYPE=INPUT,READ=BACK,......
           .
           .
           .
           .
           .
           NI          filename+DC$FG2,X'EF'

```

- Variable-Length Record Residual Space

For output files with variable-length blocked records to be processed in the I/O area, when a work area is not specified, the following keyword parameter is required:

VBLD=(r)

where (r) is the number of the register (2–12) into which logical IOCS places the number of bytes of residual space left in the current I/O area. Before building a record, the user is responsible for determining whether the residual space is large enough for the next record. If it is not, a TRUNC macro instruction must be issued to write out the current block, thereby making available the entire I/O area.

- Work Area Processing

If records in an input file are to be transferred from the input I/O area to a work area, the following keyword parameter must be specified:

WORK=YES

This keyword parameter must also be specified for output files when records are to be built in a work area.

The user must specify the address of the current work area with each GET and PUT macro instruction.

LABEL (1)	OPERATION (10)	OPERAND (16)	72
* MAGNETIC TAPE INPUT FILE DEFINITION - SAMPLE			
XAMPLE	DTFMT	BKNO=YES,	-
		BKSZ=1600,	-
		CKPT=YES,	-
		CLRW=RWD,	-
		EOFA=ENDRUN,	-
		ERROR=CHECKOUT,	-
		FLBL=STD,	-
		IOA1=AREABASE,	-
		IOA2=AREAALT,	-
		RCFM=FIXBLK,	-
		RCSZ=160,	-
		READ=FORWARD,	-
		TYPE=INPUT,	-
		WORK=YES	

KEY-WORD	SPECIFICATION	FILES		REMARKS
		INPUT	OUTPUT	
BKNO	YES	X	X	Performs writing or checking of block numbers
BKSZ	n = maximum block size	R	R	The maximum block size in bytes
CKPT	YES	X		Checkpoint dumps, if present, on input files are bypassed
CLRW	NORWD	X	X	No tape rewind after CLOSE
	RWD	X	X	Rewind tape without interlock after CLOSE
EOFA	symbolic label	R		Identifies end-of-file routine
ERRO	IGNORE	X		Treat error block in input file normally
	SKIP	X		Skip the block containing the error
	symbolic label	X		Address of user routine to process unique errors
ERROR	symbolic label	X	X	Address of user's unrecoverable error routine
FLBL	STD †	Y	Y	For standard labeled files
	NO	Y	Y	For unlabeled files
	NSTD	Y	Y	For nonstandard labeled files
IOA1	symbolic label	R	R	Address of input/output area
IOA2	symbolic label	X	X	Address of alternate input/output area
IORG	(r) = general register	X	X	Required if records are processed in the I/O area and there are two I/O areas or records are blocked
LBAD	symbolic labels of user's label routine		X	Required for output files with FLBL=NSTD
		X		Required for input files with FLBL=NSTD if no tape marks separate header labels from data blocks
OPTION	YES	X		For specifying an optional file
OPRW	NORWD	X	X	No tape rewind before OPEN (does not apply when READ = BACK is specified; NORWD is assumed)

Table 3–1. Summary of Keyword Parameters for the DTFMT Macro Instruction (Part 1 of 2)

KEY-WORD	SPECIFICATION	FILES		REMARKS
		INPUT	OUTPUT	
RCFM	FIXUNB†	Y	Y	For fixed-length, unblocked records; assumed by logical IOCS
	FIXBLK	Y	Y	For fixed-length, blocked records
	VARBLK	Y	Y	For variable-length, blocked records
	VARUNB	Y	Y	For variable-length, unblocked records
	UNDEF	Y	Y	For undefined records
RCSZ	n = number of bytes in record	X	X	For fixed-length, blocked records
	(r) = general register		X	For undefined output records; register contains record size
		X		Optional for undefined input records; register contains record size
READ	FORWARD†	X		Optional to read a tape forward
	BACK	R		Required to read a tape backward
TPMK	NO		X	Write no tape mark for FLBL=NSTD or FLBL=NO
TYPE	INPUT†	X		For input files
	OUTPUT		R	For output files
	INOUT	X	X	For input/output files
VBLD	(r) = general register		X	Required for variable-length, blocked records built in output area; register contains number of bytes left in output area
WORK	YES	X	X	Process records in work area

LEGEND:

R = Required

X = Optional

Y = One option required

† = Assumed parameter, if none specified

Table 3-1. Summary of Keyword Parameters for the DTFMT Macro Instruction (Part 2 of 2)

3.2.2. Direct Access Storage Devices

The declarative macro instruction, DTFSD, is required for input and output files processed in sequential order on the direct access storage subsystem. Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFSD macro instruction.

A description of the individual keyword parameters follows the listing. A summary of the keyword parameters is given in Table 3–2 following the descriptions.

LABEL	♭ OPERATION ♭	OPERAND
filename	DTFSD	BKSZ=n EOFA=symbol (input files only) IOA1=symbol [CNTRL=YES] [DEVICE={8411 | 8414 | 8424}] [ERRO={IGNORE | SKIP | symbol}] [ERROR=symbol] [IOA2=symbol] [IORG=(r)] [LBAD=symbol] [OPTION=YES] [RCFM={FIXUNB | FIXBLK | VARBLK | VARUNB | UNDEF}] [RCSZ={n | (r)}] [TRUNCS=YES] [TYPE={INPUT | OUTPUT | INOUT}] [UPDT=YES] [VBLD=(r)] [VERIFY=YES] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

■ Block Size

This keyword parameter specifies the length of the I/O area and has the following format:

BKSZ=n

where n is the size of the block in bytes. If the records in the file are variable length, n is the maximum size for blocks in the file and must include the block and record length fields.

NOTE: For all output files, n must be a multiple of the record size, plus eight bytes for the count field.

The value for n may not exceed 3625 bytes for UNIVAC 8411 Disc Subsystems, or 7294 bytes for UNIVAC 8414 and UNIVAC 8424 Disc Subsystems (plus eight bytes for output files).

- Control Entry

If a CNTRL macro instruction is to be issued by a problem program, the following keyword parameter must be specified when the file is defined.

CNTRL=YES

When the CNTRL macro instruction is issued, a seek to the current track in the user's file is generated.

- Device Type

The type of direct access storage device upon which the data file resides may be specified by one of the following keyword parameters:

$$\text{DEVICE}=\left\{\begin{matrix}8411\\8414\\8424\end{matrix}\right\}$$

where 8411 represents the UNIVAC 8411 Disc Subsystem, 8414 represents the UNIVAC 8414 Disc Subsystem, or 8424 represents the UNIVAC 8424 Disc Subsystem.

This is not a required parameter; the OPEN transient routines determine the device characteristics from the Physical Unit Block (PUB) device type of the first volume. All other volumes of the file must be consistent with the first.

- End of an Input File

The address to which control is transferred, when the end of data is sensed, must be supplied by the following keyword parameter:

EOFA=symbol

where symbol (label) is the address of the user's end-of-file routine. This is required for input files.

- Unique File Errors

A unique error is an unrecoverable parity error. If a unique error occurs when reading a block of data, the user has the following options: he may process the block as if no error had occurred, bypass the block, enter a special routine for individually processing error blocks, or terminate the job. The absence of this parameter specification will cause logical IOCS to assume that a job should be terminated if a unique error occurs. The other options are specified as follows:

– ERRO=IGNORE

This option permits the records in the block to be available to the user as though no error had occurred.

– ERRO=SKIP

This option causes the block in error to be bypassed and the next error-free block is read. This option should not be used for output files, or for input files with update mode (UPDT=YES).

– ERRO=symbol

For this option, symbol (label) is the address of a user's error routine to process any errors. When logical IOCS transfers control to the user's error routine, register 14 contains the normal return address. Registers 2 through 12 are restored. Register 1 contains the address of the block in question for both input and output files. In addition, the following rules are observed:

1. Do not destroy the present register save area (pointed to by register 13).
2. Return with register 13 containing the save area address.
3. Do not issue a GET or PUT instruction on the file.
4. Access the record by means of register 1.
5. Use another save area for any other IOCS file.
6. Return by transferring control to the address contained in register 14.

■ Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the ERROR address from a processing macro instruction (such as GET and PUT), the address of the instruction following the macro call can be found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN, CLOSE and FEOV). Register 0 contains the following information concerning the reasons for the error:

– Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary zeros.

– Byte 2 contains a one-character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.

– Byte 3 contains a one-character EBCDIC value indicating the reason for the error; refer to Appendix E.

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs, but the console typeout still appears.

Further access of the file is not permitted after a major file error has occurred.

- Input/Output Area

Each input or output file must have an area reserved for its individual use. The area must be defined by the following keyword parameter.

IOA1=symbol

where symbol (label) is the address of the I/O area. The length of the area is specified by the keyword parameter BKSZ. The I/O area must be defined as a multiple of the record size plus, for output files, the count field consisting of the first eight bytes of the area (see 2.3.9).

- Secondary Input/Output Area

A secondary I/O area for standby processing improves efficiency by overlapping I/O and record processing time. The format of the keyword parameter for a secondary I/O area is:

IOA2=symbol

where symbol (label) is the address of the secondary I/O area. It is subject to the same requirements as noted in IOA1.

- Current Record Pointer

When a general register is used as an index register to reference current data, the following keyword parameter should be specified:

IORG=(r)

where (r) is the number of the general register (2–12). If a work area is not required, this keyword parameter must be specified when there are two I/O areas or records are blocked.

For input files, logical IOCS places the address of the next available record in the specified register.

For output files, logical IOCS places in the specified register the address of the next available I/O output area.

- Special Label Handling

This keyword parameter supplies the address of a user routine which processes user standard labels (UHLn/UTLn). This format is:

LEAD=symbol

where symbol (label) is the address of the user routine.

During the processing of the OPEN macro instruction, the user receives control after the header label has been checked on an input file or after the header label has been written on an output file.

For both input and output files, logical IOCS places the EBCDIC character O, denoting OPEN, in the low order byte of register 0, and places the address of the I/O area in register 1. The input data pointed to by the address in register 1 is available for processing; labels for output files can be built in the I/O area pointed to by register 1. The special label handling routine must return control to logical IOCS by means of the LBRET macro instruction with the proper option specified (see 3.3.7). Logical IOCS completes the processing of the OPEN macro instruction and prepares the file for the first GET or PUT macro instruction.

For user trailer labels, after reading or writing the end-of-file record or reaching the end-of-volume address and before transferring control to the special label handling routine, logical IOCS loads the EBCDIC character V or F, whichever is applicable, into register 0. The conventions regarding the use of the LBRET macro instruction are the same as those for processing the user header labels.

After an end of volume has been reached, logical IOCS transfers control to the special label handling routine with the address of the I/O area (IOA1) in register 1.

User header and trailer labels are read into IOA1. On input, register 1 always points to the proper area (IOA1) into which the label has been read. On output, register 1 always points to the area (IOA1) into which the user is expected to place the label to be written. The label is moved and written from the transient area. Since the length of a user label (UHL1–8 or UTL0–7) is always 80 characters, IOA1 must be 80 characters minimum or needed data may be destroyed.

In processing multiple labels, note that user label handling procedures are entered at the LBAD address for each label, and that LBAD is the same for both header and trailer labels. The user must, therefore, expect to identify the label being processed. (Use of the LBRET (2) macro instruction does not imply that control is returned following the LBRET (2) instruction, but that return is made at the LBAD address.) Each time LBAD is entered, the registers (other than 0 and 1) are set as they were when the appropriate macro call (PUT, TRUNC, OPEN, CLOSE, or FEOV), which resulted in label handling procedures being instituted, was issued. The user is, therefore, cautioned not to save any information in a register which he may require at the next entry to LBAD, or upon return to inline user coding. The table in Appendix A.2 indicates the conditions and requirements for label procedures.

After an end of file has been reached, logical IOCS transfers control to the user's end-of-file routine (EOFA). The end-of-file routine should issue a CLOSE macro instruction for the file. During the processing of the CLOSE macro instruction, logical IOCS transfers control to the special label handling routine if there are other user trailer labels.

- Optional Input File

The keyword parameter to allow the user to specify an optional input file follows:

OPTION=YES

When this keyword parameter is specified and the file has not been allocated to a device by Job Control, the OPEN transient routine ensures that logical IOCS transfers control to the address of the user's end-of-file routine (EOFA) following the execution of the first GET macro instruction. This procedure maintains continuity in the user's program. The user is required to CLOSE the optional file.

When this keyword parameter is not specified and the file has not been allocated by Job Control, it is impossible to obtain records from the file. Logical IOCS transfers control to the address of the user's error routine (ERROR). If a user error routine is not supplied, the job is aborted.

- Record Format

The record format of a disc file is specified by a keyword parameter. There are five options to indicate whether the records are fixed length, variable length, blocked, unblocked, or undefined. The options are coded as follows:

- RCFM=FIXUNB Fixed-length, unblocked
- RCFM=FIXBLK Fixed-length, blocked
- RCFM=VARUNB Variable-length, unblocked
- RCFM=VARBLK Variable-length, blocked
- RCFM=UNDEF Undefined

Logical IOCS assumes a fixed-length, unblocked record format when this keyword parameter is not specified.

- Record Size

The number of bytes in a record is made available to logical IOCS by specifying this keyword parameter. Logical IOCS assumes that the record size is equal to the block size for fixed-length, unblocked records and that the size of the record is available in the first two bytes of each variable-length record.

- RCSZ=n

  This form of the keyword parameter should be used for fixed-length, blocked records, where n is the number of bytes in the record.

- RCSZ=(r)

  This form of the keyword parameter is required for output files with undefined record formats. The general register (2–12) specified by (r) must contain the block length.

  The keyword parameter may also be used for input files with undefined records. Logical IOCS places the block length into the general register specified by (r).

- Short Blocks

The following keyword parameter is provided only for purposes of compatibility with comparable systems:

TRUNCS=YES

Logical IOCS is always prepared for the possibility of short blocks on an input file of fixed-length blocked records, or for output files of either fixed- or variable-length blocked records.

- Type of File

  The type of file is specified by this parameter.

  – TYPE=INPUT

    This keyword parameter specifies an input file (one that is to be read). This option is assumed if this keyword parameter is not specified.

  – TYPE=OUTPUT

    This keyword parameter specifies an output file (one that is to be written).

  – TYPE=INOUT

    This keyword parameter specifies a file which can be used as either an output or an input file.

    The user expects the file to be an output type first; the DTF is set up accordingly. After the use of the file as an output type, the user changes to an input file by resetting the DTF File Type indicator (bit 7 of DC$FG1) and by decrementing the block size by 8 in DC$BKS before reopening the file. DTFD with SD=YES must be included in the program (see 1.5.1).

```
LABEL      OPERATION  OPERAND
1          10         16
           DTFD       SD=YES
filename   DTFSD      TYPE=INOUT,BKSZ=n,.........
           .
           .
           .
           .
           .
           .
           .
           .
           NI         filename+DC$FG1,X'FE'
           AI         filename+DC$BKS,-8
```

The significance of the file-type indicator bits (bits 6 and 7 of DC$FG1) follows:

Bit 6	Bit 7	
0	0	Input file
0	1	Output file
1	0	Input file with update
1	1	Not used (invalid)

Users with file updating (UPDT=YES) requirements must specify the UPDT keyword parameter to ensure the inclusion of the correct modules.

- File Updating

When an input file on a direct access storage device is to be updated, the following keyword parameter must be specified.

UPDT=YES

Each record is read, modified if necessary, and if modified then written back into the same storage location. This applies only to input or inout files.

- Variable-Length Record Residual Space

For output files with variable-length blocked records to be processed in the I/O area and a work area is not specified, the following keyword parameter is required:

VBLD=(r)

where (r) is the number of the register (2–12) into which logical IOCS places the number of bytes of residual space left in the current I/O area. Before building a record, the user is responsible for determining whether the residual space is large enough for the next record. If it is not, a TRUNC macro instruction must be issued to write out the current block, thereby making available the entire I/O area.

- Write Verification

When records should be check-read after they have been written, the following keyword parameter should be specified:

VERIFY=YES

When both the UPDT=YES and TYPE=INPUT parameters are specified, the VERIFY option may be chosen to check-read the updated records.

Data Management reads back one byte of data and checks for a successful completion thus verifying that the record transferred was written correctly. The hardware cyclic check feature establishes if the transfer was completed successfully. Verification of records necessarily increases the execution time for the commands associated with the WRITE macro instruction.

- Work Area Processing

If records in an input file are to be transferred from the input I/O area to a work area, the following keyword parameter must be specified.

WORK=YES

This keyword parameter must also be specified for output files when records are to be built in a work area.

The user must specify the address of the current work area with each GET and PUT macro instruction.

```
LABEL      OPERATION      OPERAND                                          72
1          10             16
*  DISC SEQUENT IAL INPUT FILE - SAMPLE
ACCNTS     DTFSD  BKSZ=800,                                               X
                  IOA1=READ,                                              X
                  EOFA=FINIS,                                             X
                  ERRO=IGNORE,                                            X
                  RCFM=FIXBLK,                                            X
                  RCSZ=100,                                               X
                  TYPE=INPUT,                                             X
                  UPDT=YES,                                               X
                  WORK=YES
```

NOTE: For output files, BKSZ in this example must equal 808 to include the eight-byte count field.

KEY-WORD	SPECIFICATION	FILES		REMARKS
		INPUT	OUTPUT	
BKSZ	n = maximum block size	R	R	The maximum block size, in bytes
CNTRL	YES	X	X	A CNTRL macro will be issued for the file.
DEVICE	8411, 8414, or 8424	X	X	Identifies the direct access storage device
EOFA	symbolic label	R		Identifies EOF routine
ERRO	IGNORE	X	X	Treat error block in input file normally. On output, VERIFY=YES is specified.
	SKIP	X		Skip the block containing the error
	symbolic label of user's error routine	X	X	User handles the block in error. On output, VERIFY=YES is specified.
ERROR	symbolic label	X	X	Address of user's unrecoverable error routine
IOA1	symbolic label	R	R	Address of input/output area
IOA2	symbolic label	X	X	Address of alternate input/output area
IORG	(r) = general register	X	X	Required if records are processed in the I/O area and there are two I/O areas or records are blocked

LEGEND:

R = Required Y = One option required

X = Optional † = Assumed parameter, if none specified

Table 3-2. Summary of Keyword Parameters for the DTFSD Macro Instruction (Part 1 of 2)

KEY-WORD	SPECIFICATION	FILES		REMARKS
		INPUT	OUTPUT	
LBAD	symbolic label of user's label routine		X	Required if user header or trailer labels are to be created
		X		Required if user header or trailer labels are to be checked
OPTION	YES	X		For specifying an optional file
RCFM	FIXUNB†	X	X	For fixed-length, unblocked records; assumed
	FIXBLK	Y	Y	For fixed-length, blocked records
	VARBLK	Y	Y	For variable-length, blocked records
	VARUNB	Y	Y	For variable-length, unblocked records
	UNDEF	Y	Y	For undefined records
RCSZ	n = number of bytes in record	X	X	For fixed-length, blocked records
	(r) = general register		X	For undefined output records; register contains record size
		X		Optional for undefined input records; register contains record size
→ TRUNCS	YES	X		For use if a fixed-length, blocked disc file contains short blocks
→			X	For use if short blocks are to be written for a fixed-length, blocked record of a disc file
TYPE	INPUT†	X		For input files; assumed
	OUTPUT		R	For output files
	INOUT	X	X	For input/output files
UPDT	YES	X		This parameter is required if records are to be written back to the same location from which they were read
VBLD	(r) = general register		X	Required for variable-length, blocked records built in output area; register contains number of bytes left in output area
VERIFY	YES	X	X	Check parity after records have been written
WORK	YES	X	X	Process records in work area

LEGEND:

R = Required
X = Optional
Y = One option required
† = Assumed parameter, if none specified

Table 3–2. Summary of Keyword Parameters for the DTFSD Macro Instruction (Part 2 of 2)

3.2.3. Card Device

The declarative macro instruction, DTFCD, is required for card input and output files. It serves the UNIVAC 9000 Series 600 cpm Reader, Type 0711, the 600/1000 cpm Reader, Type 0716, and the Row Punch Subsystem, Type 0604. Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFCD macro instruction.

A description of each keyword parameter follows the listing. A summary of the keyword parameters is given in Table 3–3 following the descriptions.

LABEL	♭ OPERATION ♭	OPERAND
filename	DTFCD	BKSZ=n EOFA=symbol (Input files only) IOA1=symbol MODE={BINARY / STD / CC / TRANS} RCFM={FIXUNB / UNDEF / VARUNB} STUB={51 / 66} TYPE={COMBND / INPUT / OUTPUT} [AUE=YES] [CNTRL=YES] [CTLCHR=YES] [ERROR=symbol] [IOA2=symbol] (Required for combined files) [IORG=(r)] [ITBL=symbol] [OBSZ=n] [ORLP=YES] [OPTION=YES] [OTBL=symbol] [PUNR=YES] [RCSZ=(r)] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- Validity Check

  This keyword parameter is to be used only with card input files (TYPE=INPUT) and specifies that data cards that fail the hardware "validity check" are to be ignored. Validity check is an optional hardware feature which must be present in the hardware in order to use this keyword parameter. The format is:

  AUE=YES

  The validity check detects the presence of more than one punched hole in the numeric rows 1 through 7 of any card column during reading of 80-, 66- or 51-column cards in the hardware translate (EBCDIC) mode. This mode is determined by specifying the keyword parameter MODE=STD.

  If the reader is a Type 0716, this specification also sets up the hardware to function in the SORT-ON-ERRORS mode, in which case all error cards at the completion of reading are sorted into the reject stacker.

- Block Size

  This keyword parameter specifies the length of the I/O area and has the following format:

  BKSZ=n

  where n is the size of the block in bytes. If the records in a file have an undefined format, then n specifies the size of the largest record.

  When the 51- and 66-column stub card read feature is used, the block size must be specified correctly. A block size that is larger than the number of columns in the cards causes transmission of erroneous data. A smaller block size may be specified if all columns are not used.

  If this keyword parameter is omitted, the block size (80 or 160) is determined by the keyword parameter MODE.

- Control Entry

  If a CNTRL macro instruction is to be issued by a problem program, the following keyword parameter must be specified when the file is defined.

  CNTRL=YES

  Use of the keyword parameter indicates that a CNTRL macro instruction to be issued for stacked selection on the UNIVAC 0604 Card Punch Subsystem. To select the stacker for a particular card, the CNTRL macro instruction that selects the stacker must be issued after the PUT macro instruction that punched the card or it must precede the PUT macro instruction that delivers the card from the postread station to the selected stacker.

  This keyword parameter is also used if the unit is to operate in the read/punch mode. When operating in the read/punch mode with or without overlap, each CNTRL macro instruction issued must correspond to a GET or a PUT macro instruction. The CNTRL macro instruction can be issued after the GET or PUT macro instruction that reads or punches the card to be controlled, or it can be issued before the GET or PUT macro instruction that delivers the card from the post-read station to the selected stacker.

  The CNTRL macro instruction does not apply to input files on the 600 cpm reader.

  If the first character control (CTLCHR=YES) or punch recovery (PUNR=YES) is specified for an output file, the CNTRL keyword parameter must not be specified.

- First Character Control

  The following keyword parameter must be specified when a control character is to be used in data records.

  CTLCHR=YES

  However, this parameter does not apply to input or combined files. Also, punch recovery (PUNR=YES) should not be specified when a control character is used. If the CTLCHR keyword parameter is specified, every record in the file must contain a control character. The character is normally used for stacker selection on the UNIVAC 0604 Card Punch Subsystem. Command codes for the punch are as follows:

0xcdef01 (Write)

0xcdex11 (Device Control)

where:

c=1 Feed a card and select stacker. The card that was punched on the previous punch order is placed in the selected stacker.

d=1 Feed and punch a card.

e=1 Feed a card.

f=0 Write in compressed mode.

f=1 Write in image mode.

x Ignored by the control unit.

- End of an Input File

The address to which control is transferred when the end-of-data card is sensed must be supplied by the following keyword parameter:

EOFA=symbol

where symbol (label) is the address of the routine which handles end-of-data processing. This keyword parameter is required for all input and combined files.

- Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the ERROR address from a processing macro instruction (such as GET and PUT), the address of the instruction following the macro call is found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN and CLOSE). Register 0 contains the following information concerning the reasons for the error:

- Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary 0's.

- Byte 2 contains a one character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.

- Byte 3 contains a one character EBCDIC value indicating the reason for the error; refer to Appendix E.

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs, but the console typeout still appears.

- Input/Output Area

  Each input or output file must have an area reserved for its individual use. The area must be defined by the keyword parameter.

  IOA1=symbol

  where symbol (label) is the address of the I/O area. This keyword parameter specifies the input area for a combined file.

- Secondary Input/Output Area

  A secondary I/O area for standby processing may be specified by the following keyword parameter:

  IOA2=symbol

  where symbol (label) is the address of the secondary I/O area. This keyword parameter specifies the output area for a combined file.

- Current Record Pointer

  When a general register is used to reference current data, the following keyword parameter is specified:

  IORG=(r)

  where (r) may be general registers 2 through 12. The register must be specified if two I/O areas are used, if records are not to be processed in a work area, and if the file is not combined.

  For input files, logical IOCS places the address of the next available record in the specified register.

  For output files, logical IOCS places the address of the next available I/O output area in the specified register.

- Input Translation Table

  Where records in a combined file are to be translated, the following keyword parameter is specified

  ITBL=symbol

  where symbol (label) is the address of the translation table in the problem program. If the keyword parameter MODE=TRANS is specified, the keyword parameter ITBL must also be specified.

- Input/Output Mode

  This keyword parameter is used to specify the input/output mode of the file and is required as part of the DTFCD macro instruction. There are four forms of the keyword parameter which can be used with the two card devices.

  – MODE=BINARY

    This form is used for cards read on the 600 cpm Reader, Type 0711, or the 600/1000 cpm Reader, Type 0716, in column binary mode or for cards read or punched on the Row Punch Subsystem, Type 0604, in column binary (image) mode. An I/O area of 160 bytes is required. The card code image is untranslated.

– MODE=STD

On the 600 cpm Reader, Type 0711, or the 600/1000 cpm Reader, Type 0716, when cards are to be read with hardware translation, this form should be specified. The hardware translation is to EBCDIC code. This option is assumed if this keyword parameter is not specified.

On the Row Punch Subsystem, Type 0604, when cards are read in compressed code and translated to EBCDIC code or when cards are translated from EBCDIC code and punched in compressed code, this form should be specified.

– MODE=CC

On the Row Punch Subsystem, Type 0604, this form must be specified for cards read or punched in compressed code without translation. An I/O area of 80 bytes is usually required.

– MODE=TRANS

On the Row Punch Subsystem, Type 0604, this form should be specified for cards read or punched in compressed code and translated by the table specified by the ITBL or OTBL keyword parameter, respectively.

■ Output Block Size

For a combined file, the length of the secondary I/O area (IOA2) is specified by the following parameter:

OBSZ=n

where n is the length, in bytes, of the area. If IOA2 is specified and OBSZ is omitted, the size of the output block is assumed to be the same length as BKSZ.

■ Overlap

When the Row Read/Punch unit is to be processed in an overlap mode, the following keyword parameter should be specified:

ORLP=YES

The preread feature enables the punch to preread cards. If the information read during the previous cycle is to be processed, it is necessary to transfer the information from the preread buffer before issuing another punch order. A GET macro instruction issued for the file initiates the reading of the next card as well as transferring the previous card image from the buffer to the user input I/O area.

Where overlap is not specified, a GET macro instruction causes a transfer, or a transfer and translate, of the input record. A PUT macro instruction causes a transfer of the output image from the output I/O area to the punch buffer and then to the card, or it causes a transfer with translation from the output I/O area to the buffer and then from the buffer to the card.

In the overlap mode, each GET and PUT macro instruction causes the read/punch unit to advance one card. Without overlap, the unit advances one card on PUT macro instructions and also when one GET is followed by another GET macro instruction.

Three possible combinations for issuing GET and PUT macro instructions with overlap specified are explained in the following paragraphs.

- Alternating GET and PUT macro instructions when used with alternating prepunched and blank cards produce valid results if each GET macro instruction applies to prepunched input data cards and if each PUT macro instruction applies to punching data into a blank card.

- Multiple GET macro instructions between single PUT macro instructions when used with multiple prepunched cards between single blank cards produce valid results if, in every case, the number of GET macro instructions corresponds to the number of prepunched cards between each of the blanks that the PUT macro instructions references.

- Multiple GET and multiple PUT macro instructions when used with multiple prepunched cards between multiple blanks produce valid results if the number of GET macro instructions and PUT macro instructions and the number of prepunched and blank cards are consistent through the program.

Out-of-order records in the data file produce errors.

■ Optional Input File

Some users may wish to define files for which at some specific run time no data exists; for example, when he may desire to issue change notices against a master file. In this case, he could specify an optional input file. The keyword parameter that allows the user to specify an optional input file follows:

OPTION=YES

When this keyword parameter is specified and the file has not been allocated to a device by Job Control, the OPEN transient routine ensures that logical IOCS transfers control to the address of the user's end-of-file routine (EOFA) following the execution of the first GET macro instruction. This procedure maintains continuity in the user's program. The user is required to CLOSE the optional file.

When this keyword parameter is not specified and the file has not been allocated by Job Control, it is impossible to obtain records from the file. Logical IOCS transfers control to the address of the user's error routine (ERROR). If a user error routine is not supplied, the job step is aborted.

■ Output Translation Table

When records in an output or combined file are to be translated, the following keyword parameter is specified:

OTBL=symbol

where symbol (label) is the address of the translation table in the problem program. A translation table is required if the keyword parameter MODE=TRANS is specified.

■ Punch Error Recovery

If a card punch error recovery should be attempted, the following keyword parameter should be specified:

PUNR=YES

Error cards are automatically selected into the Error Select Stacker. If error recovery is not successful, logical IOCS returns control to the address of the user's error routine (ERROR). The user can access the card image by means of the address found in filenameS in the DTFCD table. If PUNR and ERROR are not specified, logical IOCS assumes that the job must be terminated when a punch error (hole count error) is encountered.

Error recovery is not possible with combined reading and punching. Also, the CNTRL=YES or CTLCHR=YES parameters cannot be specified when punch error recovery is specified.

- Record Format

One of the three following options describing the record format should be specified.

– RCFM=FIXUNB

Fixed-length records for input and combined files are assumed by logical IOCS when this keyword parameter is omitted.

– RCFM=UNDEF

This form is used for undefined records in output files only.

– RCFM=VARUNB

This form is used for variable-length, unblocked records in output files only.

- Record Size

For output files with undefined record format, the following keyword parameter is specified:

RCSZ=(r)

where (r) indicates the number (2–12) of the general register that holds the size of the output record. The record size must be entered into the general register before the PUT macro instruction is issued.

- Stub Cards

One of the following options describing the size of the stub card should be specified if the stub card feature has been installed on the 600/1000 cpm Reader, Type 0716.

The keyword parameter MODE=STD must be specified.

– STUB=51

This option describes a stub card of 51 columns as input from the 600/1000 cpm Reader, Type 0716.

– STUB=66

This option describes a stub card of 66 columns as input from the 600/1000 cpm Reader, Type 0716.

- Type of File

The type of file must be specified by one of the following three keyword parameters.

– TYPE=INPUT

This option describes an input file from the 600 cpm UNIVAC 0711 Card Reader Subsystem, or the 600/1000 cpm UNIVAC 0716 Card Reader Subsystem. This option is assumed if this keyword parameter is not specified.

– TYPE=OUTPUT

This option describes an output file for the UNIVAC 0604 Card Punch Subsystem.

NOTE: When an output card file is closed, a /* card will be punched as the last data card for the file in the mode of the file.

– TYPE=COMBND

This option describes the combined file of the UNIVAC 0604 Card Punch Subsystem when the read/punch feature is to be used.

■ Work Area Processing

If a work area is required in which to process I/O records rather than processing the records in the I/O area, the following parameter is specified:

WORK=YES

The address of the current work area must be specified with each GET or PUT macro instruction.

```
LABEL      OPERATION      OPERAND                                  72
1          10             16
*  READER  FILE   DEFINITION - SAMPLE
CARDIN     DTFCD  IOA1=AREA1,                                      X
                  BKSZ=80,                                        X
                  EOFA=LASTCD,                                    X
                  RCFM=FIXUNB,                                    X
                  TYPE=INPUT,                                     X
                  WORK=YES
*  PUNCH   FILE   DEFINITION - SAMPLE
CARDOUT    DTFCD  IOA1=AREA1,
                  IOA2=AREA2,                                     X
                  IORG=(5),                                       X
                  BKSZ=80,                                        X
                  RCFM=UNDEF,                                     X
                  RCSZ=(7),                                       X
                  TYPE=OUTPUT,                                    X
                  MODE=STD,                                       X
                  PUNR=YES,                                       X
                  ERROR=EXIT
```

KEY-WORD	SPECIFICATION	FILES			REMARKS
		INPUT	OUTPUT	COMBND	
AUE	YES	X			Mispunched cards accepted
BKSZ	n = maximum block size	R	R	R	The maximum block size in bytes
CNTRL	YES		X	X	Specify if CNTRL macro is issued to file; CTLCHR must be omitted
CTLCHR	YES		X		For First Character Control; CNTRL must be omitted
EOFA	symbolic label	R		R	End-of-file routine for input and combined files
ERROR	symbolic label of user's error routine	X	X	X	Address of the user's unrecoverable error routine
IOA1	symbolic label	R	R	R	Address of input/output area; input area for a combined file
IOA2	symbolic label	X	X	R	Address of alternate input/output area; output area for a combined file
IORG	(r) = general register (2–12)	X	X		General register that contains the address of the current record when processing in two I/O areas. Omit WORK=YES. Must not be used for a combined file
ITBL	symbolic label			X	Address of input translate table
MODE	BINARY	Y	Y	Y	Specifies cards are to be read or punched in column binary
	CC		Y	Y	Specifies cards are to be read or punched in compressed code
	STD		Y	Y	Automatically translates from compressed code to internal code and vice versa
	TRANS		Y	Y	For cards to be read in or punched in compressed code and translated by the table specified in the ITBL or OTBL entry, respectively

LEGEND:

R = Required

X = Optional

Y = One option required

Table 3–3. Summary of Keyword Parameters for the DTFCD Macro Instruction (Part 1 of 2)

KEY-WORD	SPECIFICATION	FILES			REMARKS
		INPUT	OUTPUT	COMBND	
OBSZ	n = length of IOA2			R	Specifies the length of IOA2 for a combined file
ORLP	YES			X	Specifies that a read/punch unit file is to be processed in an overlap mode
OPTION	YES	X			Specifies an optional file
OTBL	symbolic label		X	X	Address of output translate table
PUNR	YES		X		Automatic punch error recovery (not specified with combined files)
RCFM	FIXUNB	R	Y	R	For fixed-length records
	UNDEF		Y		For undefined records
	VARUNB		Y		For variable-length records
RCSZ	(r) = general register (2–12)		X		For undefined records, register contains record size
STUB	51	R			For 51-column cards only
	66	R			For 66-column cards only
TYPE	COMBND			R	For combined read/punch file
	INPUT	R			For input files
	OUTPUT		R		For output files
WORK	YES	X	X	X	Records are to be processed in work area

LEGEND:

R = Required

X = Optional

Y = One option required

Table 3–3. Summary of Keyword Parameters for the DTFCD Macro Instruction (Part 2 of 2)

3.2.4. Printer

The declarative macro instruction, DTFPR, is required for each printer file processed in the program. It serves the UNIVAC 9000 Series 900/1100 LPM Drum Printer, Type 0768-00 and Type 0768-01, as well as the line printers furnished with the 1004/1005 Subsystems. Refer to Table 3–4 for a listing of the pertinent differences between the two types of printers. Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFPR macro instruction.

A description of each keyword parameter follows the listing. A summary of the keyword parameters is given in Table 3–5 following the descriptions.

LABEL	OPERATION	OPERAND
filename	DTFPR	BKSZ=n IOA1=symbol RCFM= {FIXUNB / UNDEF / VARUNB} [AUE=YES] [CNTRL=YES] [CODE=symbol] [CTLCHR=YES] [ERROR=symbol] [IOA2=symbol] [IORG=(r)] [PRAD=n] [PRTOV= {YES / symbol}] [RCSZ=(r)] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- Character Mismatch

  This keyword parameter and the following format allows the user to ignore a character mismatch:

  AUE=YES

  A mismatch occurs whenever the printer attempts to print a bit configuration which is not present in the printer's load code buffer. All unprintable characters are printed as the nonprinting code (NP) in the load code buffer. (When the standard load code, Table 3–6, is used, a blank space (40_{16}) is printed.)

- Block Size

  This keyword parameter and the format specifies the length of the I/O area:

  BKSZ=n

  where n is the size of the block in bytes. If the record format is undefined, n specifies the size of the largest record.

  If this keyword parameter is omitted, the assumed size of the block is determined by the RCFM and CTLCHR parameter specifications. (BKSZ may be a maximum of 141 bytes when CTLCHR=YES and RCFM=VARUNB.)

- Control Entry

  The following keyword parameter is specified if spacing or skipping of lines on the printer is controlled from the problem program by the CNTRL macro instruction.

  CNTRL=YES

  The form on the printer is under the control of the paper tape carriage-control loop.

- Printer Code Conversion

  A printer code conversion table is required by the printer control unit. The user has the option of supplying his own code conversion table by specifying the following keyword parameter:

  CODE=symbol

  where symbol (label) is the address of the user's code conversion table located within the problem program. The table requires 64 bytes and the sequence of loading must be in accordance with Table 3–6. When an OPEN macro instruction is issued for a print file, the user's code conversion table is loaded into the 64-byte printer buffer area for use.

  If the keyword parameter is omitted, the Standard Printer Code Conversion Table (Table 3–7) as loaded by Job Control into the 64-byte print buffer area is effective.

  The complete 63-character type font is repeated, in a checkerboard pattern, around the circumference of the drum for each of the 132 print positions. A complete 63-character set can be printed in one full revolution of the drum. Any subset of 49 contiguous characters is printed in about 78% of one drum revolution. The faster printer can print any subset of 43 contiguous characters in about 68% of one revolution.

  Printing of a line begins as soon as the print line buffer has been filled and is independent of the position of the rotating drum. Printing continues until all characters for the line have been printed, a condition recognized by an "end print" detector. Consequently, both the beginning and the ending of the actual printing of a line occur with the drum in varying angular positions.

- First Character Control

  The following keyword parameter must be specified when a control character is to be used in data records.

  CTLCHR=YES

  If the keyword parameter is specified, every record in the file must contain a control character. The control character is used to space over a specific number of lines or to skip to a specific location on the form. Command codes for the printer are as follows:

0cdef001	(Print)
0cdef011	(Control – advance, no print)

The meaning of the values for c, d, e, and f singly and in combination are:

c	d	e	f	Meaning
0	0	0	0	No advance
0	0	0	1	Advance one line
0	0	1	0	Advance two lines
0	0	1	1	Advance three lines
0	1	0	0	Paper is advanced under control of the Forms Control tape loop to the line corresponding to the hole combination in the loop.
.	.	.	.	
.	.	.	.	
.	.	.	.	
1	1	1	1	

Print Character Differences.

HEX CODE	UNIVAC 0768 PRINTER CHARACTER	1004 PRINTER CHARACTER
4A	¢ (CENTS SIGN)	Δ (DELTA)
4F	| (ABSOLUTE)	[(LEFT BRACKET)
5F	¬ (LOGICAL NOT)	≠ (NOT EQUAL)
6D	— (UNDERSCORE)	¤ (LOZENGE)
7F	" (QUOTE)	] (RIGHT BRACKET)

Skip/Code Conversion Table

COMMAND FOR TYPE 0768 PRINTER	COMMAND FOR TYPE 1004 PRINTER
NO SPACE	NO SPACE
SPACE 1 LINE	SPACE 1 LINE
SPACE 2 LINES	SPACE 2 LINES
SPACE 3 LINES	SPACE 3 LINES
SKIP 4	SKIP 2
SKIP 5	SKIP 3
SKIP 6	SKIP 4
SKIP 7	SKIP 5
SKIP 8	SKIP 6
SKIP 9 (FORM OVERFLOW)	SKIP 1 (FORM OVERFLOW)
SKIP 10	NO FORM ADVANCE TAKES PLACE WITH THESE SKIP OPTIONS FOR EITHER A WRITE OR A CONTROL COMMAND.
SKIP 11	
SKIP 12	
SKIP 13	
SKIP 14 (HOME PAPER 6 LPI)	SKIP 7 (HOME PAPER)
SKIP 15 (HOME PAPER 8 LPI)	SKIP 7 (HOME PAPER)
LOAD CODE	NO OPERATION

Table 3–4. Differences Between UNIVAC 0768 and 1004/1005 Printers

If this keyword parameter is specified, then the keyword parameter CNTRL=YES should not be specified.

– Home Paper Code

By programming conventions, the code 111f in the Forms Control tape loop is reserved for home paper. It requires no special recognition in the control unit. The f bit is used to control whether the spacing is six or eight lines per inch.

When f=0, space six lines per inch.

When f=1, space eight lines per inch.

– Form Overflow

The code 1001 in the Forms Control tape loop is reserved to specify an end of form.

Response to form overflow is a function of the number of I/O areas and work areas assigned and may be illustrated as follows:

IOA1	IOA2	WORK AREA	REMARKS
YES	NO	NO	One line printed after overflow is detected overflow action follows
YES	NO	YES	Two lines printed after overflow is detected overflow action follows
YES	YES	NO	
YES	YES	YES	

■ Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the ERROR address from a processing macro instruction (such as GET and PUT), the address of the instruction following the macro call is found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN and CLOSE). Register 0 contains the following information concerning the reasons for the error:

– Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary 0's.

- Byte 2 contains a one character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.

- Byte 3 contains a one character EBCDIC value indicating the reason for the error; refer to Appendix E.

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If the keyword parameter is not specified, abort procedures are executed when a major file error occurs; the console typeout still appears.

- Input/Output Area

Each input or output file must have an area reserved for its individual use. The area must be defined by the following keyword parameter:

IOA1=symbol

where symbol (label) is the address of the I/O area.

- Secondary Input/Output Area

A secondary I/O area for standby processing may be specified by the following keyword parameter:

IOA2=symbol

where symbol (label) is the address of the secondary I/O area.

- Current Record Pointer

When a general register is used to reference current data, the following keyword parameter is specified:

IORG=(r)

where (r) may be general registers 2 through 12. The register must be specified if two output I/O areas are used and records are not to be processed in a work area.

Logical IOCS places in the specified register the address of the next available output I/O area.

- Printer Advance

A standard form advance of one, two, or three lines is specified by the following parameter:

PRAD=n

where n equals 1, 2, or 3 to indicate single, double, or triple spacing respectively. The form advance takes place after the line is printed. If n equals 0, no form advance takes place.

If the keyword parameter is not specified, a value of 1 is assumed. A delayed CNTRL macro instruction to space or skip after printing overrides this keyword parameter for one print line only. The keyword parameter should not be specified if CTLCHR=YES is used with the file.

■ Printer Overflow

This keyword parameter specifies the operation to be performed when a carriage overflow condition occurs. The carriage tape channel 9 (1001_2) is used to indicate the overflow. If the keyword parameter is not specified, printer spacing is controlled by the printer advance keyword parameter or the CNTRL macro instruction specifying spacing or skipping.

There are two options available for handling a carriage overflow condition.

– PRTOV=YES

This option provides an automatic skip to home paper in the paper tape loop.

– PRTOV=symbol

When this option is specified and overflow occurs, logical IOCS transfers control to the symbolic label of the address of the user's overflow routine.

The printer carriage is not automatically restored to the home paper channel. (The form overflow punch is not recognized during a printer skip operation.) To restore the printer carriage to home paper, a CNTRL macro instruction should be issued. The user must have specified the keyword parameter CNTRL=YES when the file was defined before the CNTRL macro instruction can be issued.

The overflow routine may also print total lines, skip to home paper, and print overflow page headings. The routine should return control to logical IOCS at the address stored in general register 14. Note that if imperative macro instructions are issued during the user PRTOV routine, register 14 must first be stored and reloaded; see 1.4.

■ Record Format

Refer to Figure 2–26. One of the three following options describing the record format should be specified:

–RCFM=FIXUNB

Fixed-length records for print files are assumed by logical IOCS when this keyword parameter is omitted.

–RCFM=UNDEF

This form is used for undefined records.

–RCFM=VARUNB

This form is used for variable-length, unblocked records.

- Record Size

  For print files with undefined format, the following keyword parameter is specified:

  RCSZ=(r)

  where (r) indicates the number (2–12) of the general register that holds the size of the output record. The record size must be entered into the general register before the PUT macro instruction is issued.

- Work Area Processing

  If a work area is required in which to process output records rather than in the I/O area, the following parameter is specified.

  WORK=YES

  The address of the current work area must be specified with each PUT macro instruction.

```
LABEL      OPERATION      OPERAND                                          72
1          10             16
*PRINTER   FILE           DEFINITION-SAMPLE
PARTLST    DTFPR          IOA1=OUT1,                                       X
           IOA2=OUT2,                                                      X
           IORG=(4),                                                       X
           BKSZ=141,                                                       X
           CTLCHR=YES,                                                     X
           PRTOV=TOTALS,                                                   X
           RCFM=VARUNB,                                                    X
           ERROR=EXIT
```

KEY-WORD	SPECIFICATION	FILE OUTPUT	REMARKS
AUE	YES	X	Mismatched characters allowed
BKSZ	n=maximum block size	R	The maximum block size
CNTRL	YES	X	Specify if CNTRL macro instruction is issued to file
CODE	symbolic label	X	Address of user's Load Code Table (64-byte table)
CTLCHR	YES	X	Indicates a control character is a part of each record
ERROR	symbolic label	X	Address of user's unrecoverable error routine
IOA1	symbolic label	R	Address of output area
IOA2	symbolic label	X	Address of alternate output area
IORG	(r) = general register	X	General register (2–12) that contains the address of the current record each time a PUT is issued
PRAD	n = 0, 1, 2, 3	X	Standard form advance of zero, one, two, or three lines
PRTOV	YES		Automatic skip to home paper
	symbolic label	X	Address of user's overflow routine
RCFM	FIXUNB	Y	For fixed-length records
	UNDEF	Y	For undefined records
	VARUNB	Y	For variable-length records
RCSZ	(r) = general register	X	General register (2–12) that contains the length of each record for undefined records
WORK	YES	X	Process records in work area

LEGEND:

R = Required
X = Optional
Y = One option required

Table 3–5. Summary of Keyword Parameters for the DTFPR Macro Instruction

LOAD SEQUENCE/ CHARACTER	LOAD SEQUENCE/CHARACTER
1. & (ampersand)	32. 1
2. Z	33. 2
3. K	34. 3
4. J	35. 4
5. Q	36. 5
6. X	37. 6
7. V	38. 7
8. W	39. 8
9. Y	40. 9
10. P	41. * (asterisk)
11. G	42. / (virgule)
12. B	43. + (plus)
13. U	44. $ (dollar sign)
14. M	45. ((L paren.)
15. C	46.) (R paren.)
16. D	47. = (equal)
17. L	48. ' (apostrophe)
18. F	49. > (greater than)
19. H	50. < (less than)
20. S	51. ; (semicolon)
21. R	52. : (colon)
22. O	53. ¢ (cents sign) or [(left bracket)
23. A	54. | (absolute)
24. N	55. ¬ (logical not)
25. I	56. —(underline)
26. T	57. " (quote)
27. E	58. ! (excl. point) or] (right bracket)
28. . (period)	59. ? (question mark)
29. , (comma)	60. \ (reverse virgule)
30. – (minus)	61. % (percent sign)
31. 0 (zero)	62. # (number sign)
	63. @ (at sign)
	64. NP Nonprinting code (this code will be used to produce spaces on the paper)

Table 3–6. Ordering of Characters for the Load Code Sequence

CHARACTER	BINARY	HEX	CHARACTER	BINARY	HEX	CHARACTER	BINARY	HEX
(1) &	01010000	50	(22) O	11010110	D6	(43) +	01001110	4E
(2) Z	11101001	E9	(23) A	11000001	C1	(44) $	01011011	5B
(3) K	11010010	D2	(24) N	11010101	D5	(45) (	01001101	4D
(4) J	11010001	D1	(25) I	11001001	C9	(46))	01011101	5D
(5) Q	11011000	D8	(26) T	11100011	E3	(47) =	01111110	7E
(6) X	11100111	E7	(27) E	11000101	C5	(48) '	01111101	7D
(7) V	11100101	E5	(28) .	01001011	4B	(49) >	01101110	6E
(8) W	11100110	E6	(29) ,	01101011	6B	(50) <	01001100	4C
(9) Y	11101000	E8	(30) -	01100000	60	(51) ;	01011110	5E
(10) P	11010111	D7	(31) 0	11110000	F0	(52) :	01111010	7A
(11) G	11000111	C7	(32) 1	11110001	F1	(53) ¢ or [	01001010	4A
(12) B	11000010	C2	(33) 2	11110010	F2	(54) |	01001111	4F
(13) U	11100100	E4	(34) 3	11110011	F3	(55) ¬	01011111	5F
(14) M	11010100	D4	(35) 4	11110100	F4	(56) —	01101101	6D
(15) C	11000011	C3	(36) 5	11110101	F5	(57) "	01111111	7F
(16) D	11000100	C4	(37) 6	11110110	F6	(58) ! or]	01011010	5A
(17) L	11010011	D3	(38) 7	11110111	F7	(59) ?	01101111	6F
(18) F	11000110	C6	(39) 8	11111000	F8	(60) \	01101010	6A
(19) H	11001000	C8	(40) 9	11111001	F9	(61) %	01101100	6C
(20) S	11100010	E2	(41) *	01011100	5C	(62) #	01111011	7B
(21) R	11011001	D9	(42) /	01100001	61	(63) @	01111100	7C
						(64) NP	01000000	40

NOTE: Character positions 53 and 58 may be optional bracket symbols if the system is provided with an ASCII modified print drum.

Table 3–7. Standard Printer Code Conversion Table

3.2.5. Optical Document Reader

The declarative macro instruction, DTFOR, is required when the processing of optically read files is desired. It serves the UNIVAC 2703 Optical Document Reader (ODR). Following is an alphabetic listing of the required and optional keyword parameters which may appear in the operand of the DTFOR macro instruction.

A description of each keyword parameter follows the listing. A listing of the keyword parameters is given in Table 3—9 following the descriptions:

LABEL	♭OPERATION♭	OPERAND
filename	DTFOR	BKSZ=n EOFA=symbol EOFB=n EOFC=symbol FEED={300 / 600} IOA1=symbol LGTH= n MODE={CARDB / CARDT / MARKB / MARKT / OCR / OCRCARDB / OCRCARDT / OCRMARKB / OCRMARKT} RCFM={FIXUNB / UNDEF} STKR={2 / 3} [CNTRL=YES] [ERROR=symbol] [IOA2=symbol] [IORG=(r)] [OPTION=YES] [MD10=YES] [RCSZ=(r)] [ROWS={01 / 23 / 45 / 67}] [STSL=symbol] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- Block Size

  This keyword parameter specifies the length of the I/O area and has the following format:

  BKSZ=n

  where n is the size of the block in bytes. If the records in a file have an undefined format, then n specifies the size of the largest record.

  If dual reading from the optical character recognition (OCR) and the mark or card read stations is to be performed, the BKSZ must include both fields.

- Control Entry

  If a CNTRL macro instruction is to be issued by a problem program, the following keyword parameter must be specified when the file is defined:

  CNTRL=YES

  Use of this keyword parameter indicates that a CNTRL macro instruction is to be issued for the following functions:

  — Selection of output stacker (per document)

  — Rejection of document

  Whenever the CNTRL keyword parameter is specified, the STSL keyword parameter must also be specified. Refer to the STSL keyword parameter for further details.

- End-of-File Address

  The address to which control is transferred when the end-of-data sentinel is sensed must be supplied by the following keyword parameter:

  EOFA=symbol

  where symbol (label) is the address of the routine which handles end-of-data processing.

- End-of-File Sentinel Size

  This keyword parameter is required to specify the number of bytes contained in the end-of-file sentinel and has the following format:

  EOFB=n

  where n must be $\leqslant$BKSZ.

- End-of-File Sentinel Address

  This keyword parameter is required to specify the address of the sentinel which will be used for comparison against each document image to determine when the end of file has been reached. This keyword parameter has the following format:

  EOFC=symbol

  The end-of-file sentinel characters must represent the first group of characters read on a document and, therefore, must be positioned closest to the right, or leading, edge of the document.

  There should be one end-of-file sentinel document present for each file processed, plus an additional blank or insignificant document to run out the reader properly. These documents are never passed to the user as a document image.

- Major File Error

When a fatal hardware or detectable logical error occurs on a file, control will be transferred to a user's error routine if the following keyword parameter is specified:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When the logical IOCS transfers control to the routine, registers 1 through 12 are restored. Register 0 contains information concerning the reasons for the error as follows:

— Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, they will be binary zeros.

— Byte 2 contains a one-character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients (OPEN or CLOSE) and 1 = processing.

— Byte 3 contains a one-character EBCDIC value indicating the reason for the error (see Appendix E).

Transients will also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs.

Further access of the file is not permitted after a major file error has occurred.

- Document Feed Rate

The UNIVAC 2703 Optical Document Reader is available with two document processing rates. The 300 document per minute (dpm) rate is standard; the 600 dpm rate is an optional feature. The user must specify the rate at which he would like to have his documents processed. Logical IOCS will attempt to maintain that specified rate. However, since documents are processed upon user request in a possible multiprogramming environment, the maximum document rate may not be achieved. The user can select the document rate desired by the following keyword parameter options:

— FEED=300

When this keyword option is selected, the logical IOCS specifies to the hardware a document rate of 300 dpm.

— FEED=600

When this keyword option is selected, the logical IOCS assumes the 600 document per minute feature installed and specifies to the hardware a document rate of 600 dpm.

When this keyword parameter is not specified, a FEED rate of 300 dpm is assumed.

- Input/Output Area

  Each input or output file must have an area reserved for its individual use. The area must be defined by the keyword parameter:

  IOA1=symbol

  where symbol (label) is the address of the I/O area.

  Since all documents are read backward, that is, from right to left, the logical IOCS begins reading the data into IOA1+BKSZ−1. IOA1 is filled from back to front, so that after a document has been read in, it may be processed in the normal left to right manner.

  If mark sense or punch card fields are read in image mode, the I/O areas must be large enough to hold two bytes for each column of mark sense or punch card data read.

  If IOA2 is specified, the same reading conventions are applicable.

- Secondary Input/Output Area

  A secondary I/O area for standby processing may be specified by the following keyword parameter:

  IOA2=symbol

  where symbol (label) is the address of the secondary I/O area.

  Refer to keyword IOA1 for conventions regarding I/O areas.

  If IOA2 is specified, then either IORG or WORK must be specified. Refer to these keywords for details.

- Current Record Pointer

  When a general register is used to reference current data, the following keyword parameter is specified:

  IORG=(r)

  where (r) may be general registers 2 through 12. The two cases in which IORG must be specified are:

  1. when two I/O areas are used and records are not processed in a work area; and

  2. when the record format is undefined.

  In each case, logical IOCS places the address of the next available record in the specified register.

- Document Length (LGTH)

  This keyword parameter is used to specify the physical length of the documents to be processed.

  The format of the LGTH keyword parameter is:

  LGTH=n

  where n is a value from the following chart:

VALUE OF n	DOCUMENT LENGTH
3	3.00 — 3.30 inches
4	3.31 — 4.00 inches
5	4.01 — 5.90 inches
6	5.91 — 8.75 inches

  The ODR hardware requires that the document length be between 3.00 and 8.75 inches.

  When the LGTH keyword parameter is omitted, the IOCS assumes that a specification of 6 is valid.

- Modulo 10 Check Digit Verification

  To verify numeric data read by the OCR camera with this feature installed, the following keyword parameter may be specified:

  MD10=YES

  In order to perform this feature, the data must have been printed with a check digit as the leftmost or rightmost digit in the OCR reading line.

  In addition, the OCR code line must also be enclosed within two long vertical marks (lvm). Absence of either or both will cause the document to be rejected (see Table 3—10).

- Data Format Mode Selection

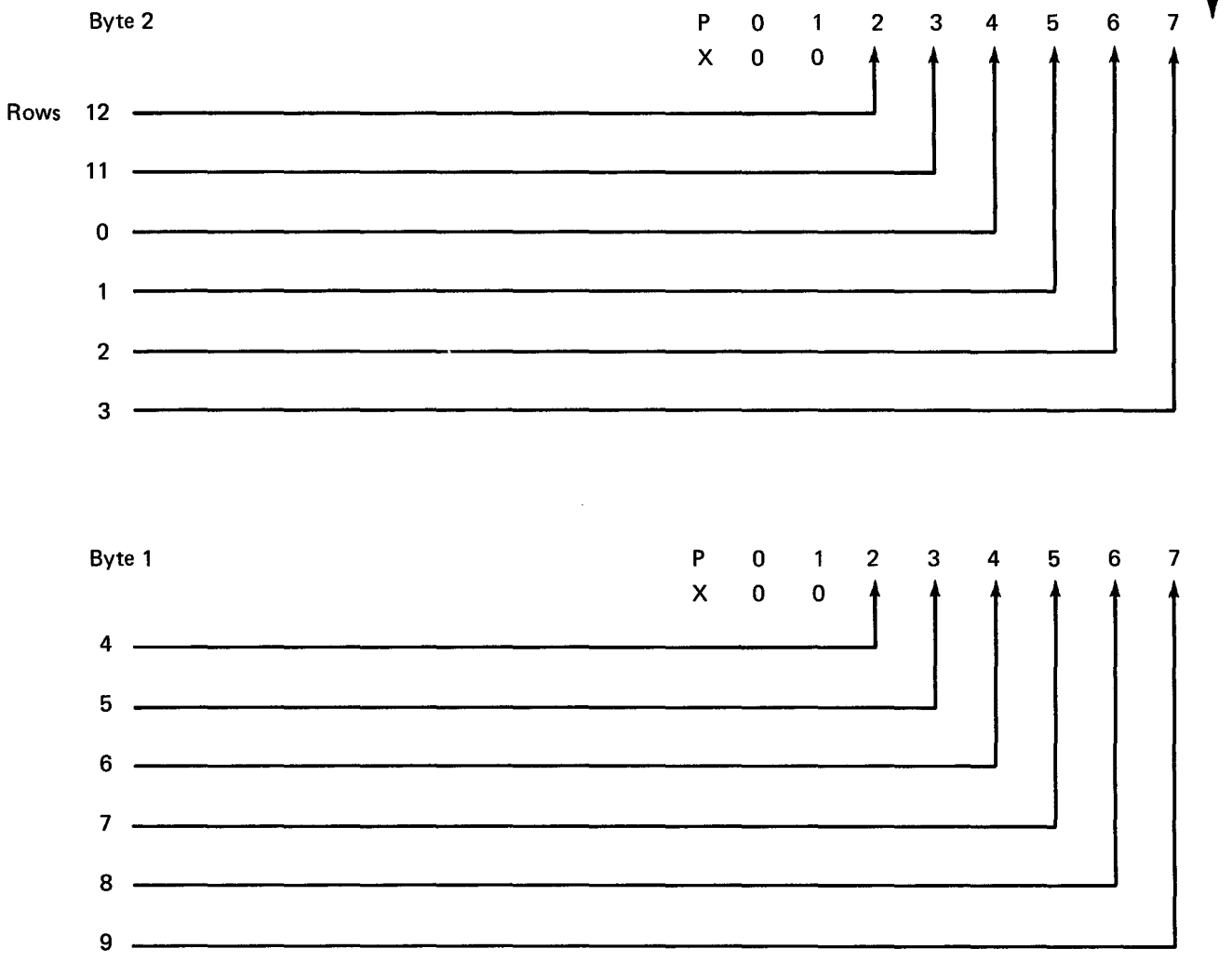
  Since the ODR will process OCR, mark sense, and punch card data fields, and the mark sense and punch card fields may be read in either translate or image mode, the following keyword parameter and its options are made available:

  MODE=CARDB

  This keyword parameter option is selected when only punch card information is to be read in image mode. Two bytes of data are read into memory for each column read on the document. The lower image of a column is transferred first (see Table 3—8).

  MODE=CARDT

  This keyword parameter option is selected when only punch card information is to be read in translate mode. Either ASCII or EBCDIC hardware translation will be performed, depending on hardware options installed.



NOTE: When selected, the mark/card reading will transfer two data bytes for each column sensed. The lower image of the last column is the first byte transferred.

Table 3–8. Image Mode Data Flow

MODE=MARKB

This keyword parameter option is selected when only mark sense information is to be read in image mode. Two bytes of data will be read for each marked column sensed on the document. The lower image of a column will be transferred first.

MODE=MARKT

This keyword parameter option is selected when only mark sense information is to be read in translate mode. Either ASCII or EBCDIC hardware translation will be made, depending on hardware options installed.

MODE=OCR

This keyword parameter option is selected when only the OCR data field is to be read from the documents. Data formats must be either USASCSOCR or UNIVAC H—14 fonts with EBCDIC or ASCII hardware translations, depending on the hardware options installed.

MODE=OCRCARDB

This keyword parameter option is selected when both OCR and punch card data is to be read and the punch card field is to be read in image mode.

MODE=OCRCARDT

This keyword parameter option is selected when both OCR and punch card data is to be read and the punch card field is to be read in translate mode.

MODE=OCRMARKB

This keyword parameter option is selected when both OCR and mark sense data is to be read and the mark sense field is to be read in image mode.

MODE=OCRMARKT

This keyword parameter option is selected when both OCR and mark sense data is to be read and the mark sense field is to be read in translate mode.

When this keyword is not specified, OCR mode is assumed.

- Optional Input File

The keyword parameter that allows the user to specify an optional input file is:

OPTION=YES

When this keyword parameter is specified and the file has not been allocated to a device by the Job Control program, the OPEN transient routine ensures that logical IOCS transfers control to the address of the user's end-of-file routine (EOFA) following the execution of the first GET macro instruction. This procedure maintains continuity in the user's program. The user is required to CLOSE the optional file.

When this keyword parameter is not specified and the file has not been allocated by the Job Control program, it is impossible to obtain records from the file. Logical IOCS transfers control to the address of the user's error routine (ERROR). If a user error routine is not supplied, the job step is aborted.

- Record Format

One of the following options describing the record format should be specified:

RCFM=FIXUNB

This form specifies fixed-length record format.

RCFM=UNDEF

This form is used for undefined record format. If the record format is undefined, either IORG or WORK must be specified. Refer to these keywords for details. If this keyword is not specified, fixed-length format is specified.

- Record Size

For files with undefined record format, the following keyword parameter is specified:

RCSZ=(r)

where (r) indicates the number (2—12) of a general register. Logical IOCS determines the size of the current record and places it in the specified register.

- Mark Read Row Selection

This keyword parameter is specified when two-row sorting is desired. There are four forms of the keyword parameter which can be used.

— ROWS=01

This form implies rows 0—1.

— ROWS=23

This form implies rows 2—3.

— ROWS=45

This form implies rows 4—5.

— ROWS=67

This form implies rows 6—7.

If this parameter is specified, STKR=2 must also be specified. The logical IOCS will direct all documents to stacker 2 except those documents with a mark present in the selected rows of the rightmost column of the mark/read field.

Whenever this keyword parameter is specified, it is assumed stacker 2 is selected. Logical IOCS cannot support this feature unless the optional hardware is installed.

- Stacker Mode Selection

  Output stacker selection can be preset by specifying one of the following keyword parameters and its options:

  STKR=2

  STKR=3

  All documents read without error will be prerouted to the output stacker specified. However, when mark read row selection has been specified, the output stacker selected is dictated by the data read.

  All documents read in error are routed to the reject stacker 1 and any erroneous document images present will not be made available to the user.

- Program Controlled Stacker Selection

  If the user wishes to select stackers on an individual document basis, he must specify the STSL keyword in addition to the CNTRL keyword. The STSL keyword parameter has the following format:

  STSL=symbol

  where symbol is the address of the first instruction of the user's stacker select subroutine.

  The user's STSL routine is incorporated into the GET processing. After the document is read in, program control is passed to the STSL routine. Upon entering the STSL routine, registers 2—12 appear as they did when the GET was issued, except that the IORG and RCSZ register, if specified, have been set. The user must not process the document at this time other than to determine the output stacker desired and to issue the appropriate CNTRL macro (see Table 3—7). The STSL routine must not alter registers 0, 1, 13, 14, 15 nor the register save area. Alterations to registers 2—12 will be preserved.

  After the CNTRL macro is issued, program control must be returned to GET by executing the following branch instruction:

  B 4(0,15)

- Work Area Processing

  The logical IOCS makes the current record available to the user in a work area if the following keyword parameter is specified:

  WORK=YES

  The address of the current work area must be specified with each GET macro instruction.

```
    LABEL    OPERATION  OPERAND                                  72
    * DOCUMENT READER FILE DEFINITION - SAMPLE

1.  DOCIN    DTFOR      BKSZ=60,                                 X
2.                      MODE=MARKB,                              X
3.                      EOFA=DONE,EOFB=2,EOFC=EOAIC=EOFCHAR,     X
4.                      IOA1=BUF1,                               X
5.                      IOA2=BUF2,                               X
6.                      WORK=YES,                                X
7.                      LGTH=3,                                  X
8.                      FEED=600,                                X
9.                      STKR=2,                                  X
10.                     ROWS=45,                                 X
11.                     ERROR=CANJOB
```

1. The label DOCIN names this file. All references to this file (e.g. GET, CNTRL) must specify this filename. This filename must also appear in the Job Control Stream (//LFD). The block size is 60 bytes. Each I/O area is 60 bytes long.

2. The data is to be read in mark sense — image mode (see Table 3–8). Since each column is read into 2 bytes and the block size is 60, there can be only 30 columns.

3. When the right or leading column of the last document is read in, it will produce a bit configuration exactly like the 2 bytes stored at EOFCHAR. When Logical IOCS detects this configuration, it will transfer control to the routine at DONE.

4,5,6. Two I/O areas are used to allow input-processing overlap. The documents are then moved to a work area the address of which is specified for each GET.

7. The documents are between 3 and 3.2 inches long.

8. The documents are to be read in fast feed mode.

9,10. With the following exceptions, all documents are directed to stacker 2.

    a. All documents read in error are directed to the reject stacker.

    b. All documents with a mark in row 4 or 5 of the right or leading column will be directed to stacker 3.

11. If Logical IOCS detects any logic errors, or if an unrecoverable hardware error occurs, control is transferred to the routine at CANJOB.

KEYWORD	SPECIFICATION	FILE INPUT	REMARKS
BKSZ	n=maximum block size	R	The maximum block size, in bytes
CNTRL	YES	X	A CNTRL macro will be issued for the file
EOFA	symbolic label	R	Identifies end of file routine
EOFB	n	R	Specifies length of end of file sentinel
EOFC	symbolic label	R	Address of end of file sentinel
ERROR	symbolic label	X	Address of user's unrecoverable error routine
FEED	300	Y	Specifies document feed rate of 300 dpm
	600	Y	Specifies document feed rate of 600 dpm
IOA1	symbolic label	R	Address of input area
IOA2	symbolic label	X	Address of alternate input area
IORG	(r)=general register (2–12)	X	Required when two I/O areas are used without a work area or when processing undefined records without a work area
LGTH	n	Y	Specifies the physical size of the documents
MD10	YES	X	To verify correctness of numeric data read by OCR
MODE	CARDB	Y	Specifies punched card data is to be read in image mode
	CARDT	Y	Specifies punched card data is to be read in translate mode
	MARKB	Y	Specifies mark sense data is to be read in image mode
	MARKT	Y	Specifies mark sense data is to be read in translate mode
	OCR	Y	Specifies OCR data is to be read
	OCRCARDB	Y	Specifies OCR with punched card data read in image mode
	OCRCARDT	Y	Specifies OCR with punched card data read in translate mode
	OCRMARKB	Y	Specifies OCR with mark sense data read in image mode
	OCRMARKT	Y	Specifies OCR with mark sense data read in translate mode
OPTION	YES	X	Specifies an optional file
RCFM	FIXUNB	Y	For fixed-length records
	UNDEF	Y	For undefined records
RCSZ	(r)=general register (2–12)	X	For undefined records, register contains record size

LEGEND:

R = Required
X = Optional
Y = One option required

Table 3–9. Summary of Keyword Parameters for the DTFOR Macro Instruction (Part 1 of 2)

KEYWORD	SPECIFICATION	FILE INPUT	REMARKS
ROWS	01	X	For mark read row selection — rows 0 and 1
	23	X	For mark read row selection — rows 2 and 3
	45	X	For mark read row selection — rows 4 and 5
	67	X	For mark read row selection — rows 6 and 7
STKR	2	X	Specifies output stacker 2
	3	X	Specifies output stacker 3
STSL	symbolic label	X	Address of the first instruction of the user stacker select subroutine for selecting stackers on an individual document basis
WORK	YES	X	For processing records in a work area

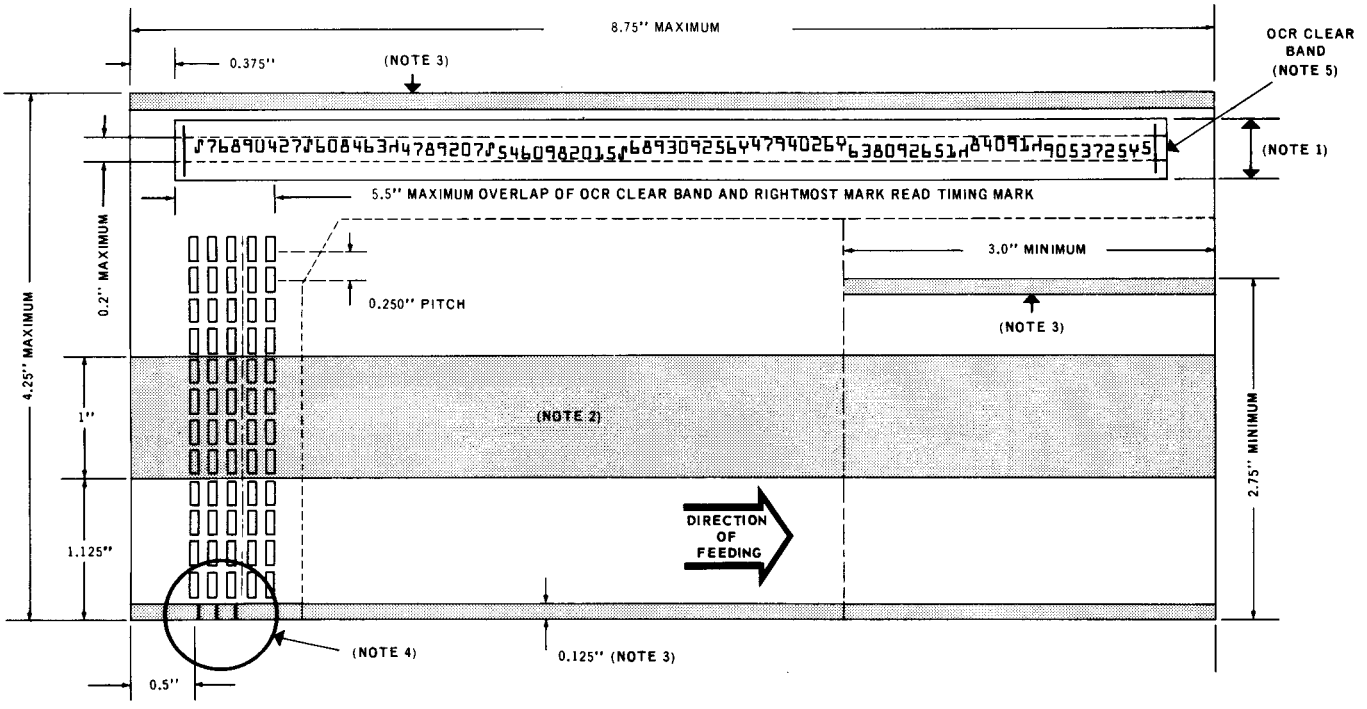
LEGEND:

R = Required
X = Optional
Y = One option required

Table 3–9. Summary of Keyword Parameters for the DTFOR Macro Instruction (Part 2 of 2)

3.2.5.1. Document Format

The format for documents used with the ODR is shown in Figure 3—1.



NOTES:

1. 3/8-inch minimum width; 1/2-inch width recommended.
2. No part of the OCR clear band may be in this one-inch wide area.
3. Edge tear zones. OCR clear band not recommended in this area.
4. Timing marks.
5. The clear band is a horizontal band, a minimum of .38 inch in width, within which is centered the character line. The clear band may be located anywhere on the document except as noted. It is recommended that the clear band extend to the right edge (leading edge) of the document and have a 1/2-inch width.

   The printing area is a horizontal band .20 inch in width in the center of the clear band in which all characters to be recognized must be printed. The printing area must terminate at least .38 inch from both right and left edges of the document.

   The horizontal character spacing (distance between adjacent character centerlines) must not be less than .09 inch and the average must not be less than .10 inch.

   All columns to be mark read are defined by the presence of pre-printed black timing or "tic" marks, located on the bottom of the document and immediately adjacent to the leading edge of the columns they identify. A maximum of twelve locations per column is permitted.

   Whenever both the OCR reading and the mark/card reading are selected, it is necessary to define an acceptable document format such that ambiguous data transfer cannot occur. The OCR code line on a punched card must not extend more than 5.50 inches from leading (right) edge. The OCR code line on a mark read document must not extend more than 5.50 inches to the left of the rightmost timing mark.

Figure 3—1. Document Format

3.2.5.2. Typical Application

A typical application of the ODR's capabilities is shown in Figure 3—2.

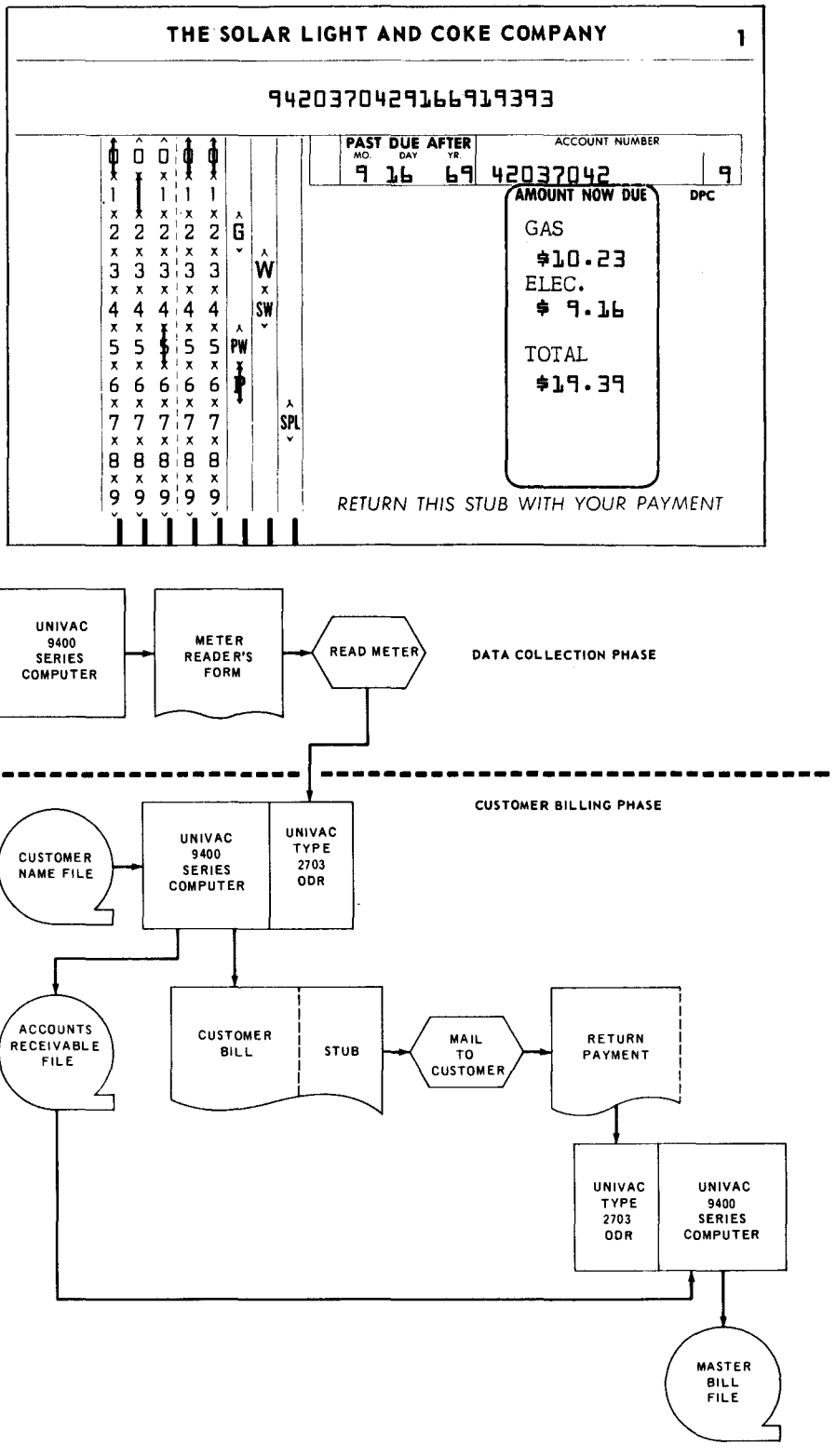


Figure 3—2. A Typical Public Utility Application

3.2.5.3. Font/Code Relationship

Table 3—10 lists the font/code relationship of the United States of America Standard Character Set for Optical Character Recognition (USASCSOCR) or the UNIVAC H—14 and the Extended Binary Coded Decimal Interchange Code (EBCDIC) or the American National Standard Code for Information Interchange (ASCII). Either font and code may be selected for use with the ODR.

USASCSOCR CHARACTER	H—14 CHARACTER	EBCDIC ENCODING	ASCII ENCODING
0	0	F0	30
1	1	F1	31
2	2	F2	32
3	3	F3	33
4	4	F4	34
5	5	F5	35
6	6	F6	36
7	7	F7	37
8	8	F8	38
9	9	F9	39
⌐	N	4C	3C
¬	F	7E	3D
⌐	\	6E	3E
|*	|*	4F	21
Multiple**	Multiple**	6F	3F

*The long vertical mark (lvm) is used to define a field within a scan line. The lvm's are printed by the computer when the forms are prepared. When the lvm is read by the ODR, the selected encoding is sent to the computer.

**This word is not printed by the computer or read by the ODR. The respective code, depending on the encoding selected, is sent to the computer when the ODR cannot recognize a character.

Table 3—10. Font/Code Relationship

3.2.5.4. Modulo 10 Check Digit Verification

This feature verifies the correctness of the numeric optical reading. If the feature is installed and the MD10=YES keyword parameter is included in the DTFOR macro instruction, the internal hardware of the ODR performs a weighted addition of the digits read from the documents. To perform this test, the data is printed with a check digit. The check digit is program-calculated and is printed as the rightmost digit in the optical character reading line. The check digit can be calculated as follows:

1. Multiply the units position and every alternate position of the basic number by two.
2. Cross add the digits in the products and the digits in the basic number not multiplied by two.
3. Subtract the cross-footed total from the next higher number ending in zero.
4. The difference is the check digit.

Example:

Basic account number to be printed:			7		3		0		7		4	
Units and every alternate position:			7				0				4	
Multiply by two:			2				2				2	
Product:			14				0				8	
Digits not multiplied by two:					3				7			
Cross-add:	1	+	4	+	3	+	0	+	7	+	8	= 23
Next higher number ending in zero:												30
Subtract crossfooted total:												− 23
Check digit:												7
Self-checking number:			7		3		0		7		4	7

When the basic account number and the check digit are read by the ODR, a similar calculation is made in reverse to assure that the weighted sum of the digits is a multiple of 10. In the example, the correct sum is 30.

If the sum is not a multiple of 10, a digit was read incorrectly. An error indicator is set and the document goes to the reject stacker (stacker number 1).

3.2.6. Paper Tape

The declarative macro instruction, DTFPT, is required for input and output paper tape files. Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFPT macro instruction.

A description of each keyword parameter follows the listing. A summary of the keyword parameters is given in Table 3—11 following the descriptions.

LABEL	↑ OPERATION ↑	OPERAND
filename	DTFPT	BKSZ=n EOFA=symbol IOA1=symbol [EOR=expression] [ERRO={IGNORE | SKIP | symbol}] [ERROR=symbol] [FSCAN=symbol] [FTRANS=symbol] [IOA2=symbol] [IORG=(r)] [LSCAN=symbol] [LTRANS=symbol] [MODE={BINARY | STD}] [OBKS=n] [OPTION=YES] [RCFM={FIXUNB | UNDEF}] [RCSZ=(r)] [SCAN=symbol] [TRANS=symbol] [TYPE={INPUT | OUTPUT}] [WORK=YES]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- Block Size

  This keyword parameter is required for both input and output files. The format is:

  BKSZ=n

  where n (decimal) is the size of the block in bytes. For undefined record format files, the size must be that of the largest record and must include the end-of-record character, on both input and output. For shifted output files of undefined length records, the size must include any figure or letter shift characters which might be added into the record.

  The maximum size of a block on paper tape is 4095 bytes.

- End of an Input File

  The address to which control is transferred when the end-of-tape condition is sensed must be specified by the following keyword parameter:

  EOFA=symbol

  where symbol (label) is the address of the user's end-of-tape routine. This is required for input files.

The end-of-tape routine may close the file, or if desired, cause another tape to be mounted and return to logical IOCS via register 14. In the latter case, IOCS reads in the next record and delivers it to the GET macro call which encountered the end-of-tape condition. If any other logical IOCS imperative macros are called, register 14 and the save area must be saved and restored.

- End-of-Record Character

The value of the user-defined end-of-record character must be specified by the following keyword parameter:

EOR=expression

This parameter must be used for output files with undefined record format only. Any self-defining term, or a symbol which has been equated to the desired value, may be used here. Logical IOCS appends this character to the undefined record before writing it.

- Unique File Errors

This parameter applies only to input files. A unique error is an unrecoverable parity error. If a unique error occurs when reading a block of data, the user has the following options: he may process the block as if no error had occurred, bypass the block, enter a special routine for individually processing error blocks, or terminate the job. The absence of this parameter specification causes the logical IOCS to assume that a job should be terminated if a unique error occurs. The other options are specified as follows:

— ERRO=IGNORE

This option permits the images in the I/O area to be available to the user as though no error had occurred. If the mode is not binary, the most significant bit of any character with a parity error is set to a 1.

— ERRO=SKIP

This option causes the block in error to be bypassed and the next error-free block to be accessed.

— ERRO=symbol

For this option, symbol (label) is the address of a user's error routine to process any errors. When logical IOCS transfers control to the user's error routine, register 14 contains the normal return address. Registers 2 through 12 are restored. Register 1 contains the address of the block in question for both input and output files. In addition, the following rules are observed:

1. Do not destroy the present register save area (pointed to by register 13).
2. Return with register 13 containing the save area address.
3. Do not issue a GET or PUT instruction on the file.
4. Access the record by means of register 1.
5. Use another save area for any other IOCS file.
6. Return by transferring control to the address contained in register 14.

- Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the error address from a processing macro instruction (such as GET and PUT), the address of the instruction following the macro call can be found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN and CLOSE). Register 0 contains the following information concerning the reasons for the error:

- Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary zeros.

- Byte 2 contains a one-character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.

- Byte 3 contains a one-character EBCDIC value indicating the reason for the error (see Appendix E).

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs, but the console typeout still appears.

Further access of the file is not permitted after a major file error has occurred.

One exception to the above holds — an end-of-record character error for an input file with undefined record format. If an incoming record fills the input area before an end-of-record character is detected (by the hardware), an end-of-record character error is assumed. In this case, logical IOCS will allow the user program to return from the error routine via register 14. The same precautions listed under ERRO=symbol must be followed. Note, however, that register 1 does not point to the record and the record is not available to the user program. If the error routine returns to logical IOCS via register 14, the next record is read.

NOTE: To avoid an EOR error, the input area (as described by BKSZ) must be at least one byte longer than longest record anticipated.

- Shifted Code for Output Files — Figure Scan

To handle shifting codes for output files, the keyword parameter LSCAN and the following keyword must be specified:

FSCAN=symbol

where symbol (label) is the address of a scan table in the problem program used to define characters to be treated as figures. The table is indexed by the characters in the output data.

Entries corresponding to letters must contain the code for the letter shift. All other entries must contain a byte of zero.

- Shifted Code for Input Files — Figure Translation

  To handle shifting codes for input files, the keyword parameters SCAN and LTRANS and the following keyword parameter must be specified:

  FTRANS=symbol

  where symbol (label) is the address of a translation table in the problem program for those characters following a figure shift character.

  NOTE: When FTRANS and LTRANS are specified, the keyword TRANS must not be specified.

- Input/Output Area

  Each input or output file must have an area reserved for its individual use. The area must be defined by the following keyword parameter:

  IOA1=symbol

  where symbol (label) is the address of the I/O area. The length of the area is specified by the keyword parameter BKSZ for all but files with shifted code and fixed-unblocked records. For the latter type of file, the keyword parameter OBKS specifies the length.

- Secondary Input/Output Area

  A secondary I/O area for standby processing improves efficiency by overlapping I/O and record processing time. The format of the keyword parameter for a secondary I/O area is:

  IOA2=symbol

  where symbol (label) is the address of the secondary I/O area. It is subject to the same requirements as noted in IOA1.

- Current Record Pointer

  When a general register is used as an index register to reference current data, the following keyword parameter should be specified:

  IORG=(r)

  where (r) is the number of the general register (2—12). If a work area is not required, this keyword parameter must be specified when there are two I/O areas.

  For input files, logical IOCS places the address of the next available record in the specified register.

  For output files, logical IOCS places in the specified register the address of the next available I/O output area.

- Shifted Code for Output Files — Letter Scan

  To handle shifting codes for output files, the keyword parameter FSCAN and the following keyword must be specified:

  LSCAN=symbol

  where symbol (label) is the address of a scan table in the problem program used to define characters to be treated as letters. The table is indexed by the characters in the output data. Entries corresponding to figures must contain the code for the figure shift; all other entries must contain a byte of zero.

- Shifted Code for Input Files — Letter Translation

  To handle shifting codes for input files, the keyword parameters SCAN and FTRANS and the following keyword must be specified:

  LTRANS=symbol

  where symbol (label) is the address of a translation table in the problem program for those characters following a letter shift character.

  NOTE: When FTRANS and LTRANS are specified, the keyword parameter TRANS must not be specified.

- Input/Output Mode

  This keyword parameter is used to specify the input/output mode of operation of the paper tape subsystem. There are two forms of specification of this keyword.

  — MODE=BINARY

  This form is used to suppress character recognition and bypass parity. All eight bits of data are read into the I/O area from the reader on input and eight bits of data from the I/O area are punched on output. The program connector must be wired to suppress parity for MODE=BINARY to be valid.

  — MODE=STD

  This form is used to allow character recognition and is the default if MODE is not specified.

- Over-Block Size

  The following keyword parameter specifies the size of the input/output area for fixed unblocked files with shifted codes:

  OBKS=n

  where n is a decimal number not exceeding 4095. If this parameter is not specified, then BKSZ is used.

  For input files, OBKS specifies the number of characters to be read in, before compression and translation, to produce the number of characters specified by BKSZ. If compression causes a reduction below BKSZ length, then more characters are read to fill the I/O area to OBKS and compression and translation continues.

For output files, OBKS specifies the maximum record size. If OBKS is large enough to allow the insertion of all the shift characters required to build the output record, a single write operation results from a PUT macro. Otherwise, several output operations may result from a single PUT call.

- Optional Input File

  The keyword parameter to allow the user to specify an optional input file follows:

  OPTION=YES

  When this keyword parameter is specified and the file has not been allocated to a device by Job Control, the OPEN transient routine ensures that logical IOCS will transfer control to the address of the user's end-of-file routine (EOFA) following the execution of the first GET macro instruction. This procedure maintains continuity in the user's program. The user is required to close the optional file.

  When this keyword parameter is not specified and the file has not been allocated by Job Control, it is impossible to obtain records from the file, if one exists. Logical IOCS transfers control to the address of the user's error routine (ERROR). If a user error routine is not supplied, the job step is aborted.

- Record Format

  The record format of a file is specified by a keyword parameter. There are two options to indicate whether the records are fixed-length unblocked, or undefined. The options are coded as follows:

  — RCFM=FIXUNB Fixed-length, unblocked

  — RCFM=UNDEF Undefined

  Logical IOCS assumes a fixed-length, unblocked record format when this keyword parameter is not specified.

- Record Size

  The number of bytes in a record is made available to logical IOCS by this keyword parameter. Logical IOCS assumes that the record size is equal to the block size for fixed-length, unblocked records.

  — RCSZ=(r)

  This form of the keyword parameter is required for output files with an undefined record format. The general register (2—12) specified by (r) contains the block length. The keyword parameter may also be used for input files. Logical IOCS places the block length into the general register specified by (r).

- SCAN Table

  This keyword parameter is required for input files with shifted code, along with FTRANS and LTRANS. It can also be specified for files which may have certain characters to be deleted before processing starts. The format is:

  SCAN=symbol

where symbol (label) is the address of a scan table in the problem program. The entries in this table are accessed by using the input data as index.

There are three nonzero entries possible:

X'04' — defines the figure shift character. All data following up to the next shift character is translated via FTRANS.

X'08' — defines the letter shift character. All data following up to the next shift character is translated via LTRANS.

X'0C' — defines a 'delete' character. The corresponding data character is deleted from the record, following characters moved one byte to the left.

There must not be any 04 or 08 entries unless FTRANS and LTRANS are specified. All other entries in the SCAN table must be 00.

- Translate Table

This keyword parameter defines a table to be used on the data via the translate (TR) instruction. The format is:

TRANS=symbol

where symbol (label) is the address of the translation table in the problem program.

For an input file, translation is performed after any deletions that may be required. TRANS cannot be specified for an input file with shifted code.

For an output file, translation is performed after any insertion of shift characters. The shift characters, however, are not translated.

- Type of File

The type of file is specified by this parameter:

— TYPE=INPUT

This keyword parameter specifies an input file (one that is to be read). This option is assumed if the parameter is not specified.

— TYPE=OUTPUT

This keyword parameter specifies an output file (one that is to be written).

- Work Area Processing

If records in an input file are to be transferred from the input I/O area to a work area, the following keyword parameter must be specified:

WORK=YES

This keyword parameter must also be specified for output files when records are to be built in a work area.

The user must specify the address of the current work area with each GET and PUT macro instruction.

```
LABEL     OPERATION  OPERAND                                              72
*PAPER TAPE INPUT FILE WITH SHIFTED CODES - SAMPLE

TAPEIN    DTFPT      BKSZ=120,                                            X
                     EOFA=ENDF,                                           X
                     FTRANS=FIGCONV,                                      X
                     IOA1=INA1,                                           X
                     IOA2=INA2,                                           X
                     LTRANS=LETCONV,                                      X
                     MODE=BINARY,                                         X
                     OBKS=128,                                            X
                     RCFM=FIXUNB,                                         X
                     SCAN=BREAKTAB,                                       X
                     TYPE=INPUT,                                          X
                     WORKS=YES

*PAPER TAPE OUTPUT FILE - SAMPLE

TAPEOUT   DTFPT      BKSZ=121,                                            X
                     EOR=X'45',                                           X
                     ERROR=USERROR,                                       X
                     IOA1=OUT1,                                           X
                     IOA2=OUT2,                                           X
                     IORG=(5),                                            X
                     RCFM=UNDEF,                                          X
                     RCSZ=(4),                                            X
                     TRANS=CONVERT,                                       X
                     TYPE=OUTPUT
```

KEY-WORD	SPECIFICATION	FILES INPUT	FILES OUTPUT	REMARKS
BKSZ	n = maximum block size	R	R	The maximum block size, in bytes
EOFA	symbolic label	R		Identifies EOF routine
EOR	expression		X	Defines end-of-record character – required for UNDEF output
ERRO	IGNORE	X		Treat error block in input file normally
	SKIP	X		Skip the block containing the error
	symbolic label	X		Address of user routine to process unique errors
ERROR	symbolic label	X	X	Address of user's unrecoverable error routine

LEGEND:

R = Required
X = Optional
Y = One option required
† = Assumed parameter, if none specified

Table 3–11. Summary of Keyword Parameters for the DTFPT Macro Instruction (Part 1 of 2)

KEY WORD	SPECIFICATION	FILES		REMARKS
		INPUT	OUTPUT	
FSCAN	symbolic label		X	Address of figure scan table
FTRANS	symbolic label	X		Address of figure translation table
IOA1	symbolic label	R	R	Address of input/output area
IOA2	symbolic label	X	X	Address of alternate input/output area
IORG	(r) = general register	X	X	Required if records are processed in the I/O area and there are two I/O areas
LSCAN	symbolic label		X	Address of letter scan table
LTRANS	symbolic label	X		Address of letter translation table
MODE	BINARY	Y	Y	Suppress character recognition or parity generation
	STD†	X	X	Character recognition or parity generation effective
OBKS	n = maximum I/O area size	X	X	Maximum I/O area size for shifted code
OPTION	YES	X		For specifying an optional file
RCFM	FIXUNB†	X	X	For fixed-length, unblocked records; assumed
	UNDEF	Y	Y	For undefined records
RCSZ	(r) = general register		X	For undefined output records; register contains record size
		X		Optional for input file; register contains record size
SCAN	symbolic label	X		Address of scan table for input files
TRANS	symbolic label	X	X	Address of translation table
TYPE	INPUT†	X		For input files; assumed
	OUTPUT		R	For output files
WORK	YES	X	X	Process records in work area

LEGEND:

R = Required
X = Optional
Y = One option required
† = Assumed parameter, if none specified

Table 3–11. Summary of Keyword Parameters for the DTFPT Macro Instruction (Part 2 of 2)

3.3. IMPERATIVE MACRO INSTRUCTIONS

The imperative macro instructions supplied for sequential accessing are OPEN, GET, PUT, RELSE, TRUNC, CNTRL, LBRET, FEOV, and CLOSE. Not all macro instructions are available for use on all devices that can be accessed sequentially. Some are specifically input type macro instructions and cannot be used for an output device; the opposite is true, also. When a parameter applies only to specific devices, attention is called to this fact in the accompanying description.

3.3.1. OPEN Macro Instruction

Before a file can be accessed by logical IOCS, an OPEN macro instruction is issued. The transient routine called by the OPEN macro instruction performs certain validation checks and initiates file processing.

A check is made to determine if all the necessary keyword parameters defining the file have been supplied. The device allocation performed by the Job Control program is determined.

The actions performed by the OPEN transient routine depend on whether the file is an input or an output file. For input files, the first data record is not available to the user until a GET macro instruction is issued. For output files no data is written; however, the data area is made available for use. The ability to reopen a file is also provided.

For magnetic tape and sequential disc files, the job stream is accessed to obtain the necessary label and volume information. Label validation is carried out. The keyword parameter FLBL of the DTFMT macro instruction determines the validation procedures for magnetic tape. Since standard labels are required on discs, the validation performed is not variable. User label processing for both tape and disc files is specified by the LBAD keyword parameter.

The format of the OPEN macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	OPEN	{ filename filename-1, filename-2,...,filename-n (1) }

POSITIONAL PARAMETER 1

filename – is the label(s) of the corresponding DTF (define the file) macro instruction(s) in the program. The filename may have a maximum of seven characters.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:

```
     LABEL      ♭ OPERATION ♭          OPERAND                    ♭
1                10          16
1.               OPEN        INPUT,OUTPUT,,LISTING
```

1. Enters the transient routines necessary to prepare the DTF macro instructions whose labels are INPUT, OUTPUT, and LISTING. Checks that they are prepared to access these files with the next imperative macro instruction (GET, PUT, and so forth).

3.3.2. GET Macro Instruction

This macro instruction causes the next logical record in an input file to become available to the user. The data is accessible either in the I/O area or in a user specified work area. The macro instruction is used for all record types.

Users that specify only one I/O area and have neither blocked records nor variable-length tape records to be read backwards may directly access data relative to the name of the I/O area. All other users must specify a register (by means of the IORG keyword parameter) to be used by logical IOCS to give the starting address of the current record or must specify a work area in the declarative macro instruction. More than one work area may be employed, since the address of the area is specified to logical IOCS with each GET macro instruction. Each GET macro instruction may specify a different work area, if necessary.

The format of the GET macro instruction is:

LABEL	↹ OPERATION ↹	OPERAND
[name]	GET	$\left\{\begin{matrix}\text{filename}\\(1)\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{workarea}\\(0)\end{matrix}\right\}\right]$

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

workarea – is the label of an area into which the current record is moved for processing.

(0) – indicates that register 0 has been preloaded with the address of a work area.

if blank – indicates the user has chosen processing either by means of a register (IORG keyword parameter) or by directly accessing the data relative to the name of the I/O area.

NOTE: When the work area is specified, the keyword parameter WORK=YES must be present in the DTF statement.

Example:

	LABEL (1)	↹ OPERATION ↹ (10)	OPERAND (16)	↹
1.	HERE	GET	INPUT,INWORK	

1. Places the next record of the file described in the DTF macro instruction, whose label is INPUT, into the area whose label is INWORK. The optional label HERE may be used to reference this point in the program.

3.3.3. PUT Macro Instruction

This macro instruction causes an output record to be delivered to logical IOCS in either the output I/O area or a user specified work area. The record is then no longer available to the user.

Depending on the record format (RCFM keyword parameter), logical IOCS groups the output records or delivers the records singly to the output peripheral device. A general register (2–12) must be supplied (by means of the IORG keyword parameter) when the records are blocked or when a standby area (IOA2) is specified and when no work area is used. This provides logical IOCS with a place to put the address of the current output area. Unblocked records processed in an I/O area can be referenced directly by means of the name given to that area (IOA1) by the user.

The output I/O area(s) is not cleared after the current output data is sent to the device. Care should be exercised to clear the area before use or to supply complete records, including blanks, to logical IOCS to prevent unwanted information from appearing in the data.

When records are processed in a work area, logical IOCS moves the record into the I/O area. This frees the work area for subsequent processing by the user.

The format of the PUT macro instruction is:

LABEL	OPERATION	OPERAND
[name]	PUT	{filename / (1)} [, {workarea / (0)}]

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

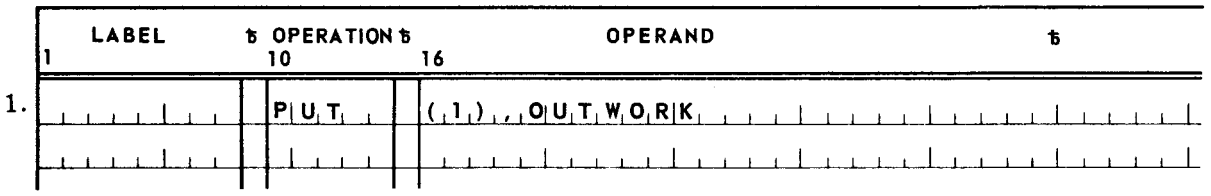
workarea – is the label of the work area from which the record may be obtained.

(0) – indicates that register 0 has been preloaded with the address of the work area.

if blank – indicates the user has chosen processing either by means of a register (IORG keyword parameter) or by directly accessing the data relative to the name of the I/O area.

NOTE: When the work area is specified, the keyword parameter WORK=YES must be present in the DTF statement.

Example:



1. Places the record in the area whose label is OUTWORK, as the next output record of the file described in the DTF macro instruction whose label address has been loaded into register 1.

3.3.3.1. Variable-Length, Blocked Records

The user must determine the size of the output record and must include the size at the beginning of the record before issuing the PUT macro instruction. Logical IOCS calculates the block size and enters it in the block before writing. The user may not access the first four bytes which are reserved for block size.

The user is reminded that for sequential disc output files, an eight-byte count field is included in the I/O area. These bytes precede the four bytes allocated for the block size and are not available to the user for data.

If the record is built in a work area when a PUT macro instruction is issued, logical IOCS determines whether this record fits in the I/O area. If it does not fit in the I/O area, logical IOCS writes the block, then starts a new block with the current record. If the record fits, it is added to the current block being built.

If the records are built directly in the I/O area, the user must determine whether the current record fits the area *before* forming the record. Logical IOCS supplies the amount of residual space after each PUT operation in the register specified by the VBLD keyword parameter. Should the residual space be inadequate, a TRUNC macro instruction (see 3.3.5) must be employed. Then the user has a full block area available for forming the record.

3.3.3.2. Undefined Records

Undefined records are assumed by logical IOCS to be unblocked. The size of these blocks must be given to logical IOCS in the register specified by the RCSZ keyword parameter. This is expected with each PUT macro instruction.

3.3.4. RELSE Macro Instruction

This macro instruction is used with blocked input records from tape or direct access storage devices. It permits the user to skip the remaining records in the current block. The next GET macro instruction makes the first record of the succeeding block available.

The format of the RELSE macro instruction is:

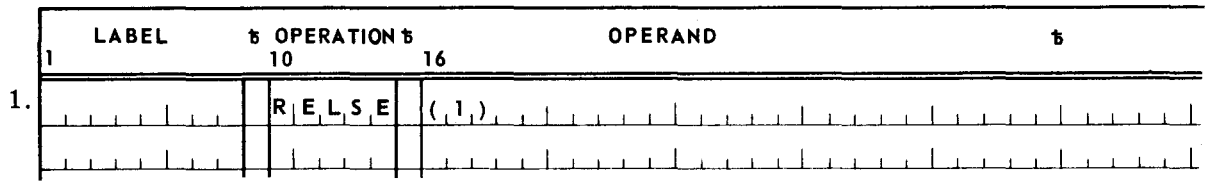
LABEL	OPERATION	OPERAND
[name]	RELSE	{ filename / (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:



1. Causes the next GET macro instruction to ignore any additional records in the block currently being processed in the file described by the DTF macro instruction whose label address is in register 1. The next GET macro instruction for this file will make available the first record of the next sequential block.

3.3.5. TRUNC Macro Instruction

This macro instruction is used with blocked output records for tape or direct access storage devices. It permits the user to write short blocks of output data. The starting address of the next available output I/O area is the current area address after the TRUNC macro instruction is used, and TRUNC resets the IORG register to point to this area.

The TRUNC macro instruction must be used with variable-length, blocked records that are built in the output area to indicate to logical IOCS that the block is to terminate at this point. When a PUT macro instruction is issued after a variable-length record has been built, logical IOCS supplies the user with the number of bytes remaining in the output area in the general register specified by the keyword parameter VBLD. If the user determines that the next variable-length record does not fit, a TRUNC macro instruction must be issued to write the output block. The entire output area is then available, and the number of available bytes is in the VBLD register.

The format of the TRUNC macro instruction is:

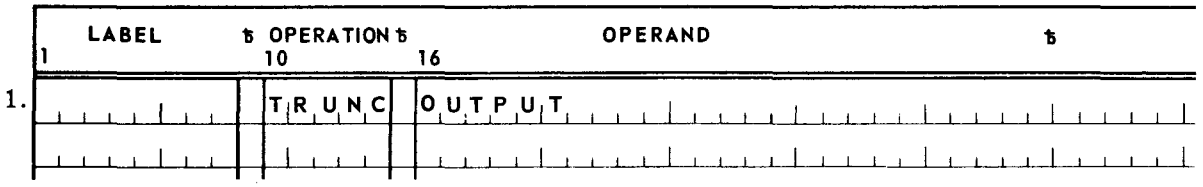
LABEL	OPERATION	OPERAND
[name]	TRUNC	{ filename / (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:



1. Sends to the output device the block of records accumulated by PUT macro instructions since the last TRUNC was issued or since a full block of records was sent automatically to the output device for the file described in the DTF macro instruction whose label is OUTPUT.

3.3.6. CNTRL Macro Instruction

This macro instruction provides commands for direct access storage devices, magnetic tape units, card punches, and printers. These commands refer to operations such as seeking to the next block of records, rewinding tape, card stacker selection, and spacing or skipping on the printer.

In the case of a direct access storage device, logical IOCS issues a seek to the current track being processed.

The keyword parameter CNTRL=YES must be included in the declarative macro instruction for that file whenever the CNTRL macro instruction is to be issued for the file by the problem program. The DTFMT macro instruction does not require this keyword parameter specification.

The third and fourth positional parameters are used with a printer file. Either m or n may be omitted; however, specifying both m and n causes spacing or skipping to take place before and after printing the line. (Execution time is minimized if forms are positioned after printing.) The specification of n for paper advancement after printing overrides the normal space advance specified by the printer advance keyword parameter (PRAD=n) for one print line only. If multiple CNTRL macro instructions for paper advance after printing are issued between the execution of successive PUT macro instructions, only the last CNTRL macro instruction is executed.

The format of the CNTRL macro instruction is:

LABEL	OPERATION	OPERAND
[name]	CNTRL	{filename / (1)},code [,{m / n}] [,n]

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

code – is the mnemonic code of the command to be performed (see Table 3–12).

POSITIONAL PARAMETER 3

m – is the immediate carriage control on the printer; spacing or skipping takes place before printing.

n – specify 2 to select stacker 2; 3 to select stacker 3. If blank, document goes to reject stacker 1.

POSITIONAL PARAMETER 4

n – applies to delayed skipping or spacing; spacing or skipping takes place after printing.

Examples:

```
LABEL      OPERATION      OPERAND                          COMMENTS
1          10             16
           CNTRL          ACCNTS,SEEK
           CNTRL          PRINT2,SK,,15
           CNTRL          {filename}(1),SS,2               REJECT DOCUMENT
           CNTRL          {filename}(1),SS,2               SELECT STACKER 2
           CNTRL          {filename}(1),SS,3               SELECT STACKER 3
```

I/O UNIT	MNEMONIC CODE	PRINTER CONTROL		DOCUMENT CONTROL	COMMAND
		IMMEDIATE	DELAY		
Magnetic Tape Units	REW RUN ERG WTM BSR BSF FSR FSF				Rewind Tape Rewind and Unload Tape Erase Gap (Writes Blank Tape) Write Tape Mark Backspace to Interrecord Gap Backspace to Tape Mark Forward Space to Interrecord Gap Forward Space to Tape Mark
Row Punch (Type 0604)	SS				Select Stacker
Drum Printer (Type 0768)	SP SK	a b	a b		Carriage Space (a) Lines Skip to Channel (b)
UNIVAC 8411 and 8414 Disc Subsystem	SEEK				Perform Seek to Current Track of Data
2703 Optical Document Reader				b 2 3	Select Stacker

NOTE: It is recommended that stacker selection on the ODR be employed in a single programming environment due to the timing considerations involved.

LEGEND:

a = 1, 2, or 3; 0 is not permitted

b = 4 through 15

Table 3–12. CNTRL Macro Instruction Parameter List

3.3.7. LBRET Macro Instruction

When the user has a special routine to process additional user standard or non-standard labels, control is returned to logical IOCS by means of a LBRET macro instruction after label processing is completed.

The format of the LBRET macro instruction is:

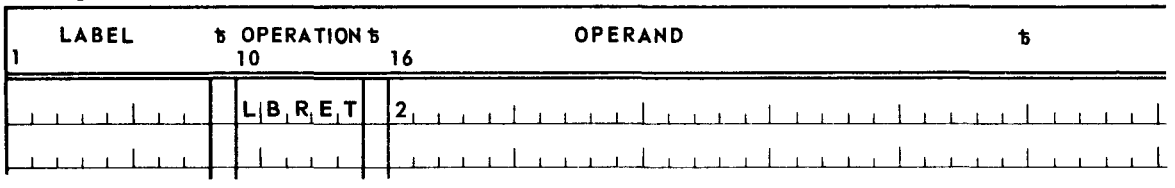
LABEL	♭ OPERATION ♭	OPERAND
[name]	LBRET	{1 / 2}

POSITIONAL PARAMETER 1

1 – informs logical IOCS that the user's label processing is finished. For input files, this prevents any more user labels from being read; for output files, it causes a tapemark for magnetic tape or an EOF record for disc to be written. Control should be returned to logical IOCS and not returned to the address supplied by the LBAD keyword parameter.

2 – requests that control be returned to LBAD after reading in the next label for the user to process, or after writing the label the user generated.

Example:



1. Causes a return to the OPEN or CLOSE transient routine currently in control and causes that routine to read or write another label, then returns control to the address specified by the LBAD keyword parameter.

On sequential disc or tape input with standard labels, when LBAD is entered, a user label (80 bytes in length) has been read into the I/O area and register 1 points to that label. Register 0 contains an O, F, or V EBCDIC character describing an open (O), end-of-file (F), or end-of-volume (V) condition, respectively. When the label has been checked, the user returns with a LBRET1 or LBRET2 macro instruction, and register 0 has no significance.

On tape input with nonstandard labels, when LBAD is entered, a user label (BKSZ in length) has been read into the I/O area and register 1 points to that label. Register 0 contains either an EBCDIC O or E character, describing an open (O) or end-of-tape (E) condition. When any label has been checked during the OPEN processing, the user returns with LBRET1 or LBRET2, and register 0 has no significance. However, during the CLOSE processing when the user returns by means of LBRET1 or LBRET2, register 0 must contain either the EBCDIC character EV or EF, describing either an end-of-volume (EV) or end-of-file (EF) condition.

On sequential disc or tape output with standard labels, when LBAD is entered, a user label (80 bytes in length) should be placed in the I/O area, as designated by register 1. Register 0 contains an EBCDIC O, F, or V character describing an open (O), end-of-file (F), or end-of-volume (V) condition, respectively. When the label to be written has been placed in the I/O area, as designated by register 1, the user returns by means of LBRET2, so that the label can be written. When no more labels are to be written, the user must return by means of LBRET1. Register 0 has no significance upon return.

On tape output with nonstandard labels, when LBAD is entered, a user label should be placed in the I/O area, as designated by register 1. Register 0 contains an EBCDIC O, F, or V character describing either an open (O), end-of-file (F), or end-of-volume (V) condition. When the label to be written has been placed in the I/O area as designated by register 1, and the length of the label is in register 0, the user returns by means of LBRET2 so that the label can be written. When no more labels are to be written, the user must return by means of LBRET1.

The following charts outline various label processing conditions encountered for tape and sequential disc files.

- Tape Output

DTF LABEL SPECIFICATIONS	LABEL BLOCK	ENTER LBAD WITH REG 0	ENTER LBAD WITH REG 1	LENGTH (BYTES)	BLOCK WRITTEN FROM	USER EXITS LBAD WITH REG 0	USER EXITS LBAD WITH REG 1 (LBRET)
STD – No LBAD	All	No Entry		80	Transient	No Exit	
STD – LBAD	VOL1-8	No Entry		80	Transient	No Exit	
	HDR1	No Entry		80	Transient	No Exit	
	EOF1	No Entry		80	Transient	No Exit	
	EOV1	No Entry		80	Transient	No Exit	
	HDR2-8	O	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	UHL1-8	O	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	EOF2-8	F	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	EOV2-8	V	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	UTL1-8	F or V	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
NSTD – No LBAD	–	No Entry		–	–	No Exit	
NSTD – LBAD	All Headers	O	IOA1 (or 2)	*	IOA1 (or 2)	Length	1 or 2
	All Trailers	F or V	IOA1 (or 2)	*	IOA1 (or 2)	Length	1 or 2

*The user places the nonstandard label to be written in IOA1 or IOA2 and exits from the LBAD routine with the byte count specified as a binary number in register 0.

■ Tape Input (Read Forward)

DTF LABEL SPECIFICATIONS	LABEL BLOCK	ENTER LBAD WITH REG 0	ENTER LBAD WITH REG 1	LENGTH (BYTES)	BLOCK READ INTO	USER EXITS LBAD WITH REG 0	USER EXITS LBAD WITH REG 1 (LBRET)
STD – No LBAD	All	No Entry		80	Transient	No Exit	
STD – LBAD	VOL1-8	No Entry		80	Transient	No Exit	
	HDR1	No Entry		80	Transient	No Exit	
	EOF1	No Entry		80	Transient	No Exit	
	EOV1	No Entry		80	Transient	No Exit	
	HDR2-8	0	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	UHL1-8	0	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	EOF2-8	F	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	EOV2-8	V	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	UTL1-8	F or V	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
NSTD – No LBAD	All	No Entry		80	Transient	No Exit *	
NSTD – LBAD	All Headers	0	IOA1 (or 2)	BKSZ	IOA1 (or 2)	–	2 (if TM)** 1 or 2 (no TM)
	All Trailers	E	IOA1 (or 2)	BKSZ	IOA1 (or 2)	EV or EF	1 or 2

* The Open or Close transient routine searches for a tape mark. If it is not found within 20 seconds, the transient is closed and the job cancelled. (The user must specify LBAD when no tape mark follows nonstandard labels.)

**If a tape mark follows nonstandard labels, the user passes all labels (whether checking or not) by issuing LBRET 2 until the transient reads a tape mark and concludes label checking. If a tape mark does not follow nonstandard labels, the user passes all labels (whether checking or not) by issuing LBRET 2 until he recognizes that the last label has been read. The user then issues LBRET 1 and the transient concludes label checking (with the first data record). If there are no header labels, and no tape mark (LBAD desired to process trailer labels), LBAD has been entered with the first data record read. The user must issue CNTRL (backspace) to position the tape at the first data record and exit with LBRET 1.

■ Tape Input (Read Backward)

DTF LABEL SPECIFICATIONS	LABEL BLOCK	ENTER LBAD WITH REG 0	ENTER LBAD WITH REG 1	LENGTH (BYTES)	BLOCK READ INTO	USER EXITS LBAD WITH REG 0	USER EXITS LBAD WITH REG 1
STD – No LBAD	All	No Entry		80	Transient	No Exit	
STD – LBAD	EOF1	No Entry		80	Transient	No Exit	
	HDR1	No Entry		80	Transient	No Exit	
	VOL1-8	No Entry		80	Transient	No Exit	
	UTL1-8	0	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	EOF2-8	0	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	UHL1-8	F	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
	HDR2-8	F	IOA1 (or 2)	80	IOA1 (or 2)	–	1 or 2
NSTD – No LBAD	All	No Entry		80	Transient	No Exit	
NSTD – LBAD	All Trailers	0	IOA1 (or 2)	BKSZ	IOA1 (or 2)	–	1 or 2
	All Headers	E	IOA1 (or 2)	BKSZ	IOA1 (or 2)	EF	1 or 2

■ Sequential Disc Output

DTF LABEL SPECIFICATIONS	LABEL KEY	ENTER LBAD WITH REG 0	ENTER LBAD WITH REG 1	LENGTH (BYTES)	BLOCK WRITTEN FROM	USER EXITS LBAD WITH REG 0	USER EXITS LBAD WITH REG 1 (LBRET)
No LBAD	None	No Entry				No Exit	
LBAD	UHL1-8	0	IOA1	80	Transient	–	1 or 2
	UTL0-7	F or V	IOA1	80	Transient	–	1 or 2

■ Sequential Disc Input

DTF LABEL SPECIFICATIONS	LABEL KEY	ENTER LBAD WITH REG 0	ENTER LBAD WITH REG 1	LENGTH (BYTES)	BLOCK READ INTO	USER EXITS LBAD WITH REG 0	USER EXITS LBAD WITH REG 1 (LBRET)
No LBAD	None	No Entry				No Exit	
LBAD	UHL1-8	0	IOA1	80	IOA1	–	1 or 2
	UTL0-7	F or V	IOA1	80	IOA1	–	1 or 2

3.3.8. CLOSE Macro Instruction

When all the data in a file has been processed, a CLOSE macro instruction should be issued. This macro instruction initiates the termination procedures for the file.

The format for the CLOSE macro instruction is:

LABEL	↹ OPERATION ↹	OPERAND
[name]	CLOSE	{ filename filename-1,filename-2,...,filename-n (1) }

POSITIONAL PARAMETER 1

filename – is the label(s) of the corresponding DTF (Define The File) macro instruction(s) in the problem program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:

```
     LABEL      ↹ OPERATION ↹      OPERAND                    ↹
1    2          10                 16
1.   ENDREAD    CLOSE              INPUT
```

1. Enters the transient routine which closes the file described in the DTF macro instruction whose label is INPUT.

3.3.9. FEOV Macro Instruction

This macro instruction causes end-of-volume procedures to be initiated on an input or output file when a user wishes to discontinue the processing of the current volume of the file and begin processing the next volume. The FEOV macro instruction applies only to magnetic tape and direct access storage devices.

Processing of input records is discontinued immediately when the FEOV macro instruction is issued for an input file. Since the end of a volume presumably has not been reached, the end-of-volume label checking procedures appropriate to the device are bypassed for magnetic tape. Direct access devices permit user label processing. When the device is a magnetic tape, the tape is rewound with interlock. A volume swap occurs and the succeeding volume is opened. The next GET macro instruction accesses the first record on the next volume of the input file.

Processing of output records is discontinued immediately when a FEOV macro instruction is issued. Any data which has been passed to logical IOCS by means of the PUT macro instruction is sent to the device either as a full or truncated block. The normal end-of-volume label procedures are accessed as if the end of a volume has been reached.

The volume swap then occurs and the next volume is opened. For a magnetic tape device, the current volume is rewound with interlock. For a direct access storage device, the last record entry field in the format 1 label (file header label) is updated to reflect the last valid record on the current volume. The FEOV macro instruction may not be issued during user label processing procedures. For sequential files (tape or disc), FEOV must not be issued for the last volume of a multivolume file. Otherwise, fatal errors are detected by logical IOCS.

The format of the FEOV macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	FEOV	{ filename / (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTF (Define The File) macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:

	LABEL	♭ OPERATION ♭	OPERAND	♭
	1	10	16	
1.		FEOV	INFILE	

1. Causes the file described by the DTF macro instruction, whose label is INFILE, to be treated as if an end-of-volume sentinel had been accessed.

3.4. DATA MANAGEMENT ERROR ANALYSIS ROUTINE (DMEAR)

The purpose of the Data Management Error Analysis Routine (DMEAR) is to provide the Data Management user with printed information concerning the nature of a fatal hardware or logical error which has occurred in opening, processing, or closing a Data Management file of any type.

The DTF macro call of all Data Management files has the optional keyword parameter:

ERROR=symbol

where symbol is the label of the user's error handling routine. The Data Management Error Analysis Routine is intended to be called in this error handling routine by the proc DMEAR. A typical error handling routine might consist of the DMEAR and CANCEL macro calls.

DMEAR prints system information, file information, and (if an I/O error has occurred) I/O information on the system list (SYSLST) device or, optionally, on the console printer.

■ CALLING PROCEDURES

The DMEAR macro call inserted in the user program's error handling routine will initiate the error analysis. General register 0 contains the reason (error code) for the error and is loaded by Data Management before passing control to the error address. Control is returned to the user program at the instruction following the DMEAR macro expansion. Registers 0 and 1 are restored. The various forms of the macro call are:

LABEL	OPERATION	OPERAND
[name]	DMEAR	{filename / (1)} [,CON]

POSITIONAL PARAMETER 1

filename — is the label of the DTF macro call in the user's program.

(1) — indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

CON — causes the error analysis text to be output to the console printer if the SYSLST device is unavailable at execution time.

■ CONSOLE OUTPUT

The text output of the error analysis routine goes to the SYSLST device if that device is available at execution time. By including the CON parameter to the DMEAR macro call, the user can cause the error analysis text to be output to the console printer if the SYSLST device is unavailable at execution time. The conditions which will cause the SYSLST device to be unavailable to the error analysis routine are:

- The SYSLST device is being used by another program.

- OPTION NODUMP is in effect.

- OPTION MAYIDUMP is in effect and the operator's reply to 'SV51 DMEAR REQUIRED?' was NO.

All console messages output by the error analysis routine have the message prefix DM. See the *UNIVAC 9400 Operations Handbook Operator Reference, UP-7871* (current version).

- RESTRICTIONS

  - The two low order bytes of general register 0 should not be altered between the ERROR=symbol address and the DMEAR macro call.

  - If the address passed to the error analysis routine in general register 1 is not within the user program's boundaries, control is returned immediately to the user with no error indication and no printed output. If register 1 contains an address within the user program's boundaries but not a valid DTF address, the results are unpredictable.

  - Under DOS operation, the PROG NAME (part of the DM02 console message) will be incorrect if an RDFCB has used the CCB in the user job's preamble before the error analysis routine is called.

  - If the CCW chain pointed to by the user's Data Management file CCB begins below the user job's preamble (e.g., in a transient area), that chain will not be printed. In addition, the ERR CCW and CMD printed may be in error.

  - The error analysis routine will not analyze chained BCW's.

  - In order to minimize console printer use, CCW chains and DATA are not output to the console.

  - If the SYSLST device is unavailable to the error analysis routine and the user has not permitted console output by including the CON parameter to the DMEAR macro call, control is returned immediately to the user with no error indication and no printed output.

4. NONSEQUENTIAL FILE PROCESSING

4.1. DIRECT ACCESS METHOD

The direct access method allows the user of a direct access storage subsystem to process a file in either a random or sequential manner.

The method of accessing a record in such a data file is specified by the user either by means of a key field to match one recorded on the disc, or by means of an identification address in absolute or relative form. When a key field is specified, the data within the file is searched for a record with a matching key. When an identification address (ID) is specified for a record, the absolute address form represents the actual location of the record on the direct access storage device. The relative form of addressing is used when the relative position of the record within the file is known.

The basic imperative macro instructions are READ and WRITE, which permit records to be read, written, replaced, or deleted at specific points within a file. When a READ or WRITE macro instruction is issued, the requested I/O operation is either initiated immediately or placed on a wait list for later execution. To ensure that the requested I/O operation has been successfully completed, a WAITF macro instruction must be issued prior to the next I/O request. The WAITF macro instruction tests for successful completion of any READ or WRITE operation.

4.1.1. Declarative Macro Instruction

A file to be processed by the direct access method must be defined in the problem program by the declarative macro instruction, DTFDA. The symbolic name of the file (filename) may have a maximum of seven characters, and is used by the imperative macro instructions to identify the file.

To enable the user to link his program with a DTFDA table generated in a separate assembly, the following references are generated by this macro instruction:

- An ENTRY definition for filename. The user must supply an EXTRN definition for filename within his program.
- An EXTRN definition for each label supplied by parameters ERRBYTE, IOA1, SEEKADR, ERROR, IDLOC, KEYARG, LBAD, and XTNTXIT. The user must also supply ENTRY definitions for these labels in his program.

Following is a listing in alphabetic order of the required and optional keyword parameters which may appear in the operand of the DTFDA macro instruction. A description of each keyword parameter follows the listing. A summary of the keyword parameters is given in Table 4–4 at the end of the descriptions.

LABEL	♭ OPERATION ♭	OPERAND
filename	DTFDA	BKSZ=n ERRBYTE=symbol IOA1=symbol RCFM={FIXUNB / UNDEF} SEEKADR=symbol TYPE={INPUT / OUTPUT} [AFTER=YES] [CNTRL=YES] [DEVICE={8411 / 8414 / 8424}] [ERROR=symbol] [HOLD=YES] [IDLOC=symbol] [KEYARG=symbol] [KEYLEN=n] [LBAD=symbol] [READID=YES] [READKEY=YES] [RCSZ=(r)] [RELATIVE={R / T}] [SRCHM=YES] [VERIFY=YES] [WRITEID=YES] [WRITEKEY=YES] [XTNTXIT=symbol]

NOTE: Assembler rules concerning commas and continuation of parameters in the operand field apply when writing this macro instruction; see 1.2.

- AFTER Keyword Parameter

  The following keyword parameter is specified if a subsequent WRITE macro instruction contains an AFTER or an RZERO positional parameter. A format write is involved.

  AFTER=YES

- Block Size

  This keyword parameter specifies the size of the I/O area and has the following format:

  BKSZ=n

  where n is the size of the I/O area in bytes. This keyword parameter is required. The size requirements of the I/O area are determined by four factors:

  (1) The length of the data to be read or written. If the format is variable, the maximum size of the data should be provided.

  (2) The size of the key. This depends on whether the keyword parameter KEYLEN is specified and one of the following keyword parameters is specified:

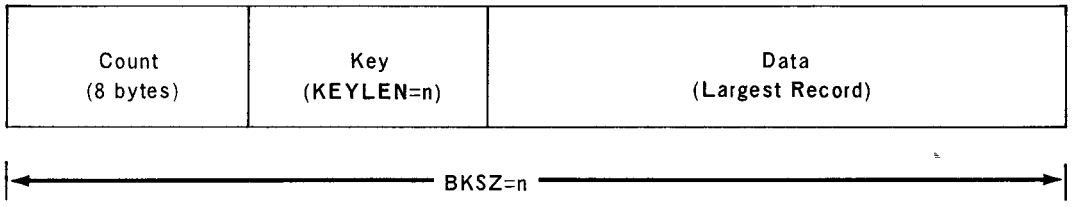
  AFTER, READID, or WRITEID

  (3) The size of the count area (eight bytes). This must be included if the keyword parameter AFTER is specified.

  (4) A maximum size of 3625 bytes for the UNIVAC 8411 Disc Subsystem or 7295 bytes for the UNIVAC 8414 and 8424 Disc Subsystems. This maximum size does not include the eight bytes required if AFTER is specified.

  Requirements for minimum sizes are described subsequently in this section at Input/Output Area (IOA1=symbol.)

  These storage requirements are illustrated in Figure 4–1.



CREATE A FILE OR WRITE ADDITIONAL RECORDS IN A FILE:

RECORDS WITH KEY AREAS

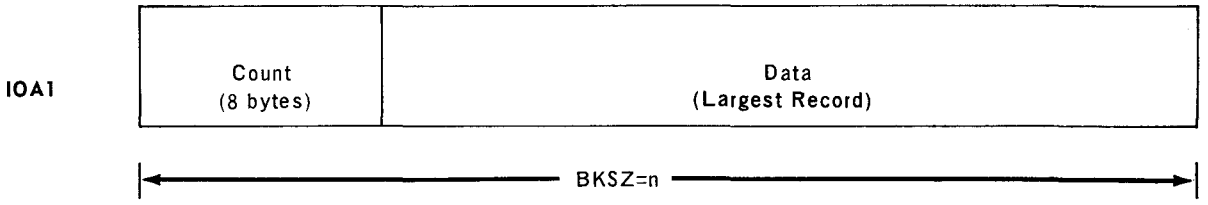
AFTER=YES

KEYLEN is specified

WRITEID or READID may be specified

WRITEKEY or READKEY may be specified

Figure 4–1. Schematic of Formats for IOA1 in Main Storage (Part 1 of 2)



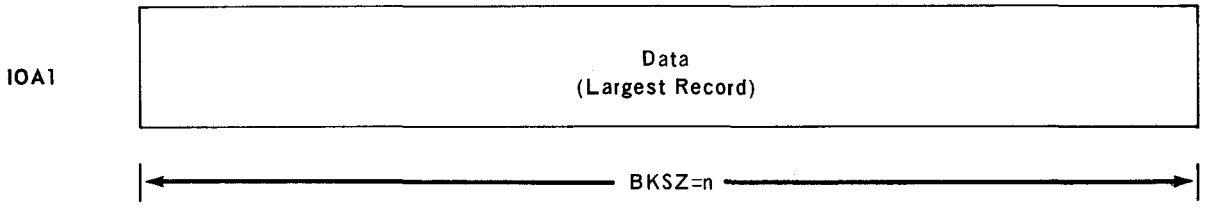
RECORDS WITHOUT KEY AREAS

AFTER=YES

KEYLEN is not specified

WRITEID or READID may be specified

WRITEKEY or READKEY may not be specified



READ OR WRITE (UPDATE) BY KEY, OR BY ID:

AFTER is not specified

and 1) KEYLEN is not specified.

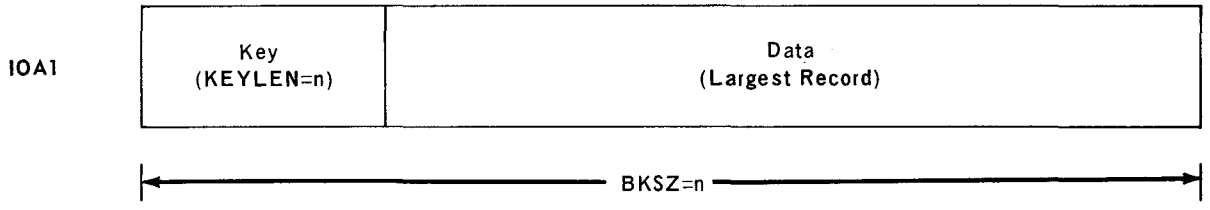
READID or WRITEID is specified.

READKEY or WRITEKEY is not specified.

or 2) KEYLEN is specified.

READKEY or WRITEKEY is specified

READID or WRITEID is not specified.



READ OR WRITE (UPDATE) BY ID:

AFTER is not specified.

KEYLEN is specified.

WRITEID or READID is specified.

READKEY or WRITEKEY may be specified.

Figure 4–1. Schematic of Formats for IOA1 in Main Storage (Part 2 of 2)

- Control Entry

  The following keyword parameter specifies that a CNTRL macro instruction is to be issued by the problem program.

  CNTRL=YES

  The orders associated with the CNTRL macro instructions are executed only for nonshared units; otherwise, they are ignored.

- Device Type

  The keyword parameter that defines the type of device upon which the data file exists is:

  DEVICE={8411 / 8414 / 8424}

  where 8411 represents the UNIVAC 8411 Disc Subsystem, 8414 represents the UNIVAC 8414 Disc Subsystem, or 8424 represents the UNIVAC 8424 Disc Subsystem.

  This is not a required parameter; the OPEN transient routines determine the device characteristics from the Physical Unit Block (PUB) device type of the first volume. All other volumes of the file must be consistent with the first.

- Error Status Codes

  The user must provide the address of a two-byte field into which logical IOCS can set error conditions by means of the following keyword parameter:

  ERRBYTE=symbol

  where symbol (label) is the address of a two-byte field to contain the error codes. The codes are made available by the WAITF macro instruction for user testing. Table 4–1 summarizes the possible return codes.

BYTE	BIT	ERROR/STATUS CODE	REASON FOR SETTING
0	0	---	
0	1	Wrong length	(1) READ Instructions (a) By KEY – FIXED – data size of disc record differs from data size specified. – UNDEF – data size of disc record is greater than maximum data size specified. (b) By ID – FIXED – if KEYLEN > 0 and key size of next record on track differs from key size specified, or data size of next record on track differs from data size specified. – UNDEF – key size of disc record differs from key size specified if KEYLEN > 0, or data size of disc record is greater than maximum data size specified. (2) Data WRITE Instructions (a) By KEY – FIXED – data size of disc record differs from data size specified. – UNDEF* – data size of record to be written is greater than maximum data size specified. (b) By ID – FIXED – if KEYLEN > 0 and key size of next record on track differs from key size specified, or data size of next record on track differs from data size specified. – UNDEF* – data size of record to be written is greater than maximum data size specified. (3) Format WRITE Instructions (a) With AFTER specified – FIXED – is meaningless. – UNDEF – data size of record to be written is greater than maximum data size specified. A check is made before the instruction is issued and only maximum data size is written. (b) With AFTER, EOF specified – is meaningless. (c) With RZERO specified – is meaningless.

*Check is made before instruction is issued and only maximum data size is written.

Table 4–1. Error Status Codes for Direct Access Method
(Part 1 of 3)

BYTE	BIT	ERROR/STATUS CODE	REASON FOR SETTING
0	2	---	---
0	3	Track not locked	This indicator is set after a RELEX macro instruction if the track to be released was not found in the lockout table for that job. It is not set if the ALL positional parameter has been specified.
0	4	No room found	This indicator is set when a WRITE macro instruction using the AFTER positional parameter has been requested and there is not enough room on the track for this record. No part of the record is written and the capacity record (R0) is not altered. This bit should never be set when using the keyword parameter RELATIVE=R and the AFTER positional parameter of the WRITE macro instruction in combination.
0	5	Track locked	This indicator is set when a READ macro instruction with the H positional parameter or a WRITE macro instruction was issued and another job has an entry in the lockout table for the address in the SEEKADR parameter. The user is responsible for re-issuing the order. This indicator is not set unless HOLD=YES has been specified.
0	6	Lockout table full	This indicator is set when a job attempts to make an entry into a full lockout table. The user must reissue the order. This condition may cause an indefinite loop if the user continues to reissue the command. The problem may be caused by inadequate table size or by failure to release entries made by any job in the system.
0	7	---	---
1	0	Data Check-Count Area	Unrecoverable error. If this error occurs, the requested WRITE or READ has not been successfully completed. The user should either ignore this record, if possible, or terminate processing of the file.
1	1	Track overrun	This bit is set when a WRITE macro instruction with the AFTER positional parameter is executed and more data remains to be written than will fit on the track. If this error occurs, some part of the count, key, and/or data has been written but the capacity record (R0) has not been updated. This is a programming type error.
1	2	End of cylinder	This bit is set when an order associated with a READ or WRITE macro instruction with a KEY positional parameter, and the search multiple option is active, is issued and the record is not found before reaching the end of the cylinder. Byte 1, bit 4 will also be set. This condition applies to both absolute and relative addressing. This bit is also set if the IDLOC keyword parameter is supplied and the search multiple option is not involved (either the option is not specified or is not allowed, or a READ or WRITE macro instruction with ID specified has been issued), and (1) the next ID is in a different cylinder or (2) the next ID is in a different volume This setting applies only to absolute addressing.

Table 4–1. Error Status Codes for Direct Access Method (Part 2 of 3)

BYTE	BIT	ERROR/STATUS CODE	REASON FOR SETTING
1	3	Data check in key or data	Unrecoverable error. If this error occurs, the requested WRITE or READ macro instruction has not been executed. The user should ignore this record if possible or terminate processing of this file.
1	4	Record not found	This indicator is set when the order associated with a READ or WRITE macro instruction with a KEY or ID positional parameter does not find the specified record. This bit may also be set in conjunction with the end-of-cylinder bit (see byte 1, bit 2). If this bit is set when the keyword parameter RELATIVE=R is specified, a record has not been found in the specified location.
1	5	End of file	This bit is set when the order associated with a WRITE macro instruction with an ID or KEY positional parameter is executed and the referenced disc record has a data length of 0. This bit is also set when the order associated with a READ macro instruction references a disc record with a data length of 0. When this bit is set, the ID returned is all 1 bits, if the IDLOC keyword parameter is specified.
1	6	End of volume	If the absolute address at the location defined by the SEEKADR keyword parameter is physically nonexistent, this bit is set. When the keyword parameter IDLOC is specified, this bit is also set in conjunction with the end-of-cylinder bit if: (1) The next ID is on the next volume. (2) The record read or written was the last record in the file. (All 1 bits are supplied in the location defined by the IDLOC keyword parameter.)
1	7	Invalid ID; record not in specified extents	If the keyword parameter RELATIVE is specified, this bit is set only for a record or track number higher than the number of records or tracks in the file. If the AFTER positional parameter of a WRITE macro instruction and RELATIVE=R are both specified, this bit may be set when the record to be written is not the next record that should be written on the requested track of the file. If RELATIVE is not specified, this bit is set when the address at the location defined by the SEEKADR keyword parameter is not in the user's extents.

Table 4–1. Error Status Codes for Direct Access Method (Part 3 of 3)

- Major File Error

  When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

  ERROR=symbol

  where symbol (label) is the address of the error handling routine.

  When the user gets control at this address, the file is not marked as being OPEN and registers 1 through 12 have been restored. Register 0 contains the following information concerning the reasons for the error:

  - Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary 0's.

  - Byte 2 contains a one character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients (such as OPEN) and 1 = processing (such as READ, WAITF, WRITE, or CNTRL).

  - Byte 3 contains a one character EBCDIC value indicating the reason for the error; refer to Appendix E.

The OPEN transient routine causes an informational typeout, inserting bytes 2 and 3 as xx in the standard Data Management message format (DMxx). The CLOSE transient also causes informational typeouts for detectable logical errors, but control is returned to the user at the instruction following the CLOSE macro instruction rather than at the ERROR address.

If control is passed to the ERROR address from a processing macro instruction (READ, WRITE, CNTRL, or WAITF), the address of the instruction following that macro call can be found in register 14. In the case of the OPEN transient routine, register 14 is restored to the value in the register preceding the OPEN call.

If this keyword parameter is not specified, the OPEN transient routine causes the informational typeout on the console, stores the above register 0 information in the DTF table at location DC$ERR, and then executes abort procedures when a major file error occurs.

If this keyword parameter is not specified and a detectable logical error occurs, the processing macro instruction (READ, WRITE, CNTRL or WAITF) stores the register 0 information in the DTF table at location DC$ERR and then executes abort procedures. However, if this keyword parameter is not specified and a fatal hardware error occurs, abort procedures are executed. These major file errors do not coincide with those set in the error status codes (ERRBYTE).

- Track Protection

  Prevention of the address specified in the SEEKADR parameter from being accessed by more than one job is accomplished by specifying the following keyword parameter:

  HOLD=YES

This parameter is used only when the Supervisor has the facility for handling record lockout. When this parameter is specified, the execution of the READ and WRITE imperative macro instructions is affected as follows:

- All WRITE macro instructions (format and update forms) check the lockout table for the address in the SEEKADR parameter. If another job has made an entry for this track, the TRACK LOCKED bit in ERRBYTE is set and the command is not issued. The user is responsible for reissuing the command. If an entry has not been made for the track, an entry is made and the command is issued. A WRITE macro instruction removes the entry from the lockout table after completion of a command whether or not the command was successfully executed.

  NOTE: A WRITE filename,KEY statement is used to protect only the track specified in the SEEKADR parameter; therefore, if the SRCHM parameter option is used, the track being accessed may not be protected.

- A READ macro instruction with the H positional parameter specified checks the lockout table for the address in the SEEKADR parameter. If an entry has been made by another job, the TRACK LOCKED bit of ERRBYTE is set and the command is not issued. The user is responsible for reissuing the command; otherwise, an entry is made in the lockout table. A job may make more than one entry for a track.

  NOTE: A WRITE filename,KEY statement is used to protect only the track specified in the SEEKADR parameter; therefore, if the SRCHM parameter option is used, the track being accessed may not be protected.

  A READ macro instruction without the H positional parameter allows commands to be issued regardless of whether HOLD=YES is specified in the DTFDA macro instruction.

CNTRL=YES must not be specified if HOLD=YES is specified. The latter specification implies a multiprogramming environment with a shareable disc device. The Supervisor does not accept the CNTRL macro instruction in such an environment.

■ Identification Address

After certain READ or WRITE macro instructions are issued, the record identification address (ID) is stored in a five-byte field by specifying the following keyword parameter:

IDLOC=symbol

where symbol (label) is the address of the five-byte field containing the record identification address. The ID may be that of the same record or the next record, depending on the macro issued and the SRCHM keyword parameter. The format of the ID is the same as that specified for the location defined by the SEEKADR keyword parameter. This ID can be used to access the next sequential record of a file. If the data length of the referenced disc record is 0, the ID is set to all 1's.

Table 4–2 shows the various settings for IDLOC after certain macro instructions are issued.

KEYWORD OR POSITIONAL PARAMETER	NORMAL RETURN	RECORD IS THE LAST RECORD IN THE FILE	REFERENCED DISC RECORD DATA LENGTH=0
ID	The ID of the next record within the file	All 1 bits	All 1 bits
READKEY or WRITEKEY; SRCHM not specified	The ID of the next record within the file	All 1 bits	All 1 bits
READKEY or WRITEKEY; SRCHM is specified	The ID of the same record	The ID of the same record	All 1 bits
RZERO AFTER AFTER,EOF	*	*	*

*Meaningless

NOTE: If the file has not been allocated on a cylinder basis, the search multiple option is disallowed even if the SRCHM keyword parameter is supplied.

Table 4–2. Identification Address (ID) Supplied After a READ or WRITE Macro Instruction

- Input/Output Area

Each input and output file requires an area reserved for its individual use. This area is specified by the following keyword parameter:

IOA1=symbol

where symbol (label) is the address of the I/O area. The length of the area is specified by the keyword parameter BKSZ and must be large enough to contain the maximum number of bytes required in any READ or WRITE macro instruction issued for the file. The minimum number of bytes required is 80 when the keyword parameters LBAD and TYPE=INPUT are specified. Otherwise the minimum length is 14 bytes.

When WRITE macro instructions are issued, logical IOCS assumes that the I/O area contains the information implied by the type of instruction. If the AFTER positional parameter is used with the WRITE macro instruction, logical IOCS builds the eight-byte count field for the data being written.

Table 4–3 shows the contents of the I/O area for a given macro instruction.

MACRO INSTRUCTION		I/O AREA CONTENTS	
OPERATION	POSITIONAL PARAMETERS	WITH KEYLEN	WITHOUT KEYLEN
READ	filename,KEY	Data (Logical IOCS supplied)	Invalid
READ	filename, ID	Key and Data (Logical IOCS supplied)	Data (Logical IOCS supplied)
WRITE	filename,KEY	Data (User supplied)	Invalid
WRITE	filename,ID	Key and Data (User supplied)	Data (User supplied)
WRITE	filename,RZERO	Anything (Logical IOCS does not change this area)	Anything (Logical IOCS does not change this area)
WRITE	filename,AFTER[,EOF]	Count (Logical IOCS supplied) Key and Data (User supplied)	Count (Logical IOCS supplied) Data (User supplied)

Table 4–3. I/O Area Contents for Logical IOCS

- Key Field

  The following keyword parameter specifies that records are to be identified by a key:

  KEYARG=symbol

  where symbol (label) is the address of the key field in the problem program. This keyword parameter must be used if the problem program issues READ or WRITE macro instructions with the KEY positional parameter specified. The KEYLEN keyword parameter must also be specified for the same file. The key is used in searching for a record with an identical key.

- Key Length

  When referencing records by key, generating new records with keys, or updating (or reading) both key or data areas of records, the following keyword parameter should be specified:

  KEYLEN=n

  where n is the length of the key in bytes. All keys in a file must have the same length. If this keyword parameter is not specified, a length of 0 is assumed for the key. If the length is 0, all keys in the file are ignored even if they are present. The maximum length for a key is 255 bytes; the minimum length is three bytes.

- Special Label Handling

  This keyword parameter supplies the address of a user routine to process user header labels. The format is:

  LBAD=symbol

  where symbol (label) is the address of the user routine. If user header labels exist or are to be written, they are located in the first track of the first extent for each volume within the file. Data begins in the second track of the first extent.

  If this keyword parameter is not specified, logical IOCS uses the format 1 label to determine where data begins. If the first extent of a volume is marked as containing labels, data is assumed to begin in the second track of the first extent for that volume. If the first extent is marked as not containing labels, data is assumed to begin in the first track of the first extent for that volume.

  If both the LBAD keyword parameter and the parameter TYPE=OUTPUT are defined, the first extent of each volume is marked as containing user labels.

  Reference should be made to the OPEN and LBRET macros for discussion of label processing.

- Read Record by ID

  If a record to be read is to be located by its address (ID), the following keyword parameter must be specified:

  READID=YES

  Logical IOCS expects that READ macro instructions with the ID positional parameter will be issued. If KEYLEN > 0, a READ macro instruction with the ID parameter will read both the key and data fields of the record. If KEYLEN = 0, it will read only the data field (even if a key field exists on the disc). Also, the record number specified at the address given by SEEKADR must be greater than zero.

- Read Record by KEY

  If a record to be read is to be located by its key, the following keyword parameter must be specified:

  READKEY=YES

  Logical IOCS expects that READ macro instructions with the KEY positional parameter will be issued. If the SRCHM keyword parameter is also specified, the file must be allocated on a cylinder basis; otherwise, the search for the key is on a single track only. The search on key always begins at the address given at the location defined by the SEEKADR keyword parameter. A READ macro instruction with the KEY positional parameter will read the data field of a record only.

- Record Format

  Record formats must be defined. In the direct access method there is no blocking; therefore, a record has the same format as a block. There are two types of record formats:

  – RCFM=FIXUNB

    This keyword parameter specifies that the records are of a fixed length.

  – RCFM=UNDEF

    This keyword parameter specifies that records are of variable or undefined length and must be used in conjunction with the RCSZ keyword parameter. If a key is specified for a record, the length of the key is the same for all records. If RELATIVE=R is specified, this option cannot be used.

  If RCFM is not supplied, fixed unblocked records are assumed.

- Record Size

  If the data length of records is undefined, the following keyword parameter must be specified:

  RCSZ=(r)

  where (r) is the number of the general register (2–12) that contains the data length of each record to be read or written. When logical IOCS reads a record, it supplies the data length of the record in the specified register during the execution of the subroutine called by the WAITF macro instruction. It is the responsibility of the user to place the data length of the record in the specified register before issuing a WRITE macro instruction for the record.

- Relative Addressing

  A direct access storage device may be addressed either in absolute or relative form by the problem program. Relative addressing is the preferred form for several reasons:

  (1) Any fragmentation of a file is not known to the user.

  (2) The problem program is not normally affected by the relocation of a file.

  (3) In a multiprogramming environment, one user does not normally control a disc pack; therefore, a problem program should operate independently of the exact allocation of the files.

  There are two types of relative addressing:

  – RELATIVE=R

    This form of the keyword parameter indicates that the record address (ID) given in the location defined by SEEKADR keyword parameter is a relative *record* number. It is applicable only to fixed-length records. The relative record number is converted to an absolute address. If the IDLOC keyword parameter is specified, the ID returned is a relative record number. A WRITE macro instruction with the RZERO positional parameter specified may not be issued when RELATIVE=R is specified.

– RELATIVE=T

This form of the keyword parameter indicates that the record address (ID) given in the location defined by the SEEKADR keyword parameter is a relative *track* number. Identification of the precise record on the track depends on whether the KEY or the ID positional parameter of the READ or WRITE macro instruction is used. If the ID positional parameter is used, the record address consists of a relative track number and an absolute record number. The record number must be greater than zero. If the KEY positional parameter is used, the record address consists of a relative track number and the record number is ignored.

■ Seek Address

The following keyword parameter specifies the address of an eight-byte field in the problem program:

SEEKADR=symbol

where symbol (label) is the address of an eight-byte field. When a READ or WRITE macro instruction with the ID positional parameter specified is issued, the SEEKADR field contains the address on the device of the record to be read or written. When a READ or WRITE macro instruction with the KEY parameter specified is issued, the SEEKADR field contains the address at which the key search starts. The value contained in SEEKADR may be expressed in one of three ways:

(1) Absolute Addressing

The keyword parameter RELATIVE is not specified.
The address of the record has the following form:

mbbcchhr

where the fields are described as follows:

m = Sequential number (1–8) of the volume in the file

bb = 00

cc = Cylinder number (0 – 199 for UNIVAC 8411 and 8414 Disc Subsystems, or 0 – 399 for UNIVAC 8424 Disc Subsystem)

hh = Read/Write head number (0–9 for a UNIVAC 8411 Disc Subsystem, or 0 – 19 for a UNIVAC 8414 or 8424 Disc Subsystem)

r = Sequential number of the record on the track

Each field is expressed as a binary value. The user should not address Record 0 on any track. When a WRITE macro instruction with the RZERO positional parameter specified is issued, the record number is ignored.

(2) Relative Track Addressing

The keyword parameter RELATIVE=T is specified.
The address of the track has the following form:

000ttttr

where:

000 = Reserved for future use

tttt = The binary track number relative to the beginning of the file. The number of the first track is 1.

r = The sequential number of the record on the track. This number (which must be greater than zero) must be provided when a record is referenced by a READ or WRITE macro instruction using the ID positional parameter. Otherwise, this byte is ignored.

(3) Relative Record Addressing

The keyword parameter RELATIVE=R is specified.

The eight-byte address field contains a binary number, right justified, representing the relative number of the record to be accessed in the file. Logical IOCS converts this number to an absolute address. The number of the first record in the file is 1.

- Search Multiple Tracks

A search on a key is extended to multiple tracks when the following keyword parameter is specified.

SRCHM=YES

The search is initiated at the address given in the location specified by the SEEKADR keyword parameter and continues until either the record is found or the end of the cylinder is reached. (The file must be allocated on a cylinder basis or this keyword parameter will be disallowed by the OPEN transient routine.)

- Type of File

This specification determines the method of label processing to be performed on the file. Either of the following options describing the type of file must be specified:

– TYPE=INPUT

This form of the keyword parameter specifies that standard labels are to be read and checked for this file. This option is assumed if the TYPE keyword is not specified.

– TYPE=OUTPUT

This form of the keyword parameter specifies that standard labels are to be written for this file.

Refer to Appendix A.2 for discussion of the type of checking and writing done in accordance with this parameter.

- Verification Requirements

  The following keyword parameter requests that a parity check be made of all records after they have been written on the device.

  VERIFY=YES

  Data Management reads back one byte of data and checks for a successful completion to verify that the record transferred was written correctly. The hardware cyclic check feature establishes if the transfer was completed successfully. Verification of records necessarily increases the execution time for the commands associated with the WRITE macro instruction.

  If the keyword parameter is not specified, a parity check of the records is not made.

- Write Record by ID

  The following keyword parameter must be specified if an output record is to be located by means of its address (ID).

  WRITEID=YES

  WRITE macro instructions with the ID positional parameter will be issued. If KEYLEN > 0, a WRITE macro instruction with the ID parameter will update both the key and data fields of the record. If KEYLEN=0, only the data field is updated (even if a key field exists on the disc).

- Write Record by KEY

  The following keyword parameter must be specified if an output record is to be located by its key:

  WRITEKEY=YES

  WRITE macro instructions with the KEY positional parameter will be issued. If the SRCHM keyword parameter is also specified, the file must be allocated on a cylinder basis. Otherwise, the search for the key is on a single track only. The search on key always begins at the address given at the location supplied by the SEEKADR keyword parameter. A WRITE macro instruction with the KEY positional parameter specified will update only the data field of a record.

- Special Extent

  The following keyword parameter causes the information in an extent to be passed to the user:

  XTNTXIT=symbol

  where symbol (label) is the address of a user routine to process each extent in the data file. During the processing associated with the OPEN macro instruction issued for the file, logical IOCS transfers control to the specified user routine for each extent in the data file and places in register 1 the address of a 14-byte area containing the extent information. The user must return control to logical IOCS by means of the LBRET macro instruction. If the first extent on any volume is identified as including user labels, the label track is excluded from the extent information passed to the user; see 2.3.8 for format details.

The format of the 14-byte area follows:

BYTES	CONTENTS
0	Extent type indicator (binary)
1	Extent sequence number (binary)
2–5	Lower limit of the extent, in the form CCHH (discontinuous binary)
6–9	Upper limit of the extent, in the form CCHH (discontinuous binary)
10–11	Sequential number (1–8) of the volume in the file (binary)
12–13	Zeros

```
LABEL    OPERATION  OPERAND                                      COMMENTS
1        10     16                                                                72
*    DEFINE THE FORMAT OF A DIRECT ACCESS INPUT FILE
*        LABELED INFILE



INFILE   DTFDA  BKSZ=1111,      INAREA IS 1111 BYTES LONG                         1
                DEVICE=8411,    DIRECT ACCESS DEVICE IS THE 8411                  2
                ERRBYTE=ER01,   ER01 IS A 2-BYTE ERROR CODE FIELD                 3
                IOA1=INAREA,    INAREA IS THE INPUT OUTPUT AREA                   4
                KEYLEN=20,      EACH RECORD HAS A 20-BYTE KEY                     5
                READID=YES,     READ INFILE,ID                                    6
                RCFM=FIXUNB,    ALL RECORDS ARE THE SAME SIZE                     7
                SEEKADR=IDADR,  IDADR IS A 8-BYTE FIELD - DISC ID                 8
                TYPE=INPUT,     LABELS ARE TO BE READ AND CHECKED                 9
                WRITEID=YES     WRITE INFILE,ID
```

KEYWORD	SPECIFICATION	MACROS		REMARKS
		READ	WRITE	
AFTER	YES		X	A capacity record on each track is assumed
BKSZ	n=maximum block size	R	R	Length of IOA1, in bytes
CNTRL	YES	X	X	Required if CNTRL macro instruction is to be used
DEVICE	8411, 8414, or 8424	X	X	Identifies device
ERRBYTE	Symbolic label	R	R	Error status byte address
ERROR	Symbolic label	X	X	Address of user error routine
HOLD	YES	X	X	Track protection option
IDLOC	Symbolic label	X	X	Address of field for ID
IOA1	Symbolic label	R	R	Name of I/O area defined by user
KEYARG	Symbolic label	X	X	Address of field for key used for key search
KEYLEN	n=key length	X	X	Length of the key in bytes
LBAD	Symbolic label	X	X	Address of user label handling routine
READID	YES	X		Record referenced by ID
READKEY	YES	X		Record referenced by KEY
RCFM	FIXUNB†	Y	Y	For fixed-length records; if omitted, FIXUNB is assumed
	UNDEF	Y	Y	Not fixed-length records
RCSZ	(r)=register number	X	X	For undefined records
RELATIVE	R	X	X	Relative addressing – record
	T	X	X	Relative addressing – track
SEEKADR	Symbolic label	R	R	Address of track reference field
SRCHM	YES	X	X	Search multiple tracks (if specified, file must be allocated on a cylinder basis)
TYPE	INPUT †	Y	Y	Check standard labels
	OUTPUT	Y	Y	Write standard labels
VERIFY	YES		X	Records check-read
WRITEID	YES		X	Reference by ID
WRITEKEY	YES		X	Reference by KEY
XTNTXIT	Symbolic label	X	X	Address of extent processing routine

LEGEND:

R = Required
X = Optional
Y = One option required
† = Assumed parameter, if none specified

Table 4–4. Summary of Keyword Parameters for the DTFDA Macro Instruction

4.1.2. Imperative Macro Instructions

Logical IOCS allows records to be accessed by record key or by record identifier. If records contain keys, those records can then be accessed by a search for a match of the key. All records contain an identifier as part of the count area. The identifier is actually the address of the record, and access by record identifier is by a search for an address match. Access by key or identifier may be either to read or to write the record.

The imperative macro instructions which may be used in the direct access method are OPEN, LBRET, READ, WRITE, WAITF, CNTRL, and CLOSE. To access a file, an OPEN macro instruction must be executed, and when all processing is completed on the given file, a CLOSE macro instruction must be executed. The imperative macro instructions are described in detail in the following paragraphs.

4.1.2.1. OPEN Macro Instruction

After a file has been defined by the DTFDA declarative macro instruction, the OPEN macro instruction must be used to initialize the file before issuing a READ, WRITE, CNTRL, WAITF, or CLOSE macro instruction for the file. The OPEN macro instruction calls on a transient routine which performs the required operations. The function of OPEN is to initialize the file and perform standard label processing. Logical IOCS either checks or writes standard labels depending on the type of file. Extent information is passed to the user if the keyword parameter XTNTXIT has been specified. In addition, if the keyword parameter LBAD is specified, the OPEN transient routine either delivers user standard header labels or it writes user standard header labels.

Logical IOCS requires that all volumes of the file be available for use at OPEN time. All such volumes are accessed one at a time during the OPEN process. A file is opened as input or output according to keyword parameter TYPE in the declarative macro instruction for that file.

- Input Files (TYPE=INPUT)

  The OPEN transient routine executes the following steps:

  (1) Checks the VOL1 label.

  (2) Checks the VTOC labels.

  (3) Preserves extent information as specified in the VTOC labels.

  (4) If the keyword parameter, LBAD, is specified, logical IOCS delivers each of the 80-byte user standard header labels to the user one at a time in the I/O area. Register 1 contains the address of the area specified by the IOA1 keyword parameter. When the user has processed each label, control is returned to logical IOCS by means of the LBRET macro instruction.

  (5) The OPEN transient routine delivers all extents to the user one at a time in the area specified by the IOA1 parameter if the keyword parameter XTNTXIT is specified. Register 1 contains the address of the area specified by the IOA1 parameter. After processing each extent the user returns control to logical IOCS by means of the LBRET macro instruction.

- Output Files (TYPE=OUTPUT)

  The OPEN transient routine executes the following steps:

  (1) Checks the VOL1 label.

  (2) Checks the VTOC labels to see if this file is available for writing.

  (3) Fills in appropriate fields in the standard file labels of the VTOC.

  (4) If the keyword parameter LBAD has been specified, logical IOCS transfers control to the user label routine. Upon return from the user label routine by means of the LBRET macro instruction, the OPEN transient operation writes the user standard header labels. The address of the 80-byte area in which the label is built should be supplied in register 0. The OPEN transient routine uses the first four bytes of the 80-byte label identifier. The other 76 bytes of the label are supplied by the user (see 2.3.8).

  (5) The OPEN transient routine delivers all extents to the user one at a time in the area specified by the IOA1 parameter if the keyword parameter XTNTXIT has been specified. Register 1 contains the address of the area specified by the IOA1 parameter. The user returns to logical IOCS by means of the LBRET macro instruction after processing each extent.

  For additional information, see LBRET (4.1.2.6) and XTNTXIT (4.1.1). For direct access storage device conventions for user header and trailer labels, see 2.3.

The format of the OPEN macro instruction is:

LABEL	b OPERATION b	OPERAND
[name]	OPEN	{ filename filename-1,filename-2,...,filename-n (1) }

POSITIONAL PARAMETER 1

filename – is the label(s) of the corresponding DTFDA declarative macro instruction(s) in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:

1 LABEL	b OPERATION b 10	16 OPERAND	b
	OPEN	PAYROLL,CHGFILE	

4.1.2.2. READ Macro Instruction

This imperative macro instruction causes a record to be read from the direct access storage device into main storage. When control is returned after an operation, the data is not necessarily available. The user must issue a WAITF macro instruction to ensure that the data has been transferred. The address into which the data is read is specified by the keyword parameter IOA1.

The format of the READ macro instruction is:

LABEL	OPERATION	OPERAND
[name]	READ	$\left\{\begin{matrix}\text{filename}\\(1)\end{matrix}\right\}, \left\{\begin{matrix}\text{KEY}\\\text{ID}\end{matrix}\right\}[,H]$

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

KEY – a search is made for the record with the key matching that defined in the location specified by the keyword parameter KEYARG. The search begins at the track and cylinder address found in the location defined by the keyword parameter SEEKADR. If the keyword parameter SRCHM is specified and the file is allocated on a cylinder basis, the search continues through the cylinder until either a match is found or the end of the cylinder is reached. If the file is not allocated on a cylinder basis or the keyword parameter SRCHM is not specified, the search is confined to a single track. If the record is found, the data portion of that record is read.

ID – a search is made for a matching ID. (Refer to the SEEKADR keyword parameter for the format of the ID.) If the keyword parameter KEYLEN has been specified, the key and data portions of the record are read; otherwise, only the data portion is read. If KEYLEN is specified, it must be equal to the length of the key that is on the disc. The record number of the ID cannot be 0.

POSITIONAL PARAMETER 3

H – the direct access method processor checks the lockout table for the address in SEEKADR before issuing the I/O command. If no other job has made an entry for the track, an entry is made in the table and the command is issued. See HOLD=YES parameter in 4.1.1.

NOTE: To execute a READ macro instruction with the KEY positional parameter, the keyword parameter READKEY=YES must be specified in the DTFDA macro instruction.

To execute a READ macro instruction with the ID positional parameter, the keyword parameter READID=YES must be specified in the DTFDA macro instruction.

To execute a READ macro instruction with the H positional parameter, the keyword parameter HOLD=YES must be specified in the DTFDA macro instruction.

Example:

LABEL (1)	OPERATION (10)	OPERAND (16)
	READ	MASTER,KEY
	READ	MASTER,,ID,,H

4.1.2.3. WRITE Macro Instruction

This imperative macro instruction causes a record to be written from main storage to the direct access storage device. There are two forms for the WRITE macro instruction in the direct access method. One form provides for a direct write into a defined area on the disc and is used to update existing records. The count field is never altered in this form of the write instruction and the rest of the track is not cleared. The other form provides for writing a record following a specified record. A format write (which writes a count field followed by either a data field or key and data fields) is involved and the rest of the track is cleared. To ensure that the data has been transferred, a WAITF macro instruction must be used.

The main storage address from which the data is written is specified in the location defined by the keyword parameter IOA1.

If the keyword parameter VERIFY is specified, all records written are check-read for bad parity.

The format of the WRITE macro instruction for a write into a defined area on the disc is:

LABEL	OPERATION	OPERAND
[name]	WRITE	{filename / (1)} , {KEY / ID}

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

KEY –initiates a search for the record with the specified key, followed by a write of the new record over the data portion of that record. The search is initiated at the address given at the location specified by the SEEKADR keyword parameter and is confined to either a track or a cylinder depending on whether the SRCHM keyword parameter is specified.

ID —a search is made on a matching identifier. (Refer to the SEEKADR keyword parameter for format of the identifier.) The key field may or may not be updated. If the keyword parameter KEYLEN is specified, the key and data portions of the record are updated; otherwise, only the data portion is updated. The record number of the ID cannot be zero.

NOTE: When either the KEY or the ID positional parameter is specified in the WRITE macro instruction, the count field on the direct access storage device controls the key (where provided) and the data transfer. If the new record is longer than the old, the new record is truncated; if it is shorter, it is padded with binary 0's.

If the keyword parameter KEYLEN is specified and a WRITE macro instruction with the ID positional parameter is issued, the key field must be updated or rewritten and its length must equal the key length in the count area of the record. If the KEYLEN keyword parameter is not specified, the key field of the record is skipped and only the data portion of the record is updated.

The format of the WRITE macro instruction to write a record after a specified record is:

LABEL	OPERATION	OPERAND
[name]	WRITE	{filename / (1)} , {AFTER / AFTER,EOF / RZERO}

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

AFTER – a new record is written after an existing record. The new record (count, key (if present), and data) is written and the remainder of the track is cleared. The disc address given at the location defined by the SEEKADR keyword parameter is used to determine the track on which the new record is to be written. The capacity record (R0) is updated when the WRITE macro instruction has been successfully issued. This form is used to create a file, or to add records to an existing file.

RZERO – is used to clear a track specified by the address given at the location defined by the SEEKADR keyword parameter and to reset the capacity record (R0) to reflect the fact that R0 is the only record on the track. This form is used to delete records from a given track within a file, or to reset a track to its initial conditions. Also, this form may not be issued if RELATIVE=R has been specified.

POSITIONAL PARAMETER 3

EOF – is used to write an end-of-file record (data length of 0) after the last existing record on a track. The end-of-file record will contain a key, if KEYLEN has been specified. The user should supply a unique key to avoid any possible duplication. The capacity record (R0) is updated to include the end-of-file record. Positional parameter 2 must be specified as AFTER.

NOTE: When either of the AFTER or AFTER,EOF positional parameters are specified, the record is written only if there is sufficient space on the track. When the record will not fit on the requested track, the "no room found" bit of ERRBYTE is set. This should be checked every time a WRITE macro instruction, with the AFTER positional parameter specified, is issued.

Examples:

LABEL (1)	OPERATION (10)	OPERAND (16)
	WRITE	UPDATE,,RZERO
	WRITE	NEWFILE,,AFTER

4.1.2.3.1. Capacity Record

Record 0 on each track contains the identifier of the last record written on the track and the number of bytes still available. The capacity record (see Figure 4–2) is used with the WRITE macro instruction when the AFTER or RZERO positional parameter is specified to ensure that space is available for a record. When using the direct access method, there must be a capacity record on every track within the file which conforms to the following specifications.

COUNT AREA

0	1	2	3	4	5	6	7
IDENTIFIER					KEY LENGTH	DATA LENGTH	

DATA AREA

0	1	2	3	4	5	6	7
IDENTIFIER OF LAST RECORD					NUMBER OF UNUSED BYTES		RESERVED

Figure 4–2. Capacity Record – Record 0

- Count Area
  - Identifier is expressed in the form CCHH0.
  - Key length is 0.
  - Data length is 8.

- Data Area
  - Identifier of the last record is expressed in the form CCHHR.
  - Number of unused bytes is the dynamic count of the bytes available on this track. The maximum is 3625 bytes.
  - The last byte is reserved for system use.

4.1.2.4. WAITF Macro Instruction

This macro instruction is used to ensure that a command initiated by a preceding READ or WRITE macro instruction has been completed and that all the data has been transferred to the area specified in the DTFDA declarative macro instruction. When completed, the status code field (ERRBYTE) as defined in the DTFDA macro instruction contains the error or status information pertaining to the I/O request (see Table 4–1). It is the user's responsibility to check these bits. The WAITF macro instruction must be issued after a READ or WRITE macro instruction and before issuing another READ, WRITE, or CNTRL macro instruction.

The format of the WAITF macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	WAITF	{filename / (1)}

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

4.1.2.5. CNTRL Macro Instruction

This macro instruction enables the user program to control overlap between seek activity and read/write activity on nonshared units. The effect is to initiate head movement to the designated position on the direct access storage device. Since control is returned to the user when the order is initiated (instead of completed), the user can then process asynchronously with the head movement. Any previous READ or WRITE macro instruction must be accompanied by a WAITF macro instruction before a CNTRL macro instruction is executed.

A WAITF macro instruction should not be executed following a CNTRL macro instruction. If a CNTRL macro instruction is issued for a shared unit, the command associated with the macro instruction is ignored.

The format of the CNTRL macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
	CNTRL	{filename / (1)} , SEEK

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

SEEK – this parameter must be specified. The seek operation is initiated by logical IOCS at the address given at the location defined by the SEEKADR keyword parameter.

4.1.2.6. LBRET Macro Instruction

The LBRET macro instruction is used for direct access method files in connection with the OPEN macro instruction (see 4.1.2.1). The LBRET macro instruction enables the user to create or check user standard header labels and to process extent information.

The format of the LBRET macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	LBRET	{1 / 2}

POSITIONAL PARAMETER 1

1 – indicates that no more labels or extents are to be processed.

2 – indicates that there are more labels or extents to be processed.

The LBRET macro instruction may be used with direct access storage devices for any or all of the following purposes:

- Processing User Header Labels

  Logical IOCS delivers these labels one at a time to the user until either all existing user standard header labels have been read or until the user specifies there are no more labels.

  The user label routine may process the header label delivered by logical IOCS and then return control to logical IOCS by specifying either 1 or 2 for the positional parameter.

- Creating User Standard Header Labels

  The user delivers header labels to logical IOCS one at a time (up to a maximum of eight labels). The number 1 is specified in the positional parameter when the user wishes to stop writing labels before the maximum number (eight) has been written. When control is returned, logical IOCS writes a file mark, and continues with logical IOCS processing.

The number 2 is specified in the positional parameter if the user wishes to have control returned after logical IOCS has written the label. Control is returned to the user until eight labels have been written. When control is returned to logical IOCS with the eighth label, logical IOCS writes this label, writes a file mark, and continues with logical IOCS processing.

- Extent Checking

Logical IOCS delivers extent information to the user, one extent at a time. The number 1 is specified in the positional parameter when the user has completed processing all the information in the extent.

The number 2 is specified in the positional parameter if the user wishes to receive another extent. If there are no more extents, logical IOCS does not return control to the user even though the number 2 was specified in the positional parameter.

The user should design the coding that handles his header labels or extent information as island code, which permits the use of a synchronous or an asynchronous environment during OPEN transient processing. Control is given to the user at his specified address with a processing time limit of 500 milliseconds to execute a LBRET return. (If this time limit is exceeded, fatal error procedures will be followed.)

Following is an example of user extent processing, illustrating how user island code can function.

```
                LABEL      OPERATION      OPERAND
                1          10             16
1.  XTNTXIT     BALR       2,0
2.              USING      *,2
3.              L          3,EXT1
4.              MVC        0(14,3),0(1)
5.              LA         3,14(0,3)
6.              ST         3,EXT1
7.              AI         EXT2,1
8.              LBRET      2
9.  EXT1        DC         A(EXTAREA)
10. EXTAREA     DS         16CL14
11. EXT2        DC         Y(0)
12.             DROP       2
```

1. Load cover register
2. R2 is cover register
3. Point to current extent store area
4. Store current extent
5. Add 14 to address in register 3
6. Replace address in EXT constant

7. Count number of extents
8. Return control to OPEN for next extent
9. Address of extent store area
10. Extent store area; assumes a maximum of 16 extents
11. Counter for number of extents
12. Drop cover register

4.1.2.7. RELEX Macro Instruction

The RELEX macro instruction is used to remove entries from the lockout table.

The format of the RELEX macro instruction is:

LABEL	OPERATION	OPERAND
[name]	RELEX	{filename / (1)} [,ALL]

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFDA declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

POSITIONAL PARAMETER 2

ALL – indicates that all entries of the job are to be released. If this parameter is omitted, only the entry of the address in SEEKADR is released. If no entry for the track is found in the lockout table for that job, the TRACK NOT LOCKED bit of ERRBYTE is set.

4.1.2.8. CLOSE Macro Instruction

The CLOSE macro instruction is used to terminate processing for the file specified by filename. Once a file is closed it may not be accessed again except by issuing another OPEN macro instruction for that file.

The format for the CLOSE macro instruction is:

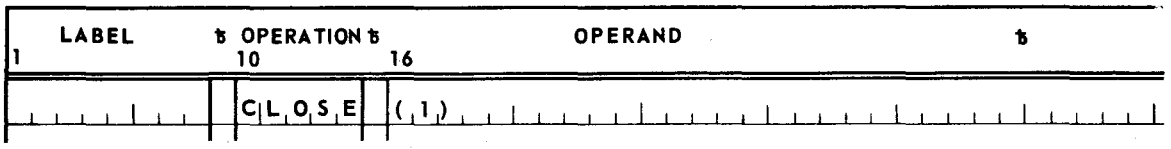
LABEL	OPERATION	OPERAND
[name]	CLOSE	{filename / filename-1,filename-2,...,filename-n / (1)}

POSITIONAL PARAMETER 1

filename – is the label(s) of the corresponding DFTDA declarative macro instruction(s) in the program.

(1) – indicates that register 1 has been preloaded with the address of the declarative macro instruction.

Example:



4.1.3. Instruction Usage

A file that uses the direct access method may be processed in either a sequential or a random manner. Following are examples which illustrate usage of the imperative macro instructions for both types of processing.

- Create a sequential file

WRITE filename,RZERO (RELATIVE ≠ R)	Used to initialize the capacity record for a track within the file (if this has not been done with a utility program).
WRITE filename,AFTER	Used to put a record on a track in the file. (Check "no-room-found" bit of ERRBYTE to determine if a rewrite to the next track is necessary.)
WRITE filename,AFTER,EOF	Used to write an end-of-file record at the end of the file.

- Read a sequential or random file

READ filename,ID READ filename,KEY	Used to access any record within the file.

- Update a sequential or random file

WRITE filename,ID WRITE filename,KEY	Used to update already existing records within the file.

- Add records to an existing file on a currently unused track.

```
WRITE filename,RZERO (RELATIVE ≠ R)
WRITE filename,AFTER
```

- Delete records on a given track within the file.

```
WRITE filename,RZERO (RELATIVE ≠ R)
```

4.1.4. **Data Management Error Analysis Routine (DMEAR)**

The Data Management Error Analysis Routine provides information for fatal hardware or logical errors that have taken place in opening, processing, or closing a Data Management file (3.4).

4.2. INDEXED SEQUENTIAL ACCESS METHOD

The Indexed Sequential Access Method (ISAM) allows the use of a UNIVAC Direct Access Storage (Disc) Subsystem to process a file in either a random or a sequential manner. Four distinct file processing functions are provided:

- Loading (creating, extending, or reloading) a file, which consists of writing presorted input records and their key fields onto the direct access storage device, while creating and writing out a set of indexes;
- Retrieving and updating records sequentially (in ascending order by key);
- Retrieving and updating records randomly (by key);
- Adding new records to an existing ISAM file.

Once a file has been created, any combination of the last three functions can be performed in a program.

An ISAM file contains three types of disc areas: prime data areas, an index area, and overflow areas. The bulk of the file is contained in the prime data areas of which there may be a maximum of eight; one on each volume of the file. Within a prime data area, the records are arranged in ascending order by key. They are always fixed in length, but for a given file may be in either an unblocked (one logical record per physical record) or blocked (several logical records per physical record) format.

When a new record is inserted into a prime data area track, all records on the track with higher keys are shifted toward the end of the track. The last record, if displaced from the track, is written into an overflow area. This may be an area associated with the cylinder containing reserved tracks (a cylinder overflow area), or it may be an area associated with the entire file (the independent overflow area). In either case, the overflow record is linked to the track from which it was displaced.

The basic control structure of an ISAM file is provided by a hierarchy of indexes. The first or lowest level index is the track (head) index, one of which appears at the beginning of each prime data cylinder. For each prime data track in the cylinder, there are two track index entries. The first entry contains the highest key on the track (that is, the key of the last record) and the disc address (cchh) of the track. The second entry of the pair contains the highest overflow record key associated with the track, and the disc address of the first overflow record (cchhr) associated with the track. The second and next higher level of index is the cylinder index which resides in the separate index area on the disc. For each prime data cylinder, there is a cylinder index entry which contains the highest key associated with the cylinder, and the disc address of the track index for the cylinder (that is, the address of the beginning of the cylinder). The third and highest level of index is the optional master index. If a master index exists for a file, it resides in the separate index area immediately preceding the cylinder index. Each entry in the master index contains the highest key on a cylinder index track and the disc address of that track.

To locate a logical record by its key, the proper track is located by searching down through the indexes. A search of the master index, if one exists for the file, yields a track address in the cylinder index. A search of the cylinder index track, or of the whole cylinder index if a master index does not exist, yields the address of the desired prime data cylinder. A search of the track index then yields the address of the prime data track, or overflow area containing the desired record. In the former case, the track is searched to locate the record; in the latter, the chain of overflow records linked to the prime data track is searched. Figure 4–3 is a flow diagram of the ISAM index search method.

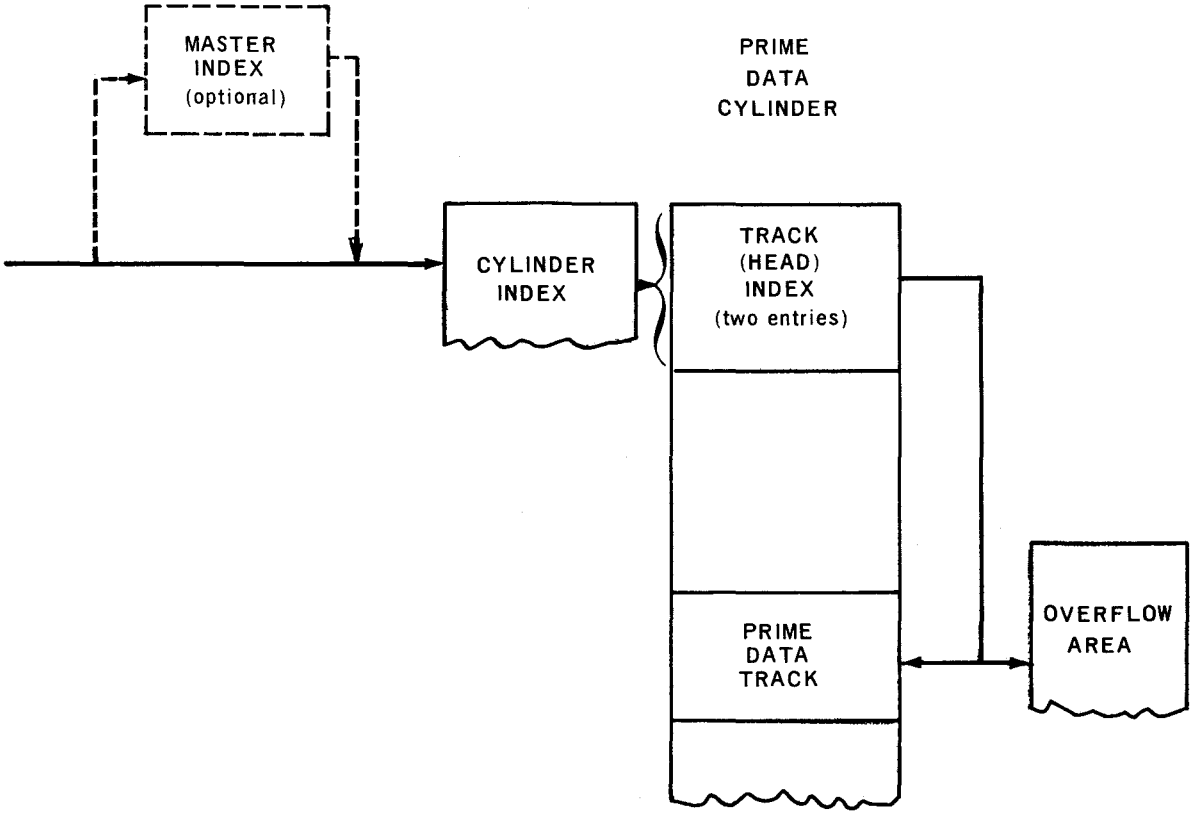


Figure 4–3. ISAM Index Search Method, Flow Diagram

4.2.1. Declarative Macro Instruction

A file that is to be processed by the indexed sequential access method must be defined in the problem program by the DTFIS declarative macro instruction. The symbolic name of the file (filename) can have a maximum of seven characters and must begin with an alphabetic character. The filename is also used in the related imperative macro instructions to identify the file.

Following is a listing, in alphabetic order, of the required and optional keyword parameters which may appear in the operand field of the DTFIS macro instruction. A description of each keyword parameter follows the listing. Figure 4–4 illustrates the dependency relationship among the parameters; that is, when certain parameters are specified, the programmer must choose two or more related parameters which, in turn, may require further choices, depending upon the file processing function being performed. A summary of the keyword parameters is given in Table 4–6 at the end of the description.

LABEL	OPERATION	OPERAND
filename	DTFIS	{IOAREAL=symbol [,IOAREAR=symbol] [,IOAREAS=symbol] IOAREAR=symbol [,IOAREAS=symbol] IOAREAS=symbol}
		IOROUT={ADD ADDRTR LOAD RELOAD RETRVE}
		KEYLEN=n
		RCFM={FIXBLK FIXUNB}
		RCSZ=n
		[CLOSE={(NOWRITE [,DISPLAY]) (DISPLAY)}]
		[{CYLOFL=n PCYLOFL=nn}]
		[DEVICE={8411 8414 8424}]
		[EOFA=symbol]
		[ERROR=symbol]
		[INDAREA=symbol]
		[INDSIZE=n]
		[IORG=(r)]
		[IOSIZE=n]
		[KEYARG=symbol]
		[KEYLOC=n]
		[MSTIND=YES]
		[NRECDS=n]
		[TYPE={RANDOM RANSEQ SEQNTL}]
		[VERIFY=YES]
		[{WORKS=YES [,WORKL=symbol] [,WORKR=symbol] WORKL=symbol [,WORKR=symbol] WORKR=symbol}]

NOTE: Assembler rules concerning commas and continuation of parameters in operand field apply when writing this macro instruction; see 1.2.

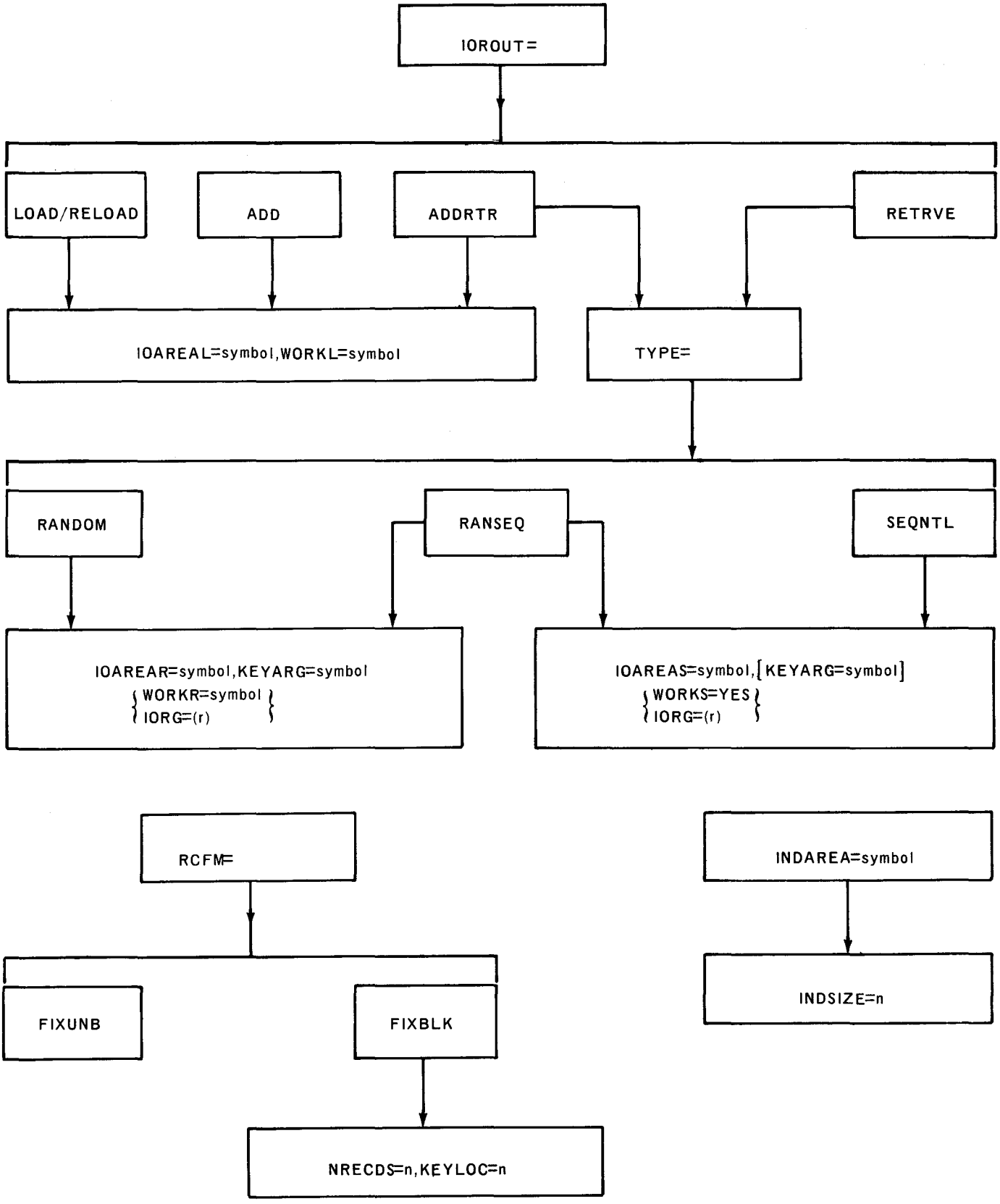


Figure 4–4. DTFIS Related Dependent Parameters

- Close Time

The optional keyword parameter CLOSE specifies two unrelated actions. When IOROUT=RETRVE, hardware file protect can be applied if the following keyword parameter is specified:

CLOSE=(NOWRITE)

The format 2 label will not be rewritten, and the fields labeled DL$TD2 and DL$NF2 will not be updated (see 2.3.3).

To display at the console the fields addressable by the filename linked with prescribed letter suffix, the following keyword parameter is specified:

CLOSE=(DISPLAY)

See E.7 for the format of the display. IOROUT determines which concatenated filename fields are available at close time:

(1) IOROUT= {LOAD / RELOAD}

filenameP
filenameS

(2) IOROUT=RETRVE

filenameR
filenameS
filenameT

(3) IOROUT=ADD

filenameA	filenameP
filenameI	filenameS
filenameO	

(4) IOROUT=ADDRTR

filenameA	filenameP
filenameI	filenameR
filenameO	filenameS
filenameT	

Therefore, when IOROUT=RETRVE it is possible to specify the following:

CLOSE=(NOWRITE,DISPLAY)

- Cylinder Overflow

The following optional keyword parameter specifies the number of tracks in each prime data cylinder that is reserved for cylinder overflow for a file being loaded:

CYLOFL=n

where n is the number of tracks (maximum of eight for UNIVAC 8411 disc or 18 for UNIVAC 8414 disc).

The following keyword parameter specifies the number of tracks reserved for cylinder overflow as a percentage of the number of tracks per cylinder:

PCYLOFL=nn

where nn is a percentage declaration less than 99 and greater than or equal to 0.

If neither CYLOFL nor PCYLOFL is specified, no tracks are reserved for cylinder overflow.

- Device Type

The following optional keyword parameter defines the type of device which contains the data file:

$$\text{DEVICE}=\left\{\begin{matrix}8411\\8414\\8424\end{matrix}\right\}$$

where 8411 denotes the UNIVAC 8411 Disc Subsystem, 8414 denotes the UNIVAC 8414 Disc Subsystem, or 8424 denotes the UNIVAC 8424 Disc Subsystem.

To process the file starting with the OPEN transient routines, use the device characteristics obtained from the Physical Unit Block (PUB) device type of the first volume. All other volumes of the file must be consistent with the first.

- End of File

When an end-of-file condition occurs in connection with sequential retrieval operations, the user may have control transferred to a special end-of-file routine by specifying the following optional keyword parameter

EOFA=symbol

where symbol is the address of the user's special end-of-file routine.

When control is transferred to the user's routine, register 14 contains the normal return address to the instruction following the imperative macro instruction during which the end-of-file condition was detected. Registers 2 through 12 are restored and register 1 contains the DTFIS macro instruction address.

If this keyword is not specified, control is returned to the normal return address in the user's program with an indication of the end-of-file condition in filenameC.

- Major File Error

When a fatal hardware or detectable logical error occurs on a file, the user may have control transferred to a special error handling routine by specifying the following keyword parameter:

ERROR=symbol

where symbol (label) is the address of the error handling routine. When logical IOCS transfers control to the routine, registers 1 through 12 are restored. If control is passed to the ERROR address from a processing macro instruction (such as GET and WRITE NEWKEY), the address of the instruction following the macro call is found in register 14. Register 14 is restored to the value in the register preceding the transient call (OPEN and CLOSE). Register 0 contains the following information concerning the reasons for the error.

- Bytes 0 and 1 reflect the first and second hardware sense bytes if the error is a hardware error; otherwise, these bytes contain binary 0's.
- Byte 2 contains a one-character EBCDIC value indicating in what section of processing the error occurred, where 4 = transients and 1 = processing.
- Byte 3 contains a one-character EBCDIC value indicating the reason for the error; refer to Appendix E.

Transient routines also cause an informational typeout, inserting bytes 2 and 3 as xx in standard Data Management message format (DMxx). If this keyword parameter is not specified, abort procedures are executed when a major file error occurs; the console typeout still appears.

Refer to Appendix F for further information regarding error conditions which are peculiar to ISAM.

■ Cylinder Index in Main Storage

This optional group of keyword parameters specifies that the cylinder index, or some section of it, is to reside in main storage, thus decreasing the number of disc accesses required to process the file when inserting records or doing random processing. The minimum number of bytes required to hold n cylinder index entries in main storage is: (key length + 6) times (n + 2). For example, given a key length of 28, it would require 408 bytes to hold 10 cylinder index entries in main storage. If the entire cylinder index is to reside in main storage, the number of bytes required is: (key length + 6) times (the number of prime data cylinders + 3). If there were 16 prime data cylinders, it would take 646 (that is, 34 times 19) bytes.

The symbolic name of the main storage area assigned to hold the cylinder index is specified as

INDAREA=symbol

The number of bytes (n) available for the cylinder index is specified as

INDSIZE=n

If n is large enough so that the entire cylinder index can be held in main storage, the file can be processed in any sequence without loss of efficiency. If n is not large enough, the file should be referenced in key-order sequence to take full advantage of the resident cylinder index section. The number of bytes required to hold the entire cylinder index in main storage is available to the user program, each time that the file is closed, in a two-byte field addressed by the concatenated label filenameS (refer to 4.2.2).

■ Input/Output Areas

Depending on the IOROUT and TYPE keyword parameters, one, two, or three input/output area keyword parameters must be specified. Each I/O keyword parameter is specified as

IOAREAx=symbol

where x is:

L for loading or adding to the file.

R for retrieving and updating in a random order.

S for retrieving and updating in sequential order.

Symbol is the address of the I/O area. Table 4–5 indicates which I/O areas must be specified for the various combinations of parameters, and the minimum size requirements for each.

Note that the size of the I/O area must be at least 96 bytes, because the OPEN transient routines use this area for reading the 96-byte format 2 label data.

IOROUT	TYPE	RCFM	I/O AREA		WORK AREA			NOTES
			KEYWORD	AREA SIZE	REQ'D?	KEYWORD	AREA SIZE	
LOAD or RELOAD		FIXBLK	IOAREAL= symbol	8+KEYLEN+ RCSZxNRECDS	YES	WORKL= symbol	RCSZ	
		FIXUNB		8+KEYLEN+ RCSZ			KEYLEN+ RCSZ	
ADD or ADDRTR		FIXBLK		8+KEYLEN+ RCSZxNRECDS			RCSZ	3
		FIXUNB		8+KEYLEN+ RCSZ+10			KEYLEN+ RCSZ	
RETRVE or ADDRTR	SEQNTL or RANSEQ	FIXBLK	IOAREAS= symbol	RCSZ xNRECDS	Not if IORG spec'f'd	WORKS= YES	RCSZ	1, 2
		FIXUNB		KEYLEN+ RCSZ+10			KEYLEN+ RCSZ	2
	RANDOM or RANSEQ	FIXBLK	IOAREAR= symbol	RCSZxNRECDS		WORKR= symbol	RCSZ	1
		FIXUNB		RCSZ+10				

NOTES: (1) I/O area size must be equal to the largest of: RCSZ+10, RCSZxNRECDS, or 96.

(2) The individual GET and PUT imperative macro instructions specify the work areas to be used.

(3) I/O area size must be equal to the largest of: 8+KEYLEN+RCSZxNRCEDS, 8+KEYLEN+RCSZ+10, or 96.

Table 4–5. I/O and Work Area Requirements for ISAM Files

- Current Record Pointer

  When records are referenced in the I/O areas rather than in the work areas, the general register to be used to store the address of the I/O area must be specified by the keyword parameter

  IORG=(r)

  where (r) is the number of the general register (2 through 12).

- File Processing Function

  The processing to be performed on the file must be specified by the keyword parameter

  $$\text{IOROUT}=\left\{\begin{array}{l}\text{LOAD}\\\text{RELOAD}\\\text{ADD}\\\text{RETRVE}\\\text{ADDRTR}\end{array}\right\}$$

  where: LOAD indicates that either a new file is created or an old file is extended; RELOAD indicates that an existing file is being re-created in the same disc space (the file identifier is the same); ADD indicates that new records are inserted into a file; RETRVE indicates that records are retrieved (and/or updated) either randomly or sequentially; and ADDRTR indicates that both the ADD and the RETRVE functions are performed.

- Expanded I/O Area

  If increased throughput is desired when insertions are to be performed, it may be achieved by allocating more than the minimum number of bytes to the IOAREAL area. The total number of bytes is specified by the optional keyword parameter

  IOSIZE=n

  where n is the total number of bytes available in the IOAREAL area. It must specify enough space to contain a prime data track in IOAREA. That is, it must be large enough to contain MXDB*(BDB+8) bytes (see H.3.2). The maximum number of bytes for the UNIVAC 8411 disc is 3613, and the maximum number of bytes for the UNIVAC 8414 and 8424 discs is 7257.

- Retrieval Search Argument

  When retrieval functions based on record key (random retrieval or sequential retrieval starting with a specific key or key group) are to be performed, the address (symbol) in the problem program of a field which contains the key to be used as a search argument must be specified as

  KEYARG=symbol

- Key Length

  All keys in an ISAM file are the same length. The key length, in bytes, for the file must be specified as

  KEYLEN=n

where n is greater than 2 and less than 256. Note, as illustrated in Figure 4–5, the key length is a part of record size for blocked records, which have embedded keys, but is an addition to record size for unblocked records, which have separated keys.

- Key Location

  If blocked records are to be processed, the location of the key field within all records of the file must be the same, and is specified as

  KEYLOC=n

  where n is the location, counting from 1, within each record of the high order byte of the key field. For example, if the key field is in bytes 17 through 24 of each record, KEYLOC=17 is specified.

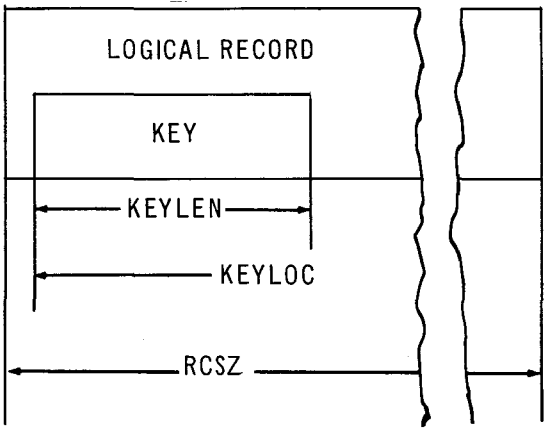
- Master Index

  If a master index is to be constructed for a file being loaded, the keyword parameter is specified as

  MSTIND=YES

  The master index immediately precedes and is in the same disc extent as the cylinder index.

A. Blocked Record Format



B. Unblocked Record Format

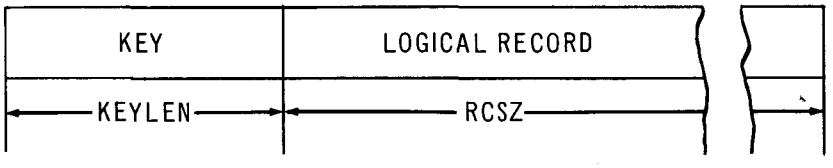


Figure 4–5. Prime Data Record Formats for ISAM Files

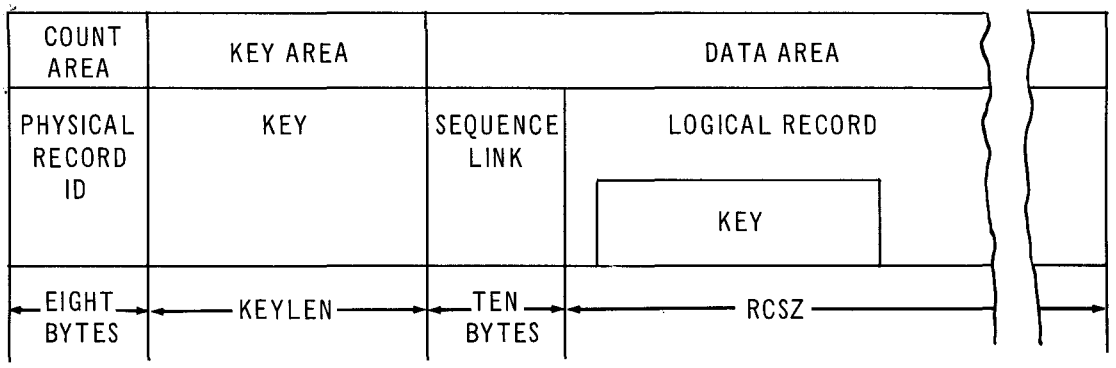
- Blocking Factor

  If blocked records are to be processed, the blocking factor (that is, the number of logical records per physical block) must be specified by the keyword parameter

  NRECDS=n

  where n is the blocking factor and is greater than 0 and less than 256.

A. Overflow Record Format, Blocked Records



B. Overflow Record Format, Unblocked Records

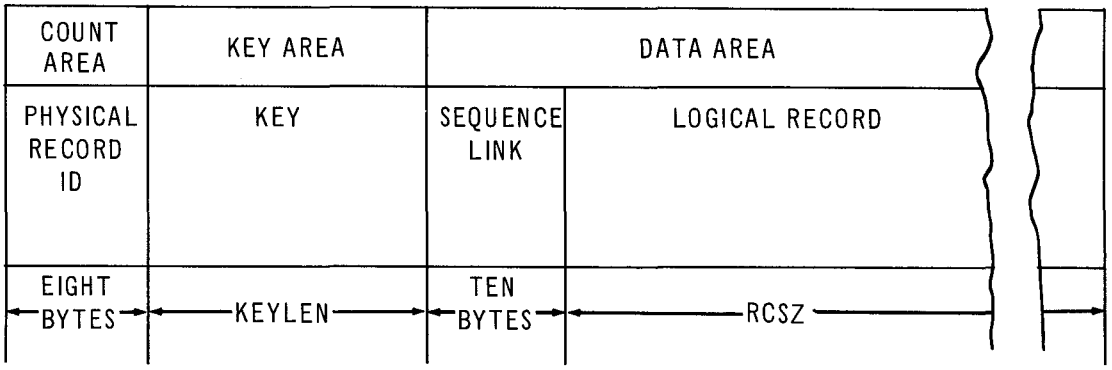


Figure 4–6. Overflow Record Formats for ISAM Files

If this keyword parameter is omitted, a blocking factor of 1 will be assumed. This parameter is ignored if unblocked records are to be processed.

- Record Format

Records in an ISAM file are fixed in length and their format must be defined as either blocked or unblocked. The required keyword parameter for this is specified as

RCFM=FIXUNB (for unblocked records)

or

RCFM=FIXBLK (for blocked records)

Figure 4–5 illustrates the differences between the two formats: blocked records have embedded keys; unblocked records have separate keys. Figure 4–6 shows the differences between the two overflow record formats.

- Record Size

  Records in an ISAM file are fixed in length and must be specified by the keyword parameter

  RCSZ=n

  where n is the number of bytes in each record. As shown in Figure 4–5, record size includes the key length for blocked records, but not for unblocked records.

- File Processing Type

  For ISAM files on which retrieval functions are to be performed (IOROUT specified as RETRVE or ADDRTR), the type of processing, random or sequential, must be specified by the TYPE keyword parameter as follows:

  – TYPE=RANDOM

    if random processing is to be performed,

  – TYPE=SEQNTL

    if sequential processing is to be performed, or

  – TYPE=RANSEQ

    if both random and sequential processing are to be performed.

- Verification Requirements

  The following optional keyword parameter requests that a parity check be made of data records after they have been written on the device:

  VERIFY=YES

  Data Management reads back one byte of data and checks for a successful completion to verify that the record transferred was written correctly. The hardware cyclic check feature establishes if the transfer was completed successfully. Verification of records necessarily increases the execution time for the commands associated with the WRITE macro instruction.

- Work Area

  Depending on the IOROUT and TYPE keyword parameters, one, two, or three work area keyword parameters must be supplied. These are specified as

  – WORKL=symbol,

    if the file is to be loaded, reloaded, or added to, and where symbol is the address of the work area;

  – WORKR=symbol,

    if records are to be retrieved (and updated) randomly, and where symbol is the address of the work area; or,

– WORKS=YES,

if records are to be retrieved (and updated) sequentially, and where each GET and PUT imperative macro instruction will provide the address of a work area.

Table 4–5 indicates when each keyword must be specified and what the minimum work area size requirements are.

Following is a summary (Table 4–6) of all the keyword parameters for the DTFIS declarative macro instruction.

KEYWORD	SPECIFICATION	FILE PROCESSING FUNCTION				REMARKS
		LOAD RELOAD	ADD	RAN RTR	SEQ RTR	
CLOSE	(NOWRITE)			X	X	Format 2 label is not rewritten; DL$TD2 and DL$NF2 are not updated.
	(DISPLAY)	X	X	X	X	Causes console display of fields addressable by the filename linked with the proper letter suffix.
CYLOFL	n=number of tracks	X				Number of tracks per prime cylinder for cylinder overflow area
DEVICE	8411, 8414, or 8424	X	X	X	X	Specifies device type
EOFA	symbolic label				X	Address of user's end-of-file routine
ERROR	symbolic label	X	X	X	X	Address of user's unrecoverable error routine
INDAREA	symbolic label		X	X		Address of cylinder index residence area
INDSIZE	n=number of bytes		X	X		Size of cylinder index residence area
IOAREAL	symbolic label	R	R			I/O area for loading or adding
IOAREAR	symbolic label			R		I/O area for random processing
IOAREAS	symbolic area				R	I/O area for sequential processing
IORG	(r)=general register			X	X	Pointer register when records are to be processed in I/O area
IOROUT	ADD		Y			Insertions are to be performed
	ADDRTR		Y	Y	Y	Insertions and retrievals are to be performed
	LOAD RELOAD	R				File is to be created or extended
	RETRVE			Y	Y	Retrievals are to be performed

Table 4–6. Summary of Keyword Parameters for the DTFIS Macro Instruction (Part 1 of 2)

KEYWORD	SPECIFICATION	FILE PROCESSING FUNCTION				REMARKS
		LOAD RELOAD	ADD	RAN RTR	SEQ RTR	
IOSIZE	n=number of bytes in I/O area		X			Size of I/O area if greater than minimum
KEYARG	symbolic label			R	X	Address of key-search argument
KEYLEN	n=number of bytes	R	R	R	R	Key length
KEYLOC	n=position of key	X	X	X	X	Location of key in blocked record
MSTIND	YES	X	X	X	X	Master index is used for file
NRECDS	n=blocking factor	X	X	X	X	Records per block
PCYLOFL	nn=a percentage declaration less than 99 and greater than or equal to 0.	X				Number of tracks reserved for cylinder overflow as a percentage of the number of tracks per cylinder.
RCFM	FIXBLK	Y	Y	Y	Y	Blocked format
	FIXUNB	Y	Y	Y	Y	Unblocked format
RCSZ	n=number of bytes	R	R	R	R	Logical record size
TYPE	RANDOM			Y		Random retrievals are to be performed
	RANSEQ			Y	Y	Random and sequential retrievals are to be performed
	SEQNTL				Y	Sequential retrievals are to be performed
VERIFY	YES	X	X	X	X	All write operations are check-read
WORKS	YES				X	Records to be processed in work areas specified by GET and PUT macro instructions
WORKL	symbolic label	R	R			User supplies records in work area
WORKR	symbolic label			X		Records to be processed in work area

LEGEND:

RAN RTR - Random retrieval
SEQ RTR - Sequential retrieval
R - Required
X - Optional
Y - One option required

Table 4-6. Summary of Keyword Parameters for the DTFIS Macro Instruction
(Part 2 of 2)

4.2.2. Filename Field Addressing

When linked with prescribed suffix letters, the symbolic name of the file (filename) is available to the user program for addressing certain fields within the file table generated by the DTFIS declarative macro instruction. For example, linking filename with the suffix letter C (filenameC) addresses the error and status indicators in the file table. Table 4–7 summarizes all the fields that can be addressed, their byte size, and whether or not they can be saved by the program. Further explanations on the use of these fields are given in the descriptions of the ISAM imperative macro instructions and in Appendix F.

SUFFIX LETTER	FIELD DESCRIPTION	TYPE (4)	LENGTH IN BYTES	SAVED (1)	FILE PROCESSING FUNCTION			
					LOAD RELOAD	ADD	RAN RTR (2)	SEQ RTR (3)
A	Number of cylinders which have full cylinder overflow areas	H	2	Yes		■		
C	Error and status indicators	H	2	No	■	■	■	■
G	Disc address from which record was retrieved (mbbcchhr)	X	8	No			■	■
H	Disc address to which record was/will be transferred (mbbcchhr)	X	8	No	■	■		
I	Number of unfilled tracks in the independent overflow area	H	2	Yes		■		
O	Total number of overflow records	H	2	Yes		■		
P	Total number of prime data records	F	4	Yes	■	■		
R	Number of overflow records retrieved which were not first in chain	X	3	Yes			■	
S	Number of bytes required to hold entire cylinder index in main storage	H	2	Yes	■ (5)	■	■	■
T	Number of records tagged for deletion by the user program	H	2	Yes			■	■

NOTES:

(1) The fields indicated as being saved are preserved in the disc format 2 label when the file is closed, and are initialized from the format 2 label when the file is reopened.

(2) RAN RTR - Random retrieval

(3) SEQ RTR - Sequential retrieval

(4) H - Halfword, F - Fullword, X - Hexadecimal

(5) Available only after CLOSE.

Table 4–7. DTFIS Fields Addressable by User Programs

4.2.3. Imperative Macro Instructions

The following paragraphs describe in detail the imperative macro instructions available for processing ISAM files. Five groups of macro instructions are described in accordance with the file processing functions involved. These are:

- Basic macro instructions: OPEN and CLOSE.
- File loading and file extending macro instructions: SETFL, WRITE NEWKEY (see 4.2.3.2.2), and ENDFL.
- Record insertion macro instructions: WRITE NEWKEY (see 4.2.3.3.1) and WAITF.
- Random processing macro instructions: READ KEY, WRITE KEY, and WAITF.
- Sequential processing macro instructions: SETL, GET, PUT, and ESETL.

In addition to using general registers 0 and 1, all the imperative macro instructions assume the presence of a 72-byte user save area, the address of which must be stored in register 13. The contents of general registers 2 through 12 are always placed in the save area when control is transferred from the user program to the file processor, and are restored when control reverts to the user program.

A responsibility of the user program is to check, after each imperative macro instruction (except OPEN and CLOSE), the two-byte field in the file table labeled filenameC for error or exceptional conditions. (Refer to Appendix F.)

4.2.3.1. Basic Macro Instructions

The basic macro instructions, OPEN and CLOSE, are used in all file processing operations. The OPEN macro instruction initializes the file processing while the CLOSE instruction terminates the file processing. The term "new file" refers to a file that has never been successfully closed and which therefore has a null format 2 label.

4.2.3.1.1. OPEN Macro Instruction

This instruction must be used to initialize the file before any other imperative macro instructions can be performed. It calls a transient routine which performs required initialization operations, such as standard label processing and validation of extent information. Depending upon whether or not it is a new file, disc format 1 and disc format 2 labels are either created or checked. All volumes of the file must be available for use at OPEN time.

For each volume, the OPEN transient routine executes the following steps:

- Checks the volume 1 (VOL1) label;
- Checks or completes the disc format 1 label;
- Saves and validates file-descriptive information, as specified in the format 1 label.

For the first volume of the file, the OPEN transient routine saves and validates file-descriptive information, as specified in the format 2 label.

The format of the OPEN macro instruction is:

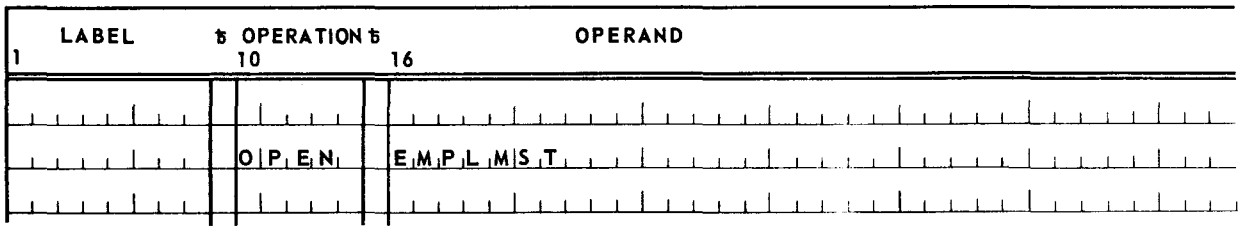
LABEL	OPERATION	OPERAND
[name]	OPEN	{ filename filename-1,filename-2,...,filename-n (1) }

POSITIONAL PARAMETER 1

filename – is the label(s) of the corresponding DTFIS declarative macro instruction(s) in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

Example:



4.2.3.1.2. CLOSE Macro Instruction

This instruction must be used to terminate processing of the file. The CLOSE macro instruction calls on a transient routine which performs required termination operations, such as updating the format 2 label. Once closed, no other macro instructions can be executed for the file until it is reopened by the OPEN macro instruction.

The format of the CLOSE macro instruction is:

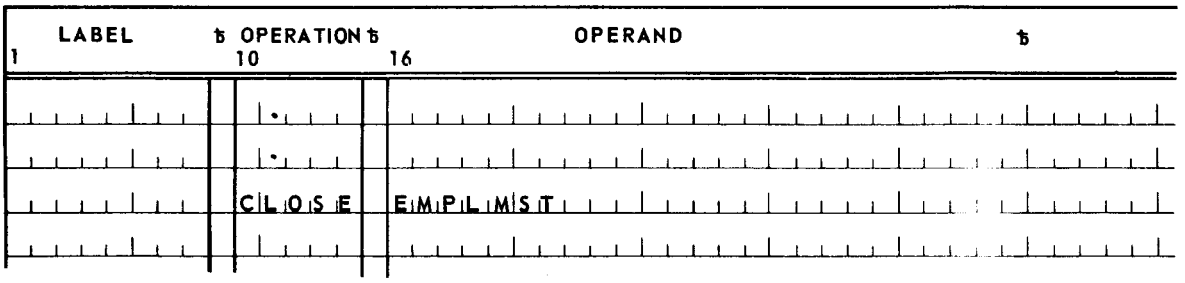
LABEL	OPERATION	OPERAND
[name]	CLOSE	{ filename (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

Example:



If desired, the user program may now access filenameS in the DTFIS addressable file table to determine the number of bytes required to hold the entire cylinder index in main storage (refer to Table 4–7).

4.2.3.2. File Loading, File Reloading, and File Extending Macro Instructions

Whether a new ISAM file is to be loaded (created), or an existing one is to be extended by adding records at the end, the file processing functions are the same. Both functions are indicated in the DTFIS declarative macro instruction by equating the IOROUT keyword to LOAD. If the file is being reloaded (re-created) the IOROUT keyword is equated to RELOAD. Records for the file are supplied in a work area in either the blocked or unblocked format (see Figure 4–5). The imperative macro instructions are the same in either case. But the two processing functions are differentiated by an indicator in the disc format 2 label. Once a file has been loaded successfully and a CLOSE macro instruction has been executed for it, the indicator is set so that subsequent processing of the file with IOROUT=LOAD will extend, rather than create, the file. IOROUT=RELOAD will re-create the file in the same disc extents from the beginning of prime data.

Three imperative macro instructions are used: SETFL initiates the processing sequence, WRITE NEWKEY adds a record to the file, and ENDFL terminates the processing sequence. After each WRITE NEWKEY instruction, filenameP (see 4.2.2) is available; filenameS is not available until after CLOSE.

4.2.3.2.1. SETFL Macro Instruction

The SETFL (set file load) macro instruction calls on a transient routine which sets up controls in the DTFIS declarative macro, and in the indexes on the direct access storage device, to prepare the file for loading (or extending). Specifically, the transient routine performs the following tasks: determines whether there is sufficient index space to support the prime data area; performs the division of the index extent between the master index and cylinder index tracks; establishes the size of the track index and the configuration of each prime data cylinder; and preformats the last track of the track index for each cylinder of the prime data area.

The format of the SETFL macro instruction is:

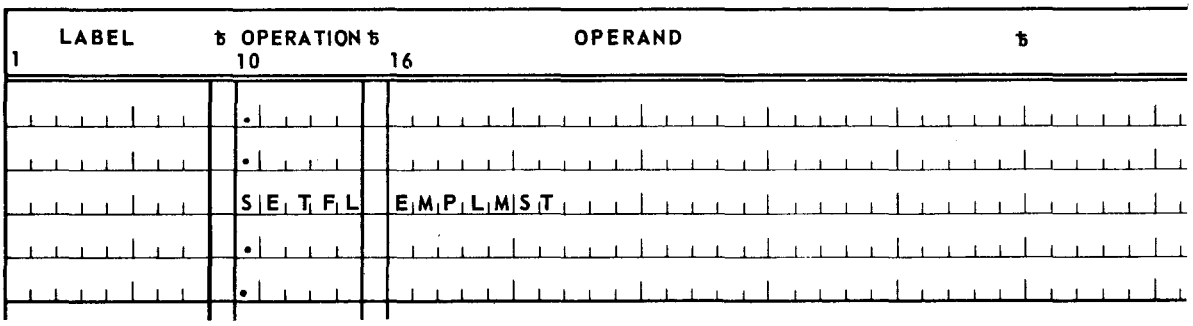
LABEL	OPERATION	OPERAND
[name]	SETFL	{ filename (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

Example:



4.2.3.2.2. WRITE NEWKEY Macro Instruction

The WRITE NEWKEY instruction (that is, WRITE is the operation code and NEWKEY is positional parameter 2) causes a logical record to be written to a file being loaded or extended. Specifically, it causes a record to be transferred from the WORKL working storage area to the IOAREAL I/O area. Before issuing the WRITE NEWKEY macro instruction, the problem program must have stored the logical record and its key in the WORKL area in either of the formats illustrated in Figure 4–5. The following actions are performed in response to this macro instruction:

- The key in the work area is checked against the key of the last record transferred into the I/O area, and if it is not greater, either a duplicate key or a sequence check error has occurred. The appropriate indication is set in the filenameC field of the DTFIS addressable file table (Table 4–7). Control returns to the problem program at the instruction address immediately following the WRITE NEWKEY macro instruction. The record in error is not transferred to the I/O area, and normal processing may be resumed with another valid logical record.

- The record and its key are transferred from the work area to the I/O area.

- If the record format is blocked, the key of the record is also transferred to the key area of the I/O area.

- When the I/O area is full, a physical record is written into the prime data area on the direct access storage device. A physical record consists of count, key, and data areas, as illustrated in Figure 4–7 for both blocked and unblocked records.

- If a physical record is written, a check is made for an end-of-track condition. If this condition exists, two track index entries (normal and overflow) are written. If the filled track was the last prime data track of the current cylinder, a cylinder index entry is written. If the file has a master index, and if the cylinder index entry filled the current cylinder index track, a master index entry is written.

- Following the WRITE NEWKEY macro instruction, the disc address (ID) of the physical record into which the logical record was written (or will be written if the I/O area is not yet full) is available in an eight-byte DTFIS addressable field labelled filenameH (see Table 4–7). The ID is in the form mbbcchhr, as described for the SEEKADR keyword parameter of the DTFDA macro instruction (see 4.1.1). Also available is the four-byte count of the total number of logical records in the prime data area, as contained in filenameP of the DTFIS table.

The format of the WRITE NEWKEY imperative macro instruction for loading or extending the file is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	WRITE	{ filename / (1) } , NEWKEY

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

NEWKEY – indicates that a new record is to be written into an ISAM file.

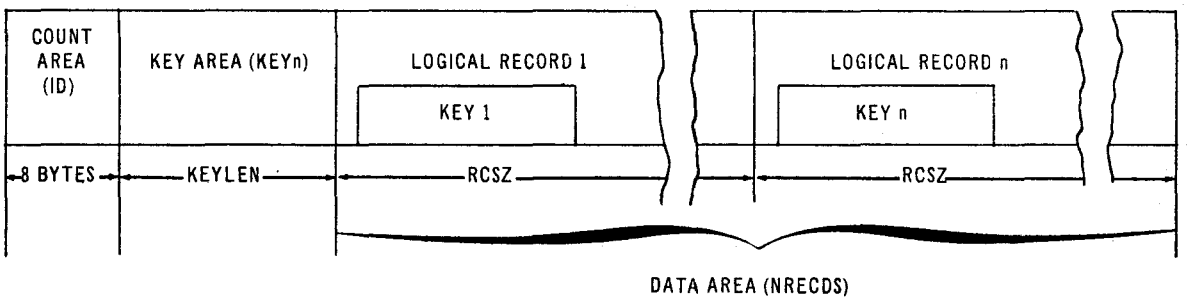
Examples:

```
LABEL      ♭ OPERATION ♭      OPERAND                    ♭
1                10                16
                 WRITE     EMPLMST,NEWKEY
                 WRITE     (1),NEWKEY
```

NOTES: (1) Assuming register 1 has been preloaded with the address of EMPLMST and one of the two forms of the WRITE NEWKEY macro instruction has been executed successfully, the eight-byte field in the DTFIS addressable file table labelled EMPLMSTH will hold the ID of the physical record area containing the new record.

(2) If an error or warning condition has occurred, the two-byte field in the file table labelled EMPLMSTC will contain an indication of the condition.

A. IOAREAL Area for Blocked Records



B. IOAREAL Area for Unblocked Records

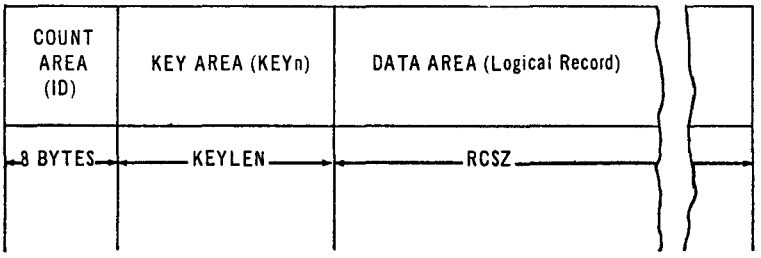


Figure 4–7. IOAREAL Area Formats for Loading or Extending ISAM Files

4.2.3.2.3. ENDFL Macro Instruction

The ENDFL (end file load) macro instruction calls on a transient routine which terminates the file loading or extending functions for the file. Any remaining unwritten records, together with an end-of-file record, are written on the disc. Also, any required index processing is performed.

The format of the ENDFL macro instruction is:

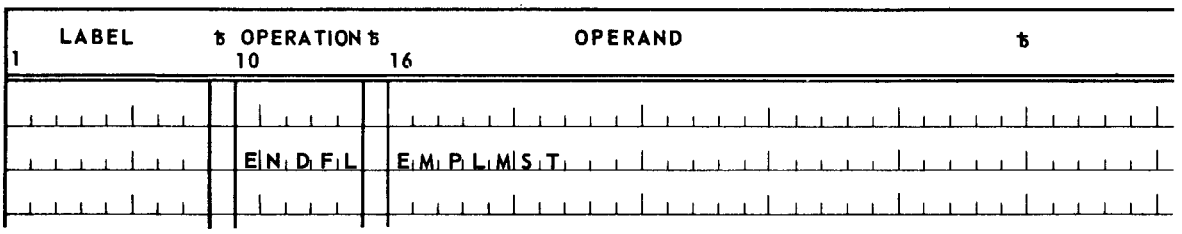
LABEL	♭ OPERATION ♭	OPERAND
[name]	ENDFL	{ filename (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

Example:



4.2.3.3. Record Insertion Macro Instructions

Once an ISAM file has been created, new records can be added to it. Each new record is inserted into the proper sequential place in the file according to its key. This file processing function is indicated in the DTFIS declarative macro by equating the IOROUT keyword to the ADD parameter (or to the ADDRTR parameter if retrieval functions are to be performed also).

Records for insertion are supplied in the same way as when the file was created; that is, in a work area in either the blocked or unblocked format. The DTFIS keyword parameters describing the records (KEYLEN, KEYLOC, NRECDS, RCFM, AND RCSZ) must have the same specifications as when the file was originally created.

The imperative macro instructions, WRITE NEWKEY and WAITF, are used to add records to an ISAM file. The form of the WRITE NEWKEY instruction for adding to the file is the same as that used for loading or extending the file, although the functions performed are different. However, when inserting records in a file, WRITE NEWKEY must be followed by WAITF. The WAITF macro instruction should not be issued when loading a file.

In the course of adding records to the file, cylinder and/or overflow areas will become filled as records are displaced from prime data areas into overflow areas. After each WAITF instruction, three user-program addressable fields in the DTFIS file table reflect the following changes in the file:

- filenameA: a two-byte counter of the number of prime data cylinders having full cylinder overflow areas; counter is set to 0 if cylinder overflow is not used.

- filenameI: a two-byte counter of the number of unfilled tracks in the independent overflow area; counter is set to 0 if the file has no independent overflow area.

- filenameO: a two-byte counter of the total number of overflow records (in cylinder and/or independent overflow areas).

In addition to these fields, filenameP, a four-byte counter of the total number of prime data records, is available.

4.2.3.3.1. WRITE NEWKEY Macro Instruction

This macro instruction inserts a new record into an existing file. Before issuing the WRITE NEWKEY macro instruction, the problem program must have stored the logical record and its key in the WORKL working storage area in either of the formats illustrated in Figure 4–5. The following actions are performed in response to this macro instruction:

- A search through the indexes is made to locate the new record's prime data track. The track is then searched to locate the physical record area into which the new record is to be inserted.
- The track is rewritten, inserting the new record into its correct position and shifting the other records toward the end of the track.
- The last logical record displaced by the shifting process is written into the overflow area associated with the prime data cylinder (that is, into the cylinder overflow area if possible; otherwise, into the independent overflow area). In either case, the record is inserted into a linked sequence of overflow entries, the beginning of which is pointed to by the overflow entry in the track index.
- The normal track index entry is adjusted to reflect the new highest key in the prime data track.

To ensure that all the actions initiated by the WRITE NEWKEY macro instruction have been completed, a WAITF macro instruction must be executed. When control is returned from the latter, the WORKL area is available for further insert records. The record just inserted is no longer in WORKL.

The format of the WRITE NEWKEY macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	WRITE	{ filename / (1) },NEWKEY

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

NEWKEY – indicates that a new record is to be written into an ISAM file.

Examples:

```
LABEL      ♭ OPERATION ♭      OPERAND                    ♭
1                 10          16
          WRITE    EMPLMST,NEWKEY
          WRITE    (1),NEWKEY
```

NOTES: (1) Assuming register 1 has been preloaded with the address of EMPLMST and one of the two forms of the WRITE NEWKEY macro instruction, followed by a WAITF macro instruction, has been executed successfully, the eight-byte field in the DTFIS addressable file table labelled EMPLMSTH will hold the disc address (ID) of the physical record area containing the new record.

(2) If an error or warning condition has occurred, the two-byte field in the file table labelled EMPLMSTC will contain an indication of the condition.

4.2.3.3.2. WAITF Macro Instruction

This macro instruction ensures that the transfer of a record between main storage and a direct access storage device has been completed. It must be issued before the program attempts to process another record. Any exceptional (error or status) conditions detected during the execution of the WAITF instruction are reflected in the DTFIS filenameC field when control is returned to the problem program.

The format of the WAITF macro instruction is:

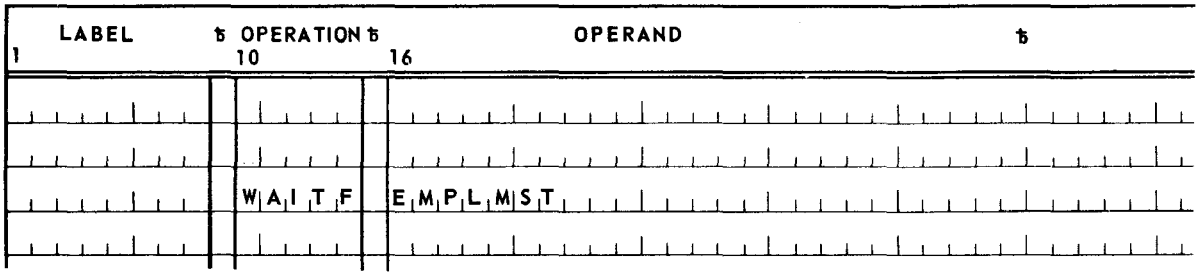
LABEL	♭ OPERATION ♭	OPERAND
[name]	WAITF	{ filename / (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

Example:



4.2.3.4. Random Processing Macro Instructions

Individual logical records can be retrieved in random order for processing and updating. The record to be retrieved from the file is designated by its key, and, in the case of an updating operation, is written back into the file.

The random retrieval (and updating) file processing function is indicated in the DTFIS macro instruction by equating the IOROUT keyword to RETRVE (or ADDRTR if new record insertions are also to be performed), and the TYPE keyword to RANDOM (or RANSEQ if sequential processing functions are also to be performed).

Three imperative macro instructions are used in the random processing of an ISAM file: READ KEY, WRITE KEY, and WAITF. The WAITF instruction has the same significance as described in 4.2.3.3.2.

4.2.3.4.1. READ KEY Macro Instruction

This macro instruction initiates the retrieval of a single logical record from an ISAM file. Before issuing the instruction, the problem program must have stored the key of the record to be retrieved in the main storage area equated to the KEYARG keyword parameter of the DTFIS declarative macro instruction. In response to the READ KEY macro instruction, the hierarchy of indexes is searched to determine whether the record is to be retrieved from a prime data track or from an overflow chain. When this has been determined, the appropriate track or chain is searched and the physical record containing the desired logical record is moved into the I/O area equated to the IOAREAR keyword.

If the WORKR keyword was specified in the DTFIS macro instruction, the logical record is moved into the work area equated to WORKR. If the IORG keyword was specified, the address of the first character of the logical record is placed in the register (2 through 12) equated to IORG. In either case, the disc address from which the record was retrieved is available to the problem program in the eight-byte DTFIS addressable field labelled filenameG (Table 4–7).

To ensure that the retrieval operation has been completed, the problem program must execute a WAITF macro instruction before attempting to access the logical record retrieved.

→ The format of the READ KEY macro instruction is:

LABEL	♿ OPERATION ♿	OPERAND
[name]	READ	{ filename / (1) },KEY

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

KEY – indicates that a random retrieval (by key) is to be performed.

Example:

```
LABEL     ♿ OPERATION ♿     OPERAND                    ♿
1         10                 16
          READ               EMPLMST,KEY
          WAITF              EMPLMST
```

NOTES: (1) When control returns to the problem program after the execution of the WAITF macro instruction, the logical record associated with the key in the KEYARG area is available either in the WORKR area or in the I/O area, depending on the DTFIS declarative macro specifications. In the latter case, the register equated to the IORG parameter contains the address of the first character of the logical record. The disc address of the physical record is available at address EMPLMSTG in the DTFIS file table. Indications of any exceptional (error or status) conditions are available at address EMPLMSTC.

(2) If the record was retrieved from an overflow area, and if it was not the first record in the overflow chain for the associated prime data track, the three-byte non-first-overflow record reference count at address EMPLMSTR is incremented by 1. This count is maintained over the life of the file, and is made available to the problem program as a possible aid to the user in deciding when to reorganize the file.

4.2.3.4.2. WRITE KEY Macro Instruction

This instruction initiates the rewriting (updating) of the last record retrieved with a READ KEY macro instruction. If the record is in blocked format, with an embedded key (Figure 4–5), the key field must not be altered in any way by the user program. The record to be rewritten is determined by the key that was supplied in the KEYARG area when the last READ KEY instruction was issued. The current contents of the KEYARG area are ignored. In response to the WRITE KEY instruction, the updated logical record from the WORKR area is moved to the correct location in the I/O area, and the record is rewritten. (If the record was updated in the I/O area by use of the IORG address, no move is required since the updated record is already in the correct location in the I/O area.) Before further processing of the file is attempted, the problem program must issue a WAITF macro instruction to ensure the completion of the rewrite operation.

The format of the WRITE KEY macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	WRITE	$\left\{\begin{matrix}\text{filename}\\(1)\end{matrix}\right\}$,KEY

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

KEY – indicates that the last record retrieved by a READ KEY macro instruction is to be rewritten in the file.

Example:

```
LABEL      ♭ OPERATION ♭      OPERAND                    ♭
1               10            16
                WRITE     EMPLMST,KEY
                .
                WAITF     EMPLMST
```

NOTES: (1) When control returns to the problem program after the execution of the WAITF macro instruction, the logical record which was last retrieved by a READ KEY instruction, as well as its physical record, have been rewritten onto the file. Any indications of exceptional (error or status) conditions detected during the write and wait operations are available at address EMPLMSTC in the DTFIS file table.

(2) If the problem program has, according to the user's own conventions, tagged the record for deletion, the two-byte tagged-for-deletion count at address EMPLMSTT in the DTFIS file table can be incremented by the program either before or after the rewrite operation. This count is maintained over the life of the file, and is made available to the problem program as a possible aid to the user in deciding when to reorganize the file.

4.2.3.5. Sequential Processing Macro Instructions

Logical records can also be retrieved and updated sequentially. The first record to be retrieved can be designated by the beginning of the file, by a physical record disc address, by a specified key, or by any key greater than or equal to a specified value. The SETL macro instruction specifies which kind of starting point is desired. Individual records are then retrieved in sequence by the GET macro instruction. Where an updating operation is to be performed, the individual records are rewritten into the file by means of the PUT macro instruction.

The sequential retrieval (and updating) file processing function is indicated in the DTFIS declarative macro by equating the IOROUT keyword parameter to RETRVE (or ADDRTR if new record insertions are also to be performed), and the TYPE keyword to SEQNTL (or RANSEQ if random processing functions are also to be performed).

To terminate a retrieval sequence, an ESETL macro instruction is issued. This ensures that any logical records committed to output by the PUT macro instruction are written onto the direct access storage device. After the ESETL instruction has been executed, another retrieval sequence can be initiated by executing a SETL macro instruction. However, if RANSEQ was specified for the TYPE keyword in the DTFIS declarative macro, the READ KEY and WRITE KEY macro instructions may be issued.

4.2.3.5.1. SETL Macro Instruction

This instruction initializes a retrieval sequence. It specifies the file from which the records are to be retrieved and the point at which the retrieval is to start. For the starting point, the SETL macro instruction can specify any of the following:

- The label of an eight-byte ID area containing the disc address (ID) of the first physical record to be retrieved;
- A register containing the address of an eight-byte ID area;
- That the retrieval sequence is to start with the first logical record of the file;
- That the area equated to the KEYARG keyword parameter contains the key of the first logical record to be retrieved;
- That the KEYARG area contains a value against which the key of the first logical record to be retrieved must test greater than or equal to.

When control is returned from the SETL macro instruction, the retrieval sequence can begin.

The format of the SETL macro instruction is:

LABEL	OPERATION	OPERAND
[name]	SETL	{ filename / (1) } , { idname / (r) / BOF / GKEY / KEY }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

idname – is the label of an eight-byte area containing the ID (disc address in the form MBBCCHHR) of the first record.

(r) – indicates that the designated general register (2 through 12) has been preloaded with the address of an eight-byte area containing the ID, as described for idname.

BOF – indicates that the retrieval sequence is to begin with the first logical record of the file.

GKEY – indicates that the retrieval sequence is to start with the first logical record whose key is greater than or equal to the value in the area equated to the KEYARG keyword parameter.

KEY – indicates that the area equated to the KEYARG keyword parameter holds the key of the first logical record to be retrieved.

Examples:

	LABEL (1)	OPERATION (10)	OPERAND (16)
1.		SETL	EMPLMST,(4)
2.		SETL	EMPLMST,GKEY
3.		SETL	EMPLMST,KEY

1. Register 4 contains the address of an eight-byte area containing the ID of the physical record whose first logical record will be made available to the program when the next GET macro instruction is executed. If the eight-byte ID area is outside the file limits, when control is returned following the SETL instruction, bit 4 will be set in the two-byte error indicator location specified by DTFIS addressable field EMPLMSTC. Any GET instructions attempted without an intervening and correct SETL instruction will have no effect, and will also return with bit 4 set in field location EMPLMSTC.

2. The area equated to the DTFIS keyword parameter KEYARG holds a value which is the lower key limit for retrieval. If, for example, the KEYARG area holds 400101, the retrieval sequence starts with the lowest key greater than or equal to 400101. If it is determined that all keys in the file are less than 400101, when control is returned following the SETL instruction, bit 3 will be set in addressable field location EMPLMSTC. Any GET instruction attempted without an intervening and correct SETL instruction will be null, and will also return with bit 3 set in EMPLMSTC. If, on the other hand, all keys in the file are greater than 400101, the first GET instruction will simply retrieve the first logical record with a key greater than 400101.

3. The specified KEYARG area holds the key of the first logical record to be retrieved. For example, if the area holds a key equal to 400101, the first GET instruction will retrieve the logical record whose key is 400101. If it is determined that no record in the file has this key, control is returned from the SETL instruction, with bit 3 of addressable field location EMPLMSTC set. Any subsequent GET instruction attempted without an intervening and correct SETL will have no effect, and will also return with bit 3 set in field location EMPLMSTC.

4.2.3.5.2. GET Macro Instruction

This instruction retrieves the next logical record in sequence. It must be part of a valid retrieval sequence initiated by a SETL macro instruction. If the physical record containing the next logical record is not already in main storage, the GET instruction causes it to be read into the I/O area equated to the IOAREAS keyword in the DTFIS declarative macro. (The logical record is made available to the problem program either in a work area or in the I/O area.) If the WORKS keyword of the DTFIS declarative macro was equated to YES, the GET instruction specifies the address of a work area into which the logical record is transferred. If WORKS=YES was not specified, a general register number (2 through 12) must have been equated to the DTFIS keyword IORG. In this case, execution of the GET macro instruction makes the main storage address of the logical record available in the specified register.

If the GET instruction requires the reading of a new physical record, and if a PUT macro instruction was executed for any logical record in the previous physical record, then the previous physical record, as updated, is written back onto the direct access storage device before the new record is read.

The format of the GET macro instruction is:

LABEL	♭ OPERATION ♭	OPERAND
[name]	GET	{filename / (1)} [,{workname / (0)}]

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

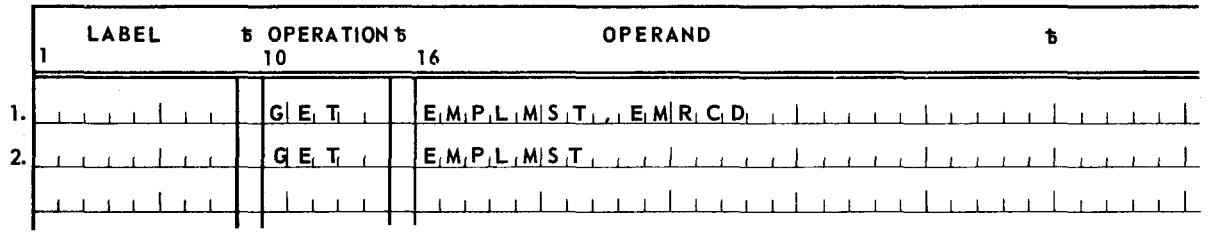
POSITIONAL PARAMETER 2

workname – is the label of the work area into which the record is to be transferred.

(0) – indicates that register 0 has been preloaded with the address of the work area into which the record is to be transferred.

Positional parameter 2 is required only if the WORKS keyword of the DTFIS declarative macro was equated to YES.

Examples:



1. The next logical record of EMPLMST is transferred into the work area labelled EMRCD. If the record format is unblocked, the key is also transferred. Any abnormal conditions are indicated by the bit settings in addressable field location EMPLMSTC. The disc address from which the logical record was transferred is available to the problem program at addressable field location EMPLMSTG.

2. The register equated to the IORG keyword parameter (for example, register 4) holds the address of the first character of the logical record. If the EMPLMST file is in the unblocked format, the key of the record is at the beginning of the I/O area equated to the IOAREAS keyword parameter. If the file is in the blocked format, the key of the logical record is in the standard location, as specified by the value equated to the KEYLOC keyword parameter. Addressable fields EMPLMSTC and EMPLMSTG have the same significance as in the preceding example.

4.2.3.5.3. PUT Macro Instruction

This instruction indicates that the last record retrieved by a GET macro instruction has been updated and is to be rewritten on the direct access storage device. It must be part of a valid retrieval sequence initiated by a SETL macro instruction and must follow a GET macro instruction. It is only through the execution of the PUT macro instruction that updating takes place in a sequential retrieval operation. If a record retrieved by a GET instruction is not updated, there is no need to execute a PUT instruction.

If the file is in the unblocked format, the record is rewritten immediately. If the file is in the blocked format, the PUT macro instruction sets an indicator in the DTFIS file control table such that the physical record containing the updated record is rewritten when the next physical record is to be accessed (that is, when all logical records in the current physical record have been processed).

Like the GET macro instruction, the PUT macro instruction has two forms: the work area form and the I/O area form. If the WORKS keyword of the DTFIS declarative macro was equated to YES, then the PUT macro instruction must specify the address of the work area from which the updated record is to be transferred to the IOAREAS area. This may be the same area as specified in the previous GET macro instruction, or it may be a different area. Note that if the record format is unblocked, only the logical record is transferred back to the I/O area.

If the WORKS keyword of the DTFIS declarative macro was not equated to YES, then a general register number (2 through 12) must have been equated to the IORG keyword parameter, and the address of the logical record was supplied in the specified register when the last GET macro instruction was executed. In this case it is assumed that the record was updated at the specified address in the IOAREAS area.

Under no circumstances may the key field of the updated record be altered.

The format of the PUT macro instruction is:

LABEL	↹ OPERATION ↹	OPERAND
[name]	PUT	{ filename / (1) } [, { workname / (0) }]

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction in the program.

(1) – indicates that register 1 has been preloaded with the address of the DTFIS macro instruction.

POSITIONAL PARAMETER 2

workname – is the label of the work area from which the record is to be transferred.

(0) – indicates that register 0 has been preloaded with the address of the work area from which the record is to be transferred.

Positional parameter 2 is required only if the WORKS keyword of the DTFIS declarative macro was equated to YES.

Examples:

```
     LABEL      ↹ OPERATION ↹      OPERAND                    ↹
1                10               16
1.               PUT              EMPLMST,(0)
2.               PUT              EMPLMST
```

1. The logical record last retrieved from EMPLMST is replaced by a record in the work area whose address is specified in register 0. If EMPLMST is in unblocked format, the physical record is rewritten from the IOAREAS area. If EMPLMST is in blocked format, an indicator in the DTFIS file control table is set to initiate rewriting of the physical record before the next record is read into the I/O area.

2. The logical record, whose address was supplied in the register specified by the IORG keyword parameter when the preceding GET macro instruction was executed, is assumed to have been updated. Refer to Note 1 for an explanation of when the physical record is rewritten.

4.2.3.5.4. ESETL Macro Instruction

This instruction terminates a retrieval sequence initiated by a SETL macro instruction. If there are any updated logical records which have not yet been rewritten, they are rewritten at this time. After the ESETL instruction has been executed, another retrieval sequence can be initiated by means of a SETL instruction.

The format of the ESETL macro instruction is:

LABEL	␣ OPERATION ␣	OPERAND
[name]	ESETL	{ filename / (1) }

POSITIONAL PARAMETER 1

filename – is the label of the corresponding DTFIS declarative macro instruction.

(1) – indicates that the address of the DTFIS macro instruction has been preloaded into register 1.

Examples:

```
LABEL     ␣ OPERATION ␣     OPERAND                    ␣
1              10        16
          ESETL     EMPLMST
          ESETL     (1)
```

4.2.3.5.5. DMEAR Macro Instruction

This instruction provides information for fatal hardware or logical errors that have taken place in opening, processing, or closing a Data Management file (3.4).

5. DISC SPACE MANAGEMENT

5.1. GENERAL

When executing jobs in a multiprogramming environment, it is always difficult (and often impossible) for the individual programmer to know the exact organization of particular direct access volumes. There is, nevertheless, a need for most jobs to create and process files on these volumes. The creation and processing of direct access files require means for allocating space, releasing unused space, scratching files when no longer needed, obtaining label and extent information, and renaming files. These procedures are required by Job Control and various service and utility programs.

Disc Space Management routines for the UNIVAC OS/4 Operating System (OS/4) provide an efficient and completely automatic space accounting and maintenance feature which relieves the user of the responsibility of knowing the precise contents of direct access volumes. These routines also permit the resolution of competing demands for allocation, and establish standard interfaces with the user, Job Control, and the service and utility programs.

The Disc Space Management routines are set up to handle individual direct access volumes. When processing multivolume files, the user is responsible for submitting all related volumes.

5.2. PROGRAMMING APPROACH

Disc Space Management in OS/4 comprises a set of transient service routines which allocate space to files on direct access volumes. This is accomplished by maintaining the volume table of contents (VTOC) through standard procedures for all files — system, temporary, and those considered permanent by the user. The VTOC is a permanently allocated, unmovable file which exists on every direct access volume by the direct access volume initialization program. The contents of the VTOC file are a format label, or set of format labels, for each file on the volume and for all unused space on the volume.

The Disc Space Management routines maintain the VTOC by creating format labels for new files and deleting format labels for files removed from the volume. When a file is to be created, unused space is found for it by searching the appropriate labels in the VTOC, allocating the space as the extents of the file, and removing it from free space. When a file is deleted, the format label for the file is removed from the VTOC; the extents previously assigned to the file are again available for allocation.

Disc Space Management includes four functional areas, two of which are concerned with basic space accounting, and two of which provide support operation capability.

- Space Accounting Routines

  ALLOCATE — allocates initial direct access space to a file.

  SCRATCH — deletes a file or portion of a file by deleting its associated extents and/or labels in the VTOC.

- Support Routines

  RENAME — changes the name of a file.

  OBTAIN — enables direct access to any label in the VTOC.

System macro instructions are provided for effecting Supervisor-assigned linkages to the ALLOCATE, SCRATCH, RENAME, and OBTAIN routines.

Disc Space Management functions are used by various components of OS/4 in response to implicit and explicit requests introduced through input job streams. For example, the space requested for a new or updated direct access file in an EXT statement included in a user's job control language file definition set is obtained for Job Control through an interface with the ALLOCATE routine. The OBTAIN function is used by the librarian routines to gain access to label and extent information required for file initialization. The ALLOCATE, SCRATCH, and RENAME functions are employed by utility programs. A SCRATCH ALL request deletes all files on a volume whose expiration date either predates or equals the present date.

5.3. DATA INPUT

The precise input to Disc Space Management is dependent upon the particular function to be exercised. The contents of the VTOC, however, serve as input (and output) to some degree for all of the Disc Space Management routines. The structure of the VTOC and the techniques of recording the use or availability of direct access space are therefore discussed in this section.

The VTOC is a file which contains a set of format labels for each file and for all unused space on a direct access volume. The extent or length of a VTOC is variable, as specified at the user's installation. The starting address of the VTOC is recorded in bytes 11—15 of the standard volume label, and can be anywhere on the volume following the IPL control record(s) and volume label(s); the starting address is recorded in the form CCHHR. This address must be that of an integral track boundary unless the VTOC starts on track 0 of cylinder 0 and shares that track with the IPL control record and volume label(s).

The VTOC extent must be only one area of contiguous tracks on one or more contiguous cylinders.

5.3.1. Format Labels

For each file contained on a direct access volume, there exists a corresponding set of format labels in the VTOC of the volume. Each such set indicates the attributes and extents of the file and may contain up to two format labels. Each physical label is 140 bytes in length; this is subdivided into a 44-byte key and 96-byte data area. Information contained in these labels is used by the Data Management routines to control access to files. In the case of multivolume direct access files, there is a set of format labels for the file in the VTOC of each volume.

There are six formats of VTOC format labels defined as follows:

- Format 1 Label

  This label identifies any file on a direct access volume except for the VTOC file itself. There must be one format 1 label for each file or part of a file on each volume. Up to three noncontiguous areas, or extents, occupied by the file can be identified in the format 1 label. One format 2 label or one format 3 label can be chained to a format 1 label.

- Format 2 Label

  This label provides additional description for an indexed sequential file. If present, the format 2 label is chained to a format 1 label.

- Format 3 Label

  This label is used if the file consists of more than three extents. Up to 13 additional extents can be identified in the format 3 label. If present, the format 3 label is chained to a format 1 label.

- Format 4 Label

  This label describes the VTOC itself. It is always the first record in the VTOC. No other labels are chained to the format 4 label.

- Format 5 Label

  This label describes up to 26 noncontiguous extents that are available for allocation on a volume. It is always the second record in the VTOC. Several format 5 labels may be chained together if more than 26 noncontiguous extents are available on the volume.

The five previously described labels are basic to the logic of Disc Space Management. See 2.3.1 for complete format descriptions of these labels.

For conceptual purposes it is helpful to think of a sixth type of VTOC format label, a format 0 label, which represents available space within the extent of the VTOC itself. The format 0 label contains all binary zeros. When a format 1, 2, 3, or 5 label is deleted from the VTOC, a format 0 label is written in its place.

5.3.2. Volume Contents Following Initialization

Before any data has been written on a direct access volume, the volume contains a single VTOC file, which is written by the disc prep routine (*UNIVAC OS/4 General Disc Prep Routine Programmer Reference, UP-8014* (current version)). Space occupied by the VTOC is described within the format 4 label. The remainder of the space on the volume (with the exception of that occupied by the IPL control record, volume labels, and IPL program, if present) is available to be allocated to files as they are created, and is described in the second record of the VTOC, the format 5 label. All other records of the VTOC are predefined as format 0 labels.

If the VTOC is recorded on cylinder 0 immediately following the last (or only) volume label, or is recorded at the end of the entire volume, the available space, after initialization, is a single area of contiguous space extending, respectively, from the end of the VTOC to the end of the volume, or from volume labels to the beginning of the VTOC.

If, on the other hand, the VTOC does not immediately follow the volume labels or appears at the end of the volume, the available space is two areas of contiguous storage, one preceding the VTOC and a second following it.

Each available extent is recorded in a separate entry in the format 5 label. The entry contains the relative address of the first track of the extent, the number of complete cylinders in the extent, and the number of tracks in addition to complete cylinders.

The total number of tracks accounted for in the VTOC is, at any time, the total number of tracks on the volume. Used tracks are recorded in format 1, 3, and 4 labels, while available tracks are recorded in format 5 labels.

5.3.3. Use/Availability Recording Technique

At the time a file is created, space is acquired by the ALLOCATE routine by scanning the format 5 labels. When space is found, the requested amount is subtracted from that described in the format 5 labels. A format 1 label is constructed for the new file and written into the VTOC at the first available location (format 0 label).

When a file is deleted, its format 1 label and format 2 or 3 label, if present, are replaced by format 0 labels. The extents previously occupied by the file are returned to available space by adding the indicated quantities back into the format 5 labels.

When a portion of a file is deleted, only specified extents are deleted from the file. If all extents are deleted from a format 3 label, the label is replaced by a format 0 label.

A file is described by a set of one or two VTOC format labels according to the type of file organization and the number of extents in the file. The following combinations are possible:

1. A single format 1 label for a nonindexed-sequential file of not more than three extents or indexed-sequential file's second and subsequent volumes.

2. A format 1 label chained to a format 3 label for a nonindexed-sequential file with more than three extents.

3. A format 1 label chained to a format 2 label for an indexed-sequential file (only the first volume of the file).

The set of VTOC format labels corresponding to a file are neither necessarily contiguous nor in a defined order within the VTOC. With the exceptions of the format 4 label and the first format 5 label, new labels are written into locations nearest the beginning of the VTOC as they are created; the relationship of any two of these labels is indicated through chain addresses.

5.3.4. Capacity of the VTOC

During the allocation process, Disc Space Management routines must take into account the availability of space for both the extents of the file on the volume and the file's corresponding labels in the VTOC. Space required for the expansion of format 5 labels is an additional consideration.

The format 4 label contains a count of the number of available slots within the VTOC at any time. This count is interrogated by Disc Space Management routines, prior to allocating space, to determine whether there is sufficient space to accommodate the required new format labels for the file. If not enough slots are available, allocation is not performed; there is no provision for dynamic extension of the VTOC.

A UNIVAC 8411 Disc Storage volume accommodates 16 140-byte VTOC format labels on a single track, and up to 160 140-byte VTOC format labels can be contained on a complete cylinder. A UNIVAC 8414 or 8424 Disc Storage volume accommodates 25 140-byte VTOC format labels on a single track, and up to 500 VTOC format labels on a complete cylinder.

5.4. IMPERATIVE MACRO INSTRUCTIONS

The imperative macro instructions supplied for Disc Space Management are ALLOC, SCRTCH, RENAME, and OBTAIN. The ALLOC and SCRTCH macros are concerned with basic space accounting (allocating and deleting file space), while RENAME and OBTAIN provide support operation availability (changing a file name and enabling access to any label). Each of these macros is discussed in detail, along with its associated error conditions, in the following paragraphs.

5.4.1. ALLOC Macro Instruction

The ALLOC macro instruction is issued to request the services of the Disc Space Management Allocate routine. Allocation is performed on a volume-by-volume basis. For each volume of a file, the inputs to the Allocate routine are the File Control Block (FCB) of the file and the corresponding Extent Request Block (ERB).

LABEL	␣ OPERATION ␣	OPERAND
	ALLOC	{ piocb-name-of-fcb / (1) }, { piocb-name-of-erb / (0) }

POSITIONAL PARAMETER 1

piocb-name-of-fcb – symbolic address of the physical I/O control block containing the FCB.

(1) – indicates that register 1 has been preloaded with the physical I/O control block address containing the FCB.

POSITIONAL PARAMETER 2

piocb-name-of-erb – symbolic address of the physical I/O control block containing the ERB.

(0) – indicates that register 0 has been preloaded with the physical I/O control block address containing the ERB.

5.4.2. SCRTCH Macro Instruction

This macro instruction is issued to request the services of the Disc Space Management Scratch routine. This routine eliminates a file from a volume, releases unused extents, or eliminates all files with expired dates from a volume.

LABEL	OPERATION	OPERAND
	SCRTCH	{param-list / (1)} [, {extent-number / ALL / (0)}]

POSITIONAL PARAMETER 1

param-list – the address of a parameter list containing the following:

6-byte volume serial number of the current volume (this is checked against the standard volume label on the disc);

44-byte file identifier; however, this file identifier is not needed when positional parameter 2 specifies ALL.

(1) – indicates that register 1 has been preloaded with the address of a parameter list.

POSITIONAL PARAMETER 2

extent-number – used for partial scratch; it indicates the last extent to be saved.

ALL – all files of the specified volume (with expired dates) will be scratched.

(0) – indicates that register 0 contains the error information established in the Allocate routine.

NOTE: The second operand is not needed for scratching entire individual files.

5.4.3. RENAME Macro Instruction

The RENAME macro instruction is issued to request the services of the Disc Space Management Rename routine. This routine changes the file identifier of an existing file format 1 label.

LABEL	♭ OPERATION ♭	OPERAND
	RENAME	{ param-list (1) }

POSITIONAL PARAMETER 1

param-list – the address of a parameter list containing the following:

44-byte old file identifier

44-byte new file identifier

6-byte volume serial number

(1) – indicates that register 1 has been preloaded with the address of the parameter list.

5.4.4. OBTAIN Macro Instruction

This macro instruction is issued to request the services of the Disc Space Management Obtain routine. This routine enables access to any user label in the VTOC. If a 44-byte filename is supplied as input, the absolute track address (in the form cchhr) of the format 1 label is placed in the first five bytes of the data area, followed by the 96-byte data area of the format 1 label. If an address ID (cchhr) is supplied as input, the entire key and data area of the specified label are returned. ←

LABEL	♭ OPERATION ♭	OPERAND
	OBTAIN	{ param-list (1) }

POSITIONAL PARAMETER 1

param-list – the address of a parameter list containing the following:

1-byte request type; if the byte contains 0, request by key; if the byte contains 1, request by ID

6-byte volume serial number; if request by key is specified, the parameter list includes a 44-byte file identifier and a 101-byte data area. If request by ID is specified, the parameter list includes a 5-byte address ID (of form cchhr) and a 140-byte data area. ←

(1) – indicates that register 1 has been preloaded with the address of the parameter list.

5.5. ERROR CODES

Disc Space Management error codes are returned to the caller in the least significant byte of register 0 for the Allocate and Scratch routines, and register 1 for the Obtain and Rename routines. If the error code is from the Allocate routine and indicates that the Scratch routine must be called, register 0 is prepared for entry to the Scratch routine with the last extent number to be saved for the file in the upper halfword of register 0.

5.5.1. Allocate Routine Error Codes

- Error Code 1. Invalid Parameter Specification

  Causes: – For a new file, the identifier already exists in the VTOC.

  – For an old file, the identifier could not be found in the VTOC.

  – The file identifier in the FCB started with binary zeros.

  – The extent request block has a count of zero.

  Result: The VTOC remains intact; no further action is required.

  User Action: Print VTOC to examine directory of file identifiers. Make sure job control stream contains an LBL card with a file identifier. Check FCB and extent request block for invalid parameter specification.

- Error Code 2. No Room in VTOC

  Causes: – There is not enough room in the VTOC to assign file labels to the user, or to allow for expansion of the VTOC file labels.

  → **– For a new file, three VTOC labels are needed.**

  – For an old file, two VTOC labels are needed.

  Result: The VTOC remains intact; no further action is required.

  User Action: Eliminate unused files by means of the Scratch routine. Also, the size of the VTOC on the disc can be expanded by use of the Disc Preparation routine.

- Error Code 3. Input/Output Error

  Causes: An unacceptable disc error has occurred while processing the VTOC.

  Result: The VTOC has been compromised. It is impossible to continue using the disc in its present state.

  User Action: Since the data records of the file could be intact, they could be copied to another disc. Also, the Disc Preparation routine could be called to reprepare and assign alternate tracks in the VTOC area only; then the file labels could be reinstated by requesting absolute allocation for required areas on the disc.

- Error Code 4. Invalid PUB or Volume Serial Number

  Causes: – No disc PUB has been found with the correct VSN.

  – The volume serial number in the Extent Request Block does not match the one in the standard volume label.

  Result: The VTOC remains intact; no further action is required.

  User Action: Check DVC card for invalid logical unit number. Also check job stream for errors in VOL cards. The proper volume serial number was probably not specified for Disc Preparation routine.

- Error Code 5. Cannot Allocate Absolute Request

  Causes: The area specified by an absolute EXT request is not available as free space on the disc.

  Result: The VTOC is not intact; Scratch routine must be called to reinstate the VTOC.

  User Action: Eliminate file using required area or alter request.

- Error Code 6. Contiguous Request Unable to be Assigned

  Causes: User specified request cannot be satisfied by any contiguous free space on the disc.

  Result: The VTOC is not intact; Scratch routine must be called to reinstate the VTOC.

  User Action: Could make the same request again, specifying noncontiguous.

- Error Code 7. More than 16 Extents

  Causes: The maximum of 16 extents assigned to the file on this volume has been reached.

  Result: The VTOC is not intact; Scratch routine must be called to reinstate the VTOC.

  User Action: Reorganize the file if possible.

- Error Code 8. No Room on Disc

  Causes: While attempting to satisfy a noncontiguous request, the Allocate routine exhausted all available free space on the disc.

  Result: The VTOC is not intact; Scratch routine must be called to reinstate the VTOC.

  User Action: The disc is full.

5.5.2. Scratch Routine Error Codes

- Error Code 1. No Room in VTOC

  Causes: **This error would only occur when a partial Scratch is specified; that is, positional parameter 2 of the SCRTCH macro instruction specifies the last extent number. One VTOC label is needed for possible expansion of the VTOC labels.**

  Result: The VTOC is intact; no further action is required.

  User Action: Could do a complete Scratch rather than a partial Scratch.

- **Error Code 2. Invalid File Identifier (Not for SCRATCH ALL)**

  Causes: – The identifier could not be found in the VTOC.

  – The file identifier in the parameter list was not in EBCDIC code.

  Result: The VTOC remains intact; no further action is required.

  User Action: Print directory of file identifier. Check parameter list for valid identifier.

- Error Code 3. Input/Output Error

  Same as described for error code 3 under Allocate Routine Error Codes.

- Error Code 4. Volume Serial Number Error

  Causes: – No disc PUB has been found for the specified VSN.

  – The volume serial number in the parameter list does not match the one in the Standard Volume Label.

  Result: The VTOC remains intact; no further action is required.

  User Action: Check parameter-list against job control stream. Check DVC card for invalid logical unit number. The proper volume serial number was not specified for the Disc Preparation routine.

5.5.3. Rename Routine Error Codes

- Error Code 1. Volume Serial Number Error

  Causes: – No disc PUB has been found for the specified VSN.

  – The volume serial number in the parameter list does not match the one in the Standard Volume Label.

  – **A blank VSN is illegal.**

  Result: The VTOC is intact; no further action is required.

  User Action: Check DVC card for invalid logical unit number. The proper volume serial number was not specified for the Disc Preparation routine. Check calling sequence and job control stream for incompatibility.

- Error Code 2. File Identifier Not Found

  Causes: Specified 44-byte file identifier is not in the VTOC.

  Result: The VTOC is intact; no further action is required.

  User Action: Check calling sequence against VTOC printout.

5.5.4. Obtain Routine Error Codes

- Error Code 1. Volume Serial Number Error

  Same as described for error code 1 under Rename Routine Error Codes.

- Error Code 2. File Identifier Not Found or Illegal ID

  Causes: – By key, specified 44-byte file identifier is not in the VTOC.

  – By ID, specified 5-byte ID is not within the VTOC limits.

  Result: The VTOC is intact; no further action is required.

  User Action: Check calling sequence against VTOC printout.

APPENDIX A. JOB CONTROL/ LOGICAL IOCS INTERFACE

A.1. GENERAL

In this appendix the various interface label handling procedures between Job Control and logical IOCS for both tape and disc files are outlined.

A.2. LABEL PROCESSING FOR TAPE FILES

Label processing for tape files consists primarily of standard label checks for tape read (forward and backward) and write operations.

A.2.1. Expiration Date Field

The expiration date or retention period field in an LBL Job Control statement period controls the expiration date, which indicates the date when the file can be scratched. If it is specified as a five-digit EBCDIC operand, it is interpreted as an expiration date in the yyddd format, where yy is the year (00 through 99) and ddd is the day (001 through 366). If it is specified as a five-character EBCDIC operand Rnnnn, it is interpreted as a retention period, where nnnn is the number of days the file is to be retained beyond the creation date. In this latter form, the number of days is added to the creation date to produce an expiration date. (For purposes of this addition, a standard year of 365 days is used.) The expiration date is never checked on an input file.

A.2.2. General Relationships and Standard Tape Label Checks

The file serial number (FSN) in an HDR1 label is specified as being the same as the volume serial number (VSN) in the VOL1 label of the *first volume of the file*. Thus, all volumes of a multivolume file have an identity which is common to all volumes in the file, but unique to a particular run of that file. As such the FSN, which may be used as a "short cut" identification of a file, reduces the number of changes in the Job Control stream necessary to identify a file. It could also be used as an additional validity check when other controls such as the file identification field, volume sequence number, or date fields are employed.

Whenever the VSN in the VOL card or the FSN field of an LBL card is other than blank, Data Management assumes that the standard labels include both VOL1 and HDR1 labels and that the VSN/FSN relationship exists.

Whenever the VSN in the VOL card and the FSN in the LBL card are both blank, Data Management assumes that the standard header labels include an HDR1 label, but may or may not include a VOL1 label, and that if VOL1 labels are present, the VSN/FSN relationship may or may not exist.

If, in processing a file, a VOL card which contains a VSN is presented, Data Management assumes that the LBL card includes the correct FSN. Mounting instructions are by VSN. The VSN/FSN is checked on input and created on output. If a VOL card which contains SCRTCH is presented as the VSN, or an LBL card which contains VCHECK is presented as the FSN, mounting instructions are by filename and the VSN/FSN is checked on input and created on output.

If in processing a file, both the VSN in the VOL card and the FSN in the LBL card are blank, mounting instructions are by filename and the FSN/VSN relationship is not checked. VOL1 labels, if present, are bypassed. Proper use of the VSN/FSN relationship, together with the retention period feature of the LBL card, can reduce the number of job control stream changes from run to run and also provide satisfactory checks on input and identification of output.

The following tables outline the standard tape label checks performed for various combinations of LBL and VOL entries in the job control stream. A separate table is provided for each of the following operations:

- Tape Output
- Tape Input (read forward)
- Tape Input (read backward)

A.2.3. Tape Output Checks

JOB CONTROL STATEMENT LBL	VOL	TAPE LABEL CONVENTIONS	MVI*	UPDATE REQUIRED	LOGICAL IOCS OUTPUT CHECKS	VOLUME MOUNTING	OPERATOR CONTROL
None	VSN blank	HDR1 required. VOL1 not required but may be present.	Filename	No	VOL labels are retained if present. VSN/FSN relationship is not created.	By filename	If job control stream errors exist, go to ERROR or CANCEL routine.
FSN VCHECK	VSN=blank				Invalid combination		
FSN=blank	VSN=SCRTCH	VOL1 required HDR1 required	Filename	No	FSN/VSN relationship is created on all output reels. FSN is typed out when the first reel is opened.	By SCRTCH	
FSN VCHECK	VSN=SCRTCH	VOL1 required HDR1 required	Filename	No	FSN/VSN relationship is created on all output reels. FSN is typed out when the first reel is opened.	By SCRTCH	If incorrect tape mounted; mount another tape and retry with R, or reject with U.
FSN=fsn	VSN=FSN	VOL1 required HDR1 required	File serial number (FSN)	Yes	VSN of VOL card must equal FSN of LBL card. VSN of first VOL of output must equal VSN of VOL card. FSN/VSN relationship is created on all output reels. FSN of the first volume is not typed out.		
File ID	If a specific VSN is entered, an expiration date check is fatal (go to ERROR or CANCEL) since mounting another tape would cause a volume label check. If FSN is SCRTCH or blank, an expiration date check provides opportunity to try another tape (see operator control).			No	If blank, the seven-character label on the DTF for this file is placed in file ID. If not blank, the ID is (1) left-justified and (2) blank-filled in the ID field in HDR1.		Operator may not override an expiration date. A tape without proper VOL1 and/or HDR1 labels is considered unexpired. Operator may: (1) Mount another tape and retry with R. (2) Reject with U.
Expiration date: Rdddd or yyddd				No for Rdddd YES for yyddd	If expiration date is blank, current date is used unless SET DATE has been used in the job control stream for this run, in which case the date for this run is used. If expiration date is Rdddd (where dddd is 1-9999), it is accepted as a retention cycle, added to creation date, and placed in expiration date in the form yyddd. If not blank and not preceded by R, it is placed in expiration date (form yyddd).		
Creation date: yyddd				Yes	If blank, current date is used unless SET DATE has been entered in the job control stream for this run in which case the date for the run is used. If not blank, it is placed in creation date in HDR1. In any case, the expiration date of the file being overwritten must be equal to or less than the creation date as determined above.		
Version number				Yes	If blank, 01 is assumed to be the version number. If not blank, version number is taken from the LBL card.		
Generation number				Yes	If blank, 0001 is assumed to be the generation number. If not blank, generation number is taken from the LBL card.		
VOL sequence number				Yes	If blank, 0001 is assumed for the first reel. If not blank, VOL sequence number is initialized at the value in the LBL card. On subsequent volumes, VOL sequence number is incrementec by 1.		

*MVI – Minimum Visual Identification – external label used for reel identification must contain at least the information in this column.

A.2.4. Tape Input Checks (Read Forward)

JOB CONTROL STATEMENT LBL	JOB CONTROL STATEMENT VOL	TAPE LABEL CONVENTIONS	MVI*	UPDATE REQUIRED	LOGICAL IOCS INPUT CHECKS	VOLUME MOUNTING	OPERATOR CONTROL
None	VSN blank	HDR1 required. VOL1 not required but may be present.	Filename	No	VOL labels are bypassed if present. HDR1 label is typed out for operator acceptance. VSN/FSN relationship is not checked.	By filename	(1) HDR1 accepted by I. (2) Mount another tape and retry with R. (3) Reject with U.
FSN VCHECK	VSN=blank	VOL1 required HDR1 required	Filename	No	HDR1 label is not typed out. FSN in HDR1 of the first reel is typed out. VSN/FSN relationship is checked.		If VSN/FSN is not correct: (1) Mount another tape and retry with R. (2) Reject with U.
FSN=blank	VSN=SCRTCH				Invalid combination		
FSN VCHECK	VSN=SCRTCH	VOL1 required HDR1 required	Filename	No	HDR1 label is not typed out. FSN in HDR1 of the first reel is typed out. VSN/FSN relationship is checked.	By SCRTCH	If VSN/FSN is not correct: (1) Mount another tape and retry with R. (2) Reject with U.
FSN=fsn	VSN=FSN	VOL1 required HDR1 required	File serial number (FSN)	Yes	VSN of VOL card must equal FSN of LBL card. VSN of first reel must equal VSN of VOL card. VSN/FSN relationship is checked. HDR1 not typed out. FSN not typed out.	By VSN	Job control stream errors go to ERROR or CANCEL. If incorrect tapes: (1) Mount another tape and retry with R. (2) Reject with U.
File ID	If a specific VSN is entered, an invalid HDR1 label is fatal (go to ERROR or CANCEL) since mounting another reel would cause a volume label check. If VSN is SCRTCH or blank, an HDR1 check provides opportunity to try another tape (see operator control).			No	If blank, file ID is not checked. If not blank, must be equal to file ID in HDR1. Does not inhibit type-out of HDR1.		(1) Mount another tape and retry with R. (2) If unable to mount a correct tape, reject with U to go to CANCEL or ERROR.
Expiration date				No	Ignored on input.		
Creation date: yyddd				Yes	If blank, creation date is not checked. If not blank, must be equal to creation date in HDR1. Inhibits type-out of HDR1.		
Version number				Yes	If blank, version number is not checked. If not blank, must be equal to version number in HDR1. Inhibits type out of HDR1.		
Generation number				Yes	If blank, generation number is not checked. If not blank, must be equal to generation number in HDR1. Inhibits typeout of HDR1.		
VOL sequence number				Yes	VOL sequence number is checked starting with number on LBL card instead of 1. Inhibits typeout of HDR1.		

*MVI – Minimum Visual Identification – external label used for reel identification must contain at least the information in this column.

A.2.5. Tape Input Checks (Read Backward)

JOB CONTROL STATEMENT LBL	JOB CONTROL STATEMENT VOL	TAPE LABEL CONVENTIONS	MVI*	UPDATE REQUIRED	LOGICAL IOCS INPUT CHECKS	VOLUME MOUNTING	OPERATOR CONTROL
None	VSN=blank	EOF1 required	Filename	No	EOF1 label is typed out for operator acceptance.		(1) EOF1 accepted by I. (2) Reject with U.
FSN=VCHECK	VSN=blank	EOF1 required	Filename	No	EOF1 label is typed out. FSN in EOF1 label is typed out.		If FSN of EOF1 is not acceptable: (1) Reject with U. (2) Accept with I.
FSN=blank	VSN=SCRTCH				Invalid combination		
FSN=VCHECK	VSN=SCRTCH	EOF1 required	Filename	No	EOF1 label is typed out. FSN in EOF1 is typed out.		If FSN of EOF1 is not acceptable: (1) Reject with U. (2) Accept with I.
FSN=fsn	VSN=FSN	EOF1 required	File serial number (FSN)	Yes	VSN of VOL card must equal FSN of LBL card. FSN of EOF1 label must equal VSN of VOL card. EOF1 not typed out. FSN not typed out.		If job control stream error or incorrect tape, go to ERROR or CANCEL.
File ID	If a specific VSN is entered, an invalid EOF1 label is fatal (go to ERROR or CANCEL).			No	If blank, file ID is not checked. If not blank, must be equal to file ID in EOF1. Does not inhibit type-out of EOF1.		If unable to ignore error, reject with U to go to ERROR or CANCEL.
Expiration date Creation date: yyddd	If VSN is SCRTCH or blank, an EOF1 check provides the opportunity to ignore the error (see operator control).			No Yes	Ignored on input If blank, creation date is not checked. If not blank, must be equal to creation date in EOF1. Inhibits typeout of EOF1.		
Version number				Yes	If blank, version number is not checked. If not blank, must be equal to version number in EOF1. Inhibits type-out of EOF1.		
Generation number				Yes	If blank, generation number is not checked. If not blank, must be equal to generation number in EOF1. Inhibits type-out of EOF1.		
VOL sequence number				Yes	If blank, 0001 is assumed. If not blank, must be equal sequence in EOF1. Inhibits type-out of EOF1.		

*MVI – Minimum Visual Identification – external label used for reel identification must contain at least the information in this column.

A.3. LABEL PROCESSING FOR DISC FILES

This part of the appendix relates the parameters of the VOL and LBL Job Control statements to the processing of the disc format 1 label on each volume of a sequential, direct access, or indexed sequential file. In the following descriptions the words "new" and "old" (in parenthesis) refer to indexed sequential files, where a new file is one which is being created (loaded) while an old file is one which is undergoing subsequent processing. For full descriptions of the Job Control statements, refer to *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version).

A.3.1. VOL Statement

The format for the VOL statement is:

♭ OPERATION ♭	OPERAND
// VOL	{ C / Mcc / CMcc / volno-1 / SCRTCH } [, { volno-1or2 / SCRTCH }] ...

Reference should be made to the *UNIVAC OS/4 Job Control Programmer Reference, UP-7793* (current version) for a discussion of all of the options appearing in the operand field. A blank or missing VOL card is not a valid option for disc Data Management files. A discussion of the parameters from the VOL card which are pertinent to disc Data Management files follows. (The C, Mcc, and CMcc parameters refer only to tape volumes; "volno-1" refers to volume serial number 1.)

When an initial volume serial number (VSN) is supplied on a VOL card, it is checked against the standard volume label (VOL1) by job control to ensure that the correct volume is mounted. Informational and/or error messages will appear as job control type-outs, as appropriate. No further check is made by either DAM or ISAM file OPEN routines, since all of these volumes must be online initially. Subsequent SAM volumes *are* checked by comparing the succeeding VSNs which have been stored in the VSN List Block with the VOL1 labels on the succeeding volumes. Errors result in a DM49 message; see Table E–1.

The designation SCRTCH is unusual as a specification for disc Data Management files since most users will want to plan on which disc to store their files.

The SCRTCH parameter receives different treatments, depending on the type of access method chosen.

- For DAM and ISAM files, the action is controlled by the Job Control program. Job control searches for a mounted volume which has the designation SCRTCH stored in the physical unit block (PUB). If a SCRTCH volume is found, job control assigns this device to the user. If no disc PUB containing SCRTCH is found, the first unassigned device is chosen. After the operator mounts a volume, job control delivers the VSN for checking to the operator. When the operator indicates satisfaction with the displayed assignment, the VSN is stored in the PUB.

- For SAM files, the job control action for the first volume of a file is the same as for DAM and ISAM files. However, logical IOCS performs volume checking on this and all subsequent volumes. Whatever is stored in the VSN list block from the VOL card is compared to the VOL1 label VSN of the volume mounted. Hence, if SCRTCH is given, a match occurs only if the VOL1 label contains SCRTCH. Appropriate error action follows any mismatch (see Appendix E.5).

A.3.2. LBL Statement

The format of the LBL statement is:

LABEL	♭ OPERATION ♭	OPERAND
	// LBL	{ file-identifier / 'file-identifier' } [, { file-serial-number / V-CHECK }] [,volume-sequence-number] [,expiration-date] [,creation-date]

The file identifier is the only required parameter in the LBL statement for a disc file. Any other parameters are optional, as explained in the following paragraphs.

POSITIONAL PARAMETER 1

file-identifier – may be 44 characters (maximum); uniquely identifies the file by name on each volume. If less than 44 characters are specified, they are left-justified within a 44-byte field, with spaces filled to the right.

For each volume of a new file, the file identifier is written by the Allocate function into the key area of the disc physical record containing the format 1 label. Before it is written, it is checked for uniqueness against the other file identifiers. This processing is done by the Disc Space Management routines when space is allocated to the file; see Section 5.

When an OPEN macro instruction is executed for both input (old) and output (new) files, the file identifier is used as a search argument to retrieve the format 1 label on each volume. If the search is unsuccessful on any volume of the file, the job step is aborted, or control is passed to the user's error-handling routine.

'file-identifier' – identifies a disc filename that contains embedded blanks.

POSITIONAL PARAMETER 2

file-serial-number or VCHECK – relates to the file serial number field of the format 1 label for each volume of the file. The file serial number can be used to uniquely identify a particular copy of a file. It should be the volume serial number of the first volume of the file, and may be specified either as a one- to six-character volume serial number, or as the character string VCHECK. If a volume serial number of less than six characters is specified, it is right-justified within a six-byte field, with EBCDIC 0's filled to the left.

For each volume of an output (new) file, one of the following applies:

(1) If a volume serial number was specified, the volume serial number of the first volume of the file is inserted into the file serial number field of the format 1 label. In addition, the volume serial number from the LBL statement is checked against that of the first volume, and, if there is a discrepancy, a warning message is printed at the console while file processing continues. (The check and warning messages, if any, are executed only once for the file, not for each volume.)

(2) If the VCHECK parameter was specified, the volume serial number of the first volume is inserted into the file serial number field of the format 1 label. This option thereby allows the file serial number to be used without requiring reference to a specific volume serial number.

(3) If the file serial number was not specified, its field in the format 1 label will be set to binary 0's.

For each volume of an input (old) file, one of the following applies:

(1) If a volume serial number was specified, it is checked against the file serial number field of the format 1 label. If a discrepancy is found, the job step is aborted or control is passed to the user's error-handling routine.

(2) If the VCHECK parameter was specified, the volume serial number of the first volume of the file is checked against the file serial number field in the format 1 label. If a discrepancy is found, the job step is aborted or control is passed to the user's error-handling routine.

(3) If a file serial number was not specified, the file serial number field will not be checked.

POSITIONAL PARAMETER 3

volume-sequence-number – may be one or two decimal digits; assures the correct sequence of the volumes in the file. If only one digit is specified, a leading 0 is assumed. The specified value is converted to binary and identifies the first volume of the file. It is then incremented by one for each succeeding volume of the file.

For each volume of an output (new) file, the appropriate value is inserted in the volume sequence number field of the format 1 label. If this parameter is not supplied, the volume sequence number field of each format 1 label will be set to all binary 0's.

For each volume of an input (old) file, the appropriate value is checked against the volume sequence number field of the format 1 label. If there is a discrepancy, the job step is aborted or control is passed to the user's error-handling routine. If a value is not specified, the volume sequence number field will not be checked.

POSITIONAL PARAMETER 4

expiration-date or retention-period – controls the expiration date; indicates the date when the file can be scratched. If it is specified as a five-digit EBCDIC operand, it is interpreted as an expiration date in the yyddd format, where yy is the year (00 through 99) and ddd is the day (001 through 366). If it is specified as a five-character EBCDIC operand Rnnnn, it is interpreted as a retention period, where nnnn is the number of days the file is to be retained beyond the creation date. In this latter form, the number of days is added to the creation date to produce an expiration date. (For purposes of this addition, a standard year of 365 days is used.)

For each volume of an output (new) file, the specified or computed expiration date is converted to a three-byte binary number (in the form ydd) and is placed into the expiration date field of the format 1 label. If this positional parameter is omitted, the expiration date field will be set to all binary 0's.

This positional parameter is not processed for an input (old) file.

POSITIONAL PARAMETER 5

creation-date – specifies a five-digit EBCDIC date, which represents the year and day (yyddd) on which the file is considered to have been created.

For each volume of an output (new) file:

(1) If a creation date is specified, it is converted to a three-digit binary number (in the form ydd) and is placed in the creation field of the format 1 label.

(2) If this positional parameter is not supplied, the operating system uses the disc Data Management date in the job preamble, which is specified either in a SET statement for the job, or in a SET command for the system. In either case, this date is converted and used as described in (1) above.

For each volume of an input (old) file:

(1) If a creation date is specified, it is converted to the binary form ydd and is checked against the creation date field in the format 1 label. If a discrepancy is found, the job step is aborted or control is passed to the user's error-handling routine.

(2) If a creation date is not supplied, the creation date field of the format 1 label is not checked.

The following tables summarize the standard label checks performed for disc files and the appropriate error action in each case.

A.3.3. Disc Label Checking

JOB CONTROL FIELD (STATEMENT)	CONTENTS	ACTION	ERROR ACTION*
Volume serial number (VOL)	Blank (or no VOL card)	(Invalid specification)	DM48 message. Enter ERROR= symbol if present; otherwise CANCEL.
	SCRTCH	See text, A.3.1.	
	nnnnnn	Compared to VSN in VOL1 label of volume.	For direct access and indexed sequential files: JC02 job control message. For sequential files: JC02 job control message for first volume. DM4N VOL1 ERR message for subsequent volumes. In either case, operator can retry or CANCEL.
File identifier (LBL)	Blank (or no LBL card)	(Invalid specification)	DM4F message. Enter ERROR =symbol if present; otherwise CANCEL.
	Key (up to 44 characters)	VTOC is searched for a format 1 label with given key.	DM4A message. Enter ERROR= symbol if present; otherwise CANCEL.
File serial number (LBL)	Blank	None	None
	VCHECK	FSN of format 1 label is compared to VSN of first volume.	DM4N message. Enter ERROR= symbol if present; otherwise CANCEL.
	nnnnnn	Compared to FSN of format 1 label.	DM4N message. Enter ERROR= symbol if present; otherwise CANCEL.
Volume sequence number (LBL)	Blank	None	None
	nnnn	Volume sequence number of format 1 label compared to sum of nnnn and relative position of volume.	DM4N message. Enter ERROR= symbol if present; otherwise CANCEL.
Expiration date (LBL)		Field Ignored	
Creation date (LBL)	Blank	None	None
	yyddd	Compared to creation date in format 1 label.	DM4N message. Enter ERROR= symbol if present; otherwise CANCEL.

**Refer to Appendix E for full explanation of these error codes.*

A.3.4. Disc Label Creation

JOB CONTROL FIELD (STATEMENT)	CONTENTS	ACTION	ERROR ACTION*
Volume serial number (VOL)	Blank (or no VOL card)	(Invalid specification)	DM48 message. Enter ERROR= symbol if present; otherwise CANCEL.
	SCRTCH	See text, A.3.1.	
	nnnnnn	Compared to VSN in VOL1 label of volume.	For direct access and indexed sequential files: JC02 job control message. For sequential files: JC02 job control message for first volume. DM4N VOL1 ERR message for subsequent volumes. In either case, operator can retry or CANCEL.
File identifier (LBL)	Blank (or no LBL card)	(Invalid specification)	DM4F message. Enter ERROR= symbol if present; otherwise CANCEL.
	Key (up to 44 characters)	VTOC is searched for a format 1 label with given key.	DM4A message. Enter ERROR= symbol if present; otherwise CANCEL.
File serial number (LBL)	Blank	None	None
	VCHECK	VSN of initial volume is placed in FSN field of format 1 label.	None
	nnnnnn	1. VSN of initial volume is placed in FSN field of format 1 label. 2. nnnnnn is compared to VSN of initial volume.	DM4N message. Processing continues.

**Refer to Appendix E for full explanation of these error codes.*

JOB CONTROL FIELD (STATEMENT)	CONTENTS	ACTION	ERROR ACTION*
Volume sequence number (LBL)	Blank	None	None
	nnnn	Volume sequence number of format 1 label is set to sum of nnnn and relative position of volume.	None
Expiration date (LBL)	Blank	None	None
	Rnnnn	Format 1 field set to sum of nnnn and creation date; correcting for year changes.	None
	yyddd	Format 1 field set to specified date (in binary).	None
Creation date (LBL)	Blank	Format 1 field set to date in preamble.	None
	yyddd	Format 1 field set to specified date (in binary).	None

* – Refer to Appendix E for full explanation of these error codes.

APPENDIX B. COMPATIBLE KEYWORD PARAMETERS

B.1. GENERAL

To help maximize the compatibility of UNIVAC OS/4 Operating System (OS/4) source programs with those of comparable systems, each of the DTF declarative macro instructions accepts as equivalent the comparable keyword parameters listed in the following tables. All other keyword parameters are identical among systems.

B.2. DTFMT PARAMETERS

UNIVAC OS/4 DTFMT KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	Block length
EOFA	EOFADDR	User's end-of-file address
FLBL	FILABL	Indicates type of labels in the file
IOA1	IOAREA1	Name of user's first I/O area
IOA2	IOAREA2	Name of user's second I/O area
IORG	IOREG	I/O pointer register
TYPE	TYPEFLE	Specifies whether file is to be used for input, output, or both
CKPT	CKPTREC	Indicates presence of checkpoint record
ERRO	ERROPT	Specifies user's error routine resulting from parity errors
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Record length
TPMK	TPMARK	Indicates tapemark is written as first record following labels
LBAD	LABADDR	Name of user's label processing routine
VBLD	VARBLD	Specifies a user register in which logical IOCS places the variable length record residual space
WORK	WORKA	Specifies a user workarea in which records are processed

Table B–1. Compatible DTFMT Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

DEVADDR	CRDT	NOTEPNT
DEVA	XPDT	REWIND
HDRINFO	GENO	SEPASMB
FLID	VOLN	WLRERR
	MODNAME	

B.3. DTFSD PARAMETERS

UNIVAC OS/4 DTFSD KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	Block length
CNTRL	CONTROL	Control entry
EOFA	EOFADDR	User's end-of-file address
ERRO	ERROPT	Unique file error address
IOA1	IOAREA1	Name of user's first input/output area
IOA2	IOAREA2	Name of user's second input/output area
IORG	IOREG	I/O pointer register
LBAD	LABADDR	User label processing address
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Logical record length
TYPE	TYPEFLE	Type of retrieval functions
UPDT	UPDATE	Input file update
VBLD	VARBLD	Variable length record residual space register
WORK	WORKA	Work area to be used for processing data

Table B-2. Compatible DTFSD Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

DELETFL	MODNAME	SEPASMB
DEVADDR	NOTEPNT	WLRERR

B.4. DTFCD PARAMETERS

UNIVAC OS/4 DTFCD KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	Block length
CNTRL	{ CONTROL / CNTL }	Control entry
EOFA	EOFADDR	User's end-of-file address
IOA1	IOAREA1	Name of user's first input/output area
IOA2	IOAREA2	Name of user's alternate input/output area
IORG	IOREG	I/O pointer register
OBSZ	OUBLKSZ	Output blocksize for combined files
PUNR	CRDERR	Hole count error recovery
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Record size register for undefined records
TYPE	{ TYPEFLE / TYPF }	Type of file
WORK	WORKA	Specifies a user workarea for processing records

Table B–3. Compatible DTFCD Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

DEVADDR SEPASMB MODNAME
DEVICE

B.5. DTFPR PARAMETERS

UNIVAC OS/4 DTFPR KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	Block length
CNTRL	{ CONTROL / CNTL }	Control entry
IOA1	IOAREA1	Name of user's first input/output area
IOA2	IOAREA2	Name of user's alternate input/output area
IORG	IOREG	I/O pointer register
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Record size register for undefined records
WORK	WORKA	Specifies a user work area for processing records

Table B–4. Compatible DTFPR Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

PRINTOV DEVADDR DEVICE
SEPASM MODNAME STLIST
UCS

B.6. DTFDA PARAMETERS

UNIVAC OS/4 DTFDA KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	I/O area size
CNTRL	CONTROL	Offline seek
IOA1	IOAREA1	Name of user's I/O area
LBAD	LABADDR	Name of user's label processing routine
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Record length
TYPE	TYPEFLE	Type of file

Table B–5. Compatible DTFDA Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

DEVADDR
MODNAME
SEPASMB

B.7. DTFIS PARAMETERS

UNIVAC OS/4 DTFIS KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Record size
IORG	IOREG	I/O pointer register
{WORK / WORKS}	WORKS	Work areas to be used for sequential processing
TYPE	TYPEFLE	Type of retrieval/update functions

Table B–6. Compatible DTFIS Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

DSKXTNT
HINDEX
INDSKIP
MODNAME
SEPASMB

B.8. DTFOR PARAMETERS

UNIVAC OS/4 DTFOR KEYWORD PARAMETERS	COMPARABLE KEYWORD PARAMETERS	USAGE
BKSZ	BLKSIZE	Block length
CNTL	CONTROL	Indicates the DJ$CTL module is needed
EOFA	EOFADDR	Name of user's end-of-file routine
IOA1	IOAREA1	Name of user's first I/O area
IOA2	IOAREA2	Name of user's second I/O area
IORG	IOREG	Indicates user's I/O register
RCFM	RECFORM	Record format
RCSZ	RECSIZE	Indicates user's RCSZ register
ROWS	MRST	Indicates which rows to be sensed
WORK	WORKA	Specifies a user work area

Table B–7. Compatible DTFOR Keyword Parameters

The following keyword parameters are found in comparable systems, but are ignored in UNIVAC OS/4 source programs:

BLKFAC	DEVA,DEVADDR	MODNAME
CHNL	DEVICE	SEPASMB
COREXIT	HEADER	

APPENDIX C. DATA MANAGEMENT I/O MODULES

C.1. GENERAL

The automatic-include feature of the Linkage Editor calls in the required I/O object modules from the Systems Reserve Library, as dictated by the EXTRN directives generated by each DTF call.

The various I/O module names associated with each DTF are given in the following sections.

C.2. DTFMT I/O MODULES

MODULE NAME	COMMENTS
DS$PTA	Output file, unblocked records, no work area specified
DS$GTA	Input file, unblocked records, no work area specified
DS$PTB	Output file, unblocked records, work area specified
DS$GTB	Input file, unblocked records, work area specified
DS$PTC	Output file, blocked records, no work area specified
DS$GTC	Input file, blocked records, no work area specified
DS$PTD	Output file, blocked records, work area specified
DS$GTD	Input file, blocked records, work area specified
DS$PTE	May be used with any of the above output file combinations
DS$GTE	May be used with any of the above input file combinations
DT$GTG	Required when processing input files backward
DS$ISS	Required with any of the above
DT$WKO	Required with any of the above output file combinations
DT$WKI	Required with any of the above input file combinations, except read backward
DT$WKB	Required when processing input files backward
DS$MVR	Required with any file using a work area
DS$TNC	Required with blocked output files
DS$RSE	Required with blocked input files
DT$CTL	Required with any file using the CNTRL macro instruction

C.3. DTFSD I/O MODULES

MODULE NAME	COMMENTS
DS$PTA	Output file, unblocked records, no work area specified
DS$GTA	Input file, unblocked records, no work area specified
DS$PTB	Output file, unblocked records, work area specified
DS$GTB	Input file, unblocked records, work area specified
DS$PTC	Output file, blocked records, no work area specified
DS$GTC	Input file, blocked records, no work area specified
DS$PTD	Output file, blocked records, work area specified
DS$GTD	Input file, blocked records, work area specified
DS$PTE	May be used with any of the above output file combinations
DS$GTE	May be used with any of the above input file combinations
DS$ISS	Required with any of the above
DS$WCO	Required with any of the above output file combinations
DS$WCI	Required with any of the above input file combinations and including the updating function
DS$MVR	Required with any file using a work area
DS$TNC	Required with blocked output files
DS$RSE	Required with blocked input files
DS$CNT	Required with any file using the CNTRL macro instruction, including INPUT with UPDT specified
DS$GTF	Input file, with UPDT specified
DS$PTF	Input file, with UPDT specified
DX$ISS	Input file, with UPDT specified
DS$RLU	Input file, blocked records, with UPDT specified
DS$ISU	Input file, with UPDT specified

C.4. DTFCD I/O MODULES

MODULE NAME	COMMENTS
DQ$GET	Combined file, input module
DQ$PUT	Combined file, output module
DR$GET	Input file (card reader only)
DW$10E	Output file, one I/O area but no work area specified
DW$11E	Output file, one I/O area with a work area specified
DW$20E	Output file, two I/O areas but no work area specified
DW$21E	Output file, two I/O areas with a work area specified
DW$62E	May be used with any of the above output file combinations
DW$CHK	Required with any of the above
DW$CNT	Required with any file using the CNTRL macro instruction
DW$RTY	Required with any file when PUNR is specified

C.5. DTFPR I/O MODULES

MODULE NAME	COMMENTS
DP$10E	Output file, one I/O area but no work area specified
DP$11E	Output file, one I/O area with a work area specified
DP$20E	Output file, two I/O areas but no work area specified
DP$21E	Output file, two I/O areas with a work area specified
DP$62E	May be used with any of the above output file combinations
DP$CHK	Required with any of the above
DP$CNT	Required with any file using the CNTRL macro instructions
DP$PRT	Required with any file when PRTOV is specified

C.6. DTFDA I/O MODULES

MODULE NAME	COMMENTS
DD$RD	Required with READ macro instructions
DD$WR	Required with data WRITE macro instructions (KEY or ID specified)
DD$FW	Required with format WRITE macro instructions (AFTER or RZERO specified)
DD$SK	Required with any file using the CNTRL macro instruction
DD$CT	Required with any file
DD$ID	Required with any file when RELATIVE is not specified
DD$RL	Required with any file when RELATIVE is specified
DD$AC	Required with any file when AFTER is not specified
DD$BC	Required with any file when AFTER is specified
DD$WT	Required with any file
DD$AW	Required with any file when IDLOC is specified (absolute addressing)
DD$RW	Required with any file when IDLOC is specified (relative addressing)

C.7. DTFIS I/O MODULES

MODULE NAME	COMMENTS
DU$ADD	Required with IOROUT = ADD or IOROUT = ADDRTR specified
DU$ABLK	Required with IOROUT = ADD or IOROUT = ADDRTR and RCFM = FIXBLK specified
DU$AUNB	Required with IOROUT = ADD or IOROUT = ADDRTR and RCFM = FIXUNB specified
DU$AMIO	Required with IOROUT = ADD or IOROUT = ADDRTR and IOSIZE specified
DU$LOAD	Required with IOROUT = LOAD or RELOAD specified
DU$RRAN	Required with IOROUT = RETRVE and TYPE = RANDOM specified
DU$RWRT	Required with IOROUT = RETRVE and TYPE = RANDOM specified
DU$RESL	Required with IOROUT = RETRVE and TYPE = SEQNTL specified
DU$RGET	Required with IOROUT = RETRVE and TYPE = SEQNTL specified
DU$RPUT	Required with IOROUT = RETRVE and TYPE = SEQNTL specified
DU$RSEQ	Required with IOROUT = RETRVE and TYPE = SEQNTL specified
DU$WTF	Required with (IOROUT = ADD or IOROUT = ADDRTR) or (IOROUT = RETRVE and TYPE = RANDOM) specified
DU$XRCI	Required with (IOROUT = ADD or IOROUT = ADDRTR) or (IOROUT = RETRVE and TYPE = RANDOM) and (INDAREA and INDSIZE) specified

C.8. DTFOR I/O MODULES

MODULE NAME	COMMENTS
DJ$GET	Input file
DJ$CTL	Required with any file using the CNTRL macro instruction
DJ$WCK	Required with any of the above

C.9. DTFPT I/O MODULES

MODULE NAME	COMMENTS
DV$GET	Input file (reader)
DV$ISS	Input file, with SCAN specified and 1 I/O area
DV$SCN	Input file, with FTRANS, LTRANS, and SCAN specified
DV$TRAN	Required if TRANS specified for output files, or FTRANS and LTRANS specified for input files
DV$WCI	Required for input files
DS$MVR	Required with any file using a work area
DV$PUT	Output file (punch)
DV$SHFT	Output file, with TRANS, FSCAN, or LSCAN specified
DV$WCO	Required for output files

APPENDIX D. DATA MANAGEMENT TRANSIENT ROUTINES

D.1. GENERAL

The Data Management transient routines are stored on the systems device as relocatable modules, which are located and brought into the transient area by the supervisor when an OPEN, CLOSE, or FEOV macro instruction is encountered in the problem program. The various transient routine names associated with each DTF follow:

D.2. DTFMT TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YF6600 – Input/output common
YF660A – Input/output common
YF6601 – Input first overlay
YF6602 – Input second overlay
YF6603 – Input third overlay
YF6604 – Input fourth overlay
YF6605 – Output first overlay
YF6606 – Output second overlay
YF6607 – Output third overlay
YF6608 – Input fifth overlay
YF6609 – Input sixth overlay

The CLOSE transient routine names are:

YF6800 – Input/output common
YF6801 – Input first overlay
YF6802 – Input second overlay
YF6803 – Input third overlay
YF6804 – Input fourth overlay
YF6805 – Output first overlay
YF680A – Output second overlay
YF6806 – Output third overlay

The FEOV common transient routine name is YF6C00.

D.3. DTFSD TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YS6600 – Input/output common (and complete the DTF)
YS6601 – Input/output (and complete the DTF)
YS6602 – Input/output label checking
YS6603 – Format 1 label processing
YS6604 – Format 1 and 3 label processing
YS6605 – User header label processing
YS6606 – Cleanup and input file initial reads
YS6607 – Special format 1 and format 3 label reads

The CLOSE transient routine names are:

YS6800 – Input/output common
YS6801 – First overlay – user trailer labels
YS6802 – Second overlay – multivolume files

The FEOV common transient routine name is YS6C00.

D.4. DTFCD TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YR6600 (READER)
YW6600 (PUNCH)
YQ6600 (READ/PUNCH)

The CLOSE transient routine names are:

YR6800 (READER)
YW6800 (PUNCH)
YQ6800 (READ/PUNCH)

D.5. DTFPR TRANSIENT ROUTINE NAMES

The OPEN transient routine name is YP6600.

The CLOSE transient routine name is YP6800.

D.6. DTFDA TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YG6600 – first overlay
YG6601 – second overlay
YG6602 – third overlay
YG6603 – fourth overlay
YG6604 – fifth overlay

The CLOSE transient routine name is YG6800.

D.7. DTFIS TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YY6600 – first overlay
YY6601 – second overlay
YY6602 – third overlay
YY6603 – fourth overlay
YY6604 – fifth overlay

The CLOSE transient routine names are:

YY6800 – first overlay
YY6801 – second overlay

For file loading routines, the transient routine names are:

YT9C00 – SETFL (set file load)
YT9C01 – second overlay
YTA000 – ENDFL (end file load)

For sequential processing, the SETL transient routine name is YT9E00.

D.8. DTFOR TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YJ6600

The CLOSE transient routine names are:

YJ6800

D.9. DTFPT TRANSIENT ROUTINE NAMES

The OPEN transient routine names are:

YV6600

The CLOSE transient routine names are:

YV6800

APPENDIX E. DATA MANAGEMENT MESSAGES

E.1. CONSOLE MESSAGES

Data Management operator communication is accomplished by means of a series of console messages. These messages are used to describe an error condition or an informational function. The following standard prefix is typed out for all Data Management messages. This prefix can appear by itself or in combination with an informational type message. When the error code appears alone, it denotes the reason for the message. When the error code occurs as a message prefix for an informational type message, the error code position (e) is usually blank.

DM4e uuu filename

where: e – represents an alphanumeric character code that specifies the error that has occurred. These error codes are defined in the Data Management portion of the Standard Equate macro instruction, STDEQU, and are listed in Table E–1.

uuu – is the device identification code.

filename – is the label assigned to the DTF macro instruction.

Refer to *UNIVAC 9400 System Operations Handbook Operator Reference, UP-7871* (current version), for explanation and action of all console messages.

Messages are possible from all OPEN and CLOSE transient modules. There is no reply to the message when the prefix appears alone. If the user supplied an ERROR= symbol address in the DTF macro instruction, control is passed to that point; otherwise the job step is cancelled. However, note exceptions to this action as indicated in Notes 2 and 4 in Table E–1.

ERROR CODE (DM4e)	STANDARD EQUATE VALUE	REASON FOR ERROR	APPLICATION FILE TYPE		
			OPEN	PROCESSING⑤	CLOSE (FEOV)
0	DI$E01	OPENing an already OPENed file	All		
1	DI$E02	CLOSEing an already CLOSEd file			MT,SD,CD, PR, IS, PT, CR
2	DI$E02A	Lockout systems error	DA		
3	DI$E03	Unreadable or missing File Control Block①	All		MT
4	DI$E05	Block size or KEYLOC specification error②	All		
5	DI$E06	DTF specification error	All		IS
6	DI$E07	Required Data Management module missing	All		
7	DI$E09	Extent table full	DA,SD, IS		
8	DI$E11	No PUB allocated, and for SAM file no OPTION=YES specified. Also for disc, missing VOL card or blank VSN field on VOL card③	All		
9	DI$E13	VOL1 or EOF1 label missing, unreadable, or incorrect	MT,SD, DA, IS		
A	DI$E15	Format 1 or HDR1 label not found or unreadable	MT,SD DA,IS		
B	DI$E16	Tape standard label (user or system) unreadable	MT		MT
	DI$E71	Format 1 label for the file indicates zero extents on current file	SD		
C	DI$E17	Format 2 or 3 label not found or unreadable	SD,DA, IS		IS
D	DI$E19	Format 4 label not found or unreadable	SA,DA, IS		IS
E	DI$E20	Error in reading user header or trailer label	MT,SD, DA		MT
F	DI$E21	Missing file ID in file control block	SD,DA, IS		
G	DI$E23	I/O error (hardware) when reading or writing data	MT,SD IS	MT,SD,CD, PR,DA,IS	MT,SD
H	DI$E25	Accessing a file which is not OPEN		DA,IS	IS,DA
I	DI$E27	VSN list block error	MI,SD		MT,SD
	DI$E65	Invalid extent specification for ISAM	IS		
J	DI$E29	Initial VSN≠FSN from LBL card (tape), or parameter 2 of LFD card not IS for ISAM	MT,IS		
K	DI$E31	Error in writing a system standard label block	MT,SD, DA,IS		MT,SD, IS
L	DI$E33	Error in writing a user label block	MT,SD, DA		MT,SD

Table E–1. Explanation of Error Codes (Part 1 of 2)

ERROR CODE (DM4e)	STANDARD EQUATE VALUE	REASON FOR ERROR	APPLICATION FILE TYPE		
			OPEN	PROCESSING (5)	CLOSE (FEOV)
M	DI$E35	Unexpired expiration date	MT		
N	DI$E37	Standard label field incorrect, as specified for checking by job control stream (4)	MT,SD, DA,IS		
O	DI$E39	I/O limits-check failure (ERRBYTE,IDLOC, IOA1, IOA2, IOAREAL, IOAREAR, IOAREAS)	All		
P	DI$E41	Invalid imperative macro instruction		SD, DA, IS, OR	IS
Q	DI$E43	WAITF macro instruction not issued following a READ or WRITE macro instruction		DA	DA
R	DI$E45	Missing or inaccessible transient overlay, or common code index error	All		
S	DI$E47	Chain area overflow	DA		
T	DI$E49	Time limit reached when processing user labels, user extents, or searching for tape marks on tape	MT,SD, DA		MT,SD
U	DI$E51	Keylength incorrectly specified (less than 3 or greater than 255), or missing when required	DA,IS		
V	DI$E53	Block length incorrect		MT, PT	
W	DI$E55 DI$E67	Checkpoint numbers on tape do not match Too many cylinder overflow tracks on disc	IS	MT	
X	DI$E57	Block number or data block count error	MT	MT	MT
Y	DI$E59	Invalid or inconsistent device (DVC) assignment	MT,SD, IS, OR, PT,CD, PR		
Z	DI$E61	Resident cylinder index cannot be supported	IS		
¢	DI$E63 DI$E69	Expanded I/O area cannot be supported Exhausted last extent of last volume	IS SD		
&	DI$E72	Printer mismatch		PR	
*	DI$E73	End of record character detected (INPUT) Low tape condition (OUTPUT)		PT	
+	DI$E74	Invalid character control (CTLCHR)		PR	
–	DI$E75	Unique unit error		CD	
#	DI$E77	Alternate track disallows add	IS		
%	DI$E78	Input and/or output translation tables not present	CD		

NOTES: (1) Check job stream for a missing or incorrect LFD card (filename is seven characters maximum).

(2) For card reader, punch, or printer only, block size parameter is adjusted to show maximum; and processing continues.

(3) Check job stream for missing DVC card or incorrect or missing VOL card.

(4) For disc output (new) files, when FSN or LBL card does not equal the initial VSN, this message is informational and program continues.

(5) In volume swap situations for tape (MT) and disc (SD) files, some errors listed as occurring during OPEN, CLOSE, or FEOV can also occur during GET or PUT processing.

Table E–1. Explanation of Error Codes (Part 2 of 2)

APPENDIX F. ERROR CONDITIONS FOR ISAM FILES

F.1. GENERAL

Error conditions encountered in the processing of indexed sequential files can be placed into three categories:

(1) Initialization errors detected during the execution of an OPEN macro instruction, which are usually unrecoverable;

(2) Recoverable errors detected other than during the execution of an OPEN instruction; and

(3) Unrecoverable errors detected other than during the execution of an OPEN instruction.

Errors detected during the execution of the OPEN macro instruction cause a message to be printed at the operator's console, which indicates in which file and logical unit the error occurred and the nature of the error. If the error is unrecoverable, the job step is aborted or control is passed to the user's error routine; if it is recoverable, the job step continues normally.

In the case of those errors detected other than during the execution of an OPEN macro instruction, the unrecoverable errors generally cause either an exit to the user's error routine or an abort (if no error routine was specified), while the recoverable errors are indicated in a field (in the DTFIS-generated file table) labeled filenameC (refer to 4.2.2), with the control returning to the normal place in the user's program.

F.2. ERROR FIELD (filenameC)

Within the table generated by the DTFIS declarative macro instruction for the file is a two-byte field which is used to convey error and status information to the user's program. This two-byte error field is addressed by the concatenated label filenameC. For example, for files labeled EMPLUST, OMST, and NC, this field in each file would be addressed as EMPLUSTC, OMSTC, and NCC, respectively. (See 4.2.2 for a further description of the use of this method of labeling.)

Unless otherwise noted, control is always returned to the next instruction following the macro instruction in which the error or condition was detected. In all cases, register 1 contains the address of the DTFIS-generated table, registers 2 through 13 remain as the user set them, and register 14 contains the address of the normal return address in the user program. Therefore, the error field filenameC should be checked after each macro instruction, except OPEN and CLOSE.

Byte 1 of the error field contains indicators for various error conditions. The type of error indicator is based on the type of file processing being performed. The error field as set during each type of file processing is described in the following paragraphs.

F.2.1. Unrecoverable Error Indicators

In byte 1 of the error field, filenameC, two bits, 0 and 1, are used in the same way for all types of processing. Bit 0 indicates an unrecoverable device error, and bit 1 is not used.

Bit 0 of filename C is set when byte 2 of the error field is other than zero. Except as noted, byte 2 (filenameC+1) reflects byte 1 of the transmission bytes in the CCB at the time of the error.

- Byte 1, Bit 0 – Unrecoverable Device Error

  If an unrecoverable device error has been detected during the execution of a macro instruction, an indicator in the DTFIS-generated table marking the file as open is turned off. Further access to the file results in a file-not-open error condition.

- Byte 1, Bit 1

  This bit is not used by the file processing functions of Data Management.

- Byte 2

  The entire byte is used to reflect the first transmission byte in the CCB as set by the I/O interrupt processing routines and error processing job. Additional settings of byte 2 are made if errors occur when records are being inserted in a file (see F.2.3).

F.2.2. Loading or Extending a File

When IOROUT=LOAD (or RELOAD) is specified, a new file is to be created or re-created, or an old file is to be extended as described in 4.2.3.2. The ENDFL macro instruction sets only bit 0 of byte 1 of the error field. In addition to bit 0, the SETFL macro instruction sets bits 2 and 3; the WRITE NEWKEY sets bits 2, 5, and 6. The following paragraphs describe the conditions for setting these bits. Figure F–1 summarizes the bit settings for errors which occur during the file loading or extending.

- Bit 2 – Prime Data Area Full

  This bit can be set for an old file by the SETFL macro instruction and for an old or new file by the WRITE NEWKEY macro instruction when the prime data area is full.

  When the SETFL macro instruction attempts to extend an old file, and the prime data is already full, an indicator in the DTFIS-generated table marking the file open is reset. Further access to the file results in a file-not-open condition.

  If a record loaded into the file by the WRITE NEWKEY macro instruction fills the prime data area, no further records may be loaded. When control returns to the user program, the four-byte filenameP field in the DTFIS-generated table contains a count of the number of logical records in the prime data area, and filenameH contains the disc address (in the form MBBCCHHR) of the logical record just written.

  If a file is to be accepted as loaded, the file loading process may be terminated normally by executing an ENDFL macro instruction. When executed successfully, ENDFL resets filenameC.

  If, however, the file is not accepted as loaded, the user program should be terminated if bit 2 of filenameC is on. The entire loading process beginning with the initial load must be repeated after this file is scratched, and the necessary amount of disc space must be allocated.

- Bit 3 – Index Area Too Small

  When the SETFL macro instruction computes the space required for the higher level cylinder and master indexes, based on the file key size and the number of prime data cylinders, and the extent (first extent on the first volume) is not large enough, this bit is set. This is an unrecoverable error, and the user program must be terminated so that additional index space can be allocated to the file.

  An indicator in the DTFIS-generated table marking the file as open is reset. If another macro instruction is executed for the file, control is transferred to the user's error routine. If an error routine was not specified, the job step is aborted.

- Bit 4 – Unused

- Bit 5 – Duplicate Key

  When the WRITE NEWKEY macro instruction attempts to load the current logical record from the area specified by WORKL and the key of the current logical record duplicates a key already in the file, this bit is set. The current record is not placed in the file, but normal processing of the file can continue with other logical records.

- Bit 6 – Key Out of Sequence

  When the WRITE NEWKEY macro instruction attempts to load the current logical record from the area specified by WORKL and the key of the current logical record is less than the key of the preceding logical record, this bit is set. The current record is not placed in the file, but normal processing of the file can continue with another logical record.

- Bit 7 – Unused

IOROUT	MACRO INSTRUCTION	BYTE 1							
		0	1	2	3	4	5	6	7
LOAD RELOAD	SETFL	X		X	X				
	WRITE NEWKEY	X		X			X	X	
	ENDFL	X							

Figure F–1. Error Field Bit Settings – File Loading and Extending

F.2.3. Inserting Records in a File

When IOROUT=ADD (or ADDRTR) is specified, the WRITE NEWKEY and WAITF macro instructions are used to insert records in a file. The WRITE NEWKEY macro instruction sets only bit 0 in byte 1 of the error field for an unrecoverable device error as described in F.2. Byte 2 of the error field reflects the first transmission byte in the CCB at the time of the error. Additional settings of byte 2 may be made by the WAITF macro instruction. Figure F–2 summarizes all bit settings. An explanation of the bit settings in the error field are given in the following paragraphs.

- Bit 0 – Unrecoverable Device Error

  If the WAITF macro instruction finds an unrecognizable track index entry type, byte 1 is set to X'80' and byte 2 is set to X'01'. If the WAITF macro instruction finds an extent table or last prime data ID anomaly, byte 1 is set to X'80' and byte 2 is set to X'02'. In either case an indicator in the DTFIS-generated table marking the file as open is turned off. Further attempts to access the file result in a file-not-open error condition.

- Bits 1 through 4 – Unused

- Bit 5 – Duplicate Key

  When the WAITF macro instruction attempts to insert the current logical record from the area specified by WORKL and the key of the current logical record duplicates a key already in the file, this bit is set. The current record is not placed in the file, but normal processing of the file can continue with another logical record.

- Bit 6 – Overflow Area Full

  When the WAITF macro instruction, issued following a WRITE NEWKEY macro instruction, attempts to insert the current logical record from the area specified by WORKL and no overflow area is available to hold the logical record displaced from the prime data tracks, this bit is set. The record is not inserted, but normal processing of the file can continue.

- Bit 7 – Unused

IOROUT	MACRO INSTRUCTION	BYTE 1 – filenameC								BYTE 2 – filenameC+1 (SET WHEN BYTE 1 = X'80')
		0	1	2	3	4	5	6	7	
ADD or ADDRTR	WRITE NEWKEY	X								transmission byte in CCB
	WAITF	X					X	X		X'01' or X'02'

Figure F–2. Error Field Bit Settings – Inserting Records

F.2.4. Random Processing of Records

When IOROUT=RETRVE (or ADDRTR) and TYPE=RANDOM (or RANSEQ) are specified so that records may be retrieved in random order for processing, the READ KEY macro instruction and WAITF macro instruction that must follow set the error field. The WRITE KEY macro instruction and the WAITF macro instruction that must follow set only bit 0 of byte 1 of the error field. Figure F–3 summarizes all bit settings.

- Bit 2 – Unused

- Bit 3 – No Record Found

  When a READ KEY macro instruction or the following WAITF macro instruction is executed and no record with a key equal to the key argument value is found in the file, this bit is set. A successful WAITF macro instruction does not preclude further processing of the file and may be followed by a valid macro instruction other than a WRITE KEY macro instruction. However, if bit 3 was set when the READ KEY macro instruction was executed, a WAITF macro instruction must be issued before the file can be processed further. Bit 3 remains set, but any valid macro instruction other than a WRITE KEY may now be issued.

- Bits 4 through 6 – Unused

- Bit 7 – Record Retrieval from Overflow

  If the logical record retrieved was taken from an overflow area, this bit is set during the execution of the WAITF macro instruction which follows a READ KEY macro instruction. Processing of the record and file proceeds normally.

IOROUT	TYPE	MACRO INSTRUCTION	BYTE 1 – filenameC							
			0	1	2	3	4	5	6	7
RETRVE or ADDRTR	RANDOM or RANSEQ	READ KEY	X			X				
		WAITF	X			X				X*
		WRITE KEY	X			*				
		WAITF	X			*				

*Set by READ KEY or the WAITF macro instruction following the READ KEY macro instruction.

Figure F–3. Error Field Bit Settings – Random Processing

F.2.5. Sequential Retrieval of Records

When IOROUT=RETRVE (or ADDRTR) and TYPE=SEQNTL (or RANSEQ) are specified, records can be retrieved from a file in sequential order using the four macro instructions SETL, GET, PUT, and ESETL. Normally, the PUT and ESETL macro instructions set only bit 0 of byte 1 in the error field. Additional bits may be set by the SETL or GET macro instructions. Figure F–4 summarizes the bit settings for errors which occur during the sequential retrieval of records.

- Bit 2 – End of File

  If a GET macro instruction attempts to make the next record available to the user program and an end-of-file indication is detected, this bit is set. The next macro instruction for the file must then be an ESETL macro instruction which terminates the retrieval sequence.

  If the user has equated a label to the EOFA keyword parameter in the DTFIS declarative macro instruction, control is transferred to the user routine; otherwise, control returns to the normal place.

- Bit 3 – No Record Found

  1. If the second parameter of a SETL macro instruction is KEY or GKEY, it is possible that no record with a key equal to or greater than the key argument value could be found in the file. This does not preclude further processing of the file; another SETL macro instruction can be attempted, a READ KEY macro instruction can be executed if DTFIS supports random retrieval, or a WRITE NEWKEY macro instruction can be issued if DTFIS supports the ADD function. If a GET, PUT, or ESETL macro instruction is attempted, it is ineffective and the bit remains set.

  2. It is possible for bit 3 to be set during the execution of the GET, PUT, or ESETL macro instruction; this reflects a hardware error. An indicator in the DTFIS-generated table marking the file as open is turned off. Further access to the file results in a file-not-open error condition.

- Bit 4 – No ID Record Found

  1. If no record with an ID equal to that specified in a SETL macro instruction can be found in the file, or if the ID is not in a legitimate address range for the file (out of bounds), this bit is set. Further processing of the file is affected in the same manner as described for part 1 of bit 3.

  2. If the second parameter of a SETL macro instruction is KEY or GKEY, the identification in the data portion of the track index entry is not within the extents assigned to the files.

- Bits 5 and 6 – Unused

- Bit 7 – Record Retrieval from Overflow

  If the logical record retrieved by a GET macro instruction was taken from an overflow area, this bit is set. Processing of the record and the file proceeds normally.

IOROUT	TYPE	MACRO INSTRUCTION	BYTE 1 – filename C							
			0	1	2	3	4	5	6	7
RETRVE or ADDRTR	SEQNTL or RANSEQ	SETL	X			X	X			
		GET	X		X	X				X
		PUT	X			X				
		ESETL	X			X				

Figure F–4. Error Field Bit Settings – Sequential Retrieval

APPENDIX G. ISAM FILE STRUCTURE

G.1. GENERAL

This appendix provides the ISAM user with the basic file structure of an ISAM file. It is intended to give a more detailed description of the internal ISAM structure than was presented in 4.2. Two complete examples, one illustrating initial loading of an ISAM file and the other showing file structure with overflow records, are given at the end of this appendix.

G.2. SPACE ALLOCATION

An indexed sequential file may be defined on one volume or on as many as eight volumes. All volumes of the file must be mounted and ready before the file is opened. The index area is always specified as being on volume 1. There may be eight prime data areas described, one for each volume. The independent overflow area may be on any volume but must only be described as one area.

The volume table of contents (VTOC) for each volume will contain a format 1 label, which contains the 44-byte filename (from the LBL card) and the disc areas (extents) assigned to the file for this volume. A format 2 label is built to maintain ISAM file information on volume 1. The VTOC also contains a description of the device characteristics of the particular volume. These values are used to calculate effective record lengths and track configurations for each type of device.

G.2.1. Disc Address

ISAM disc addresses are described either by a five-byte disc identification (ID) field or a full eight-byte file-oriented address field.

The five-byte disc ID address is of the form cchhr where,

- cc = two-byte cylinder number
- hh = two-byte head or track number
- r = one-byte record number

The first byte of cc and hh is always zero for the UNIVAC 8411 and 8414 Disc Subsystems. The first byte of cc may not be zero for the UNIVAC 8424 Disc Subsystems.

ISAM usually maintains an eight-byte disc address since it contains volume information. This address is of the form mbbcchhr where,

- m = one-byte volume number (from one to eight)
- bb = two bytes of zeros for the UNIVAC 8411, 8414 and 8424 Subsystems
- cchhr = five-byte disc ID as described previously

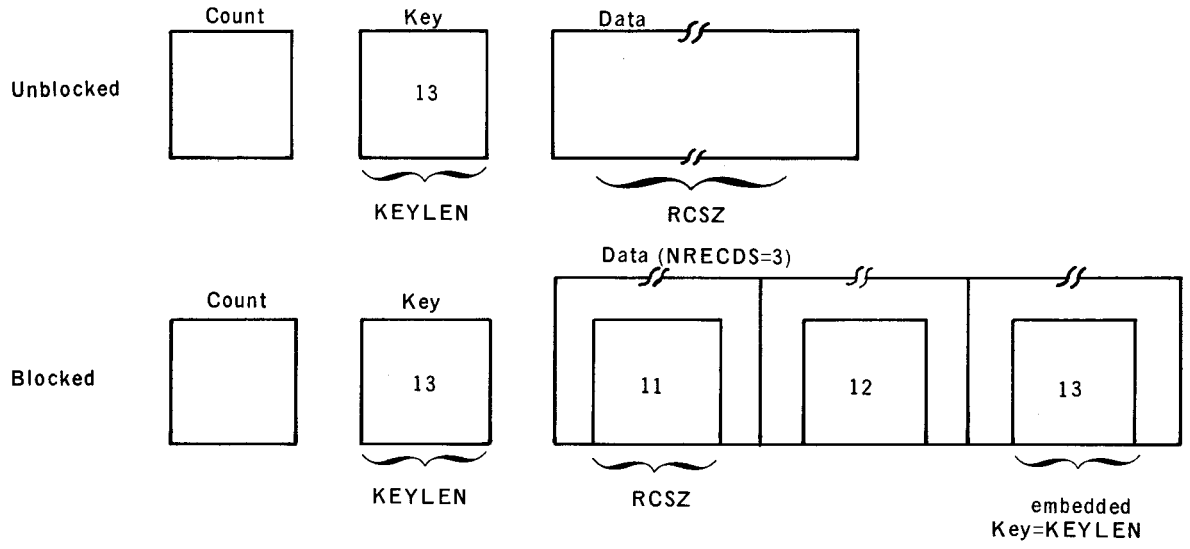
When issuing Seek or Search ID commands to the disc subsystem, the M is used to choose the correct volume device address; bbcchh is used as the Seek argument, and cchhr as the Search ID argument. The disc ID forms the first five bytes of the count area of each disc record.

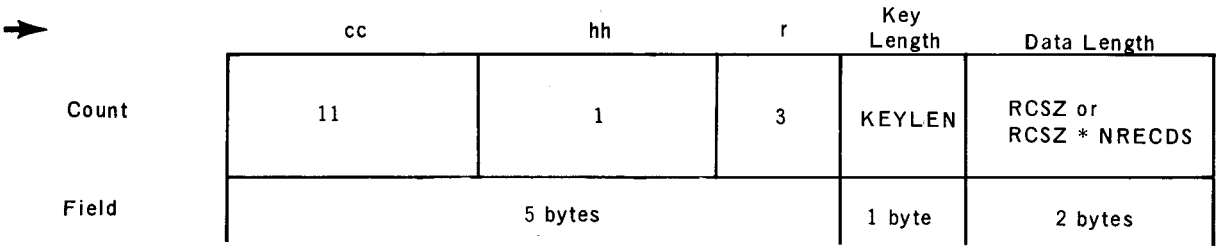
G.2.2. Prime Data Records

Records in the prime data areas are stored sequentially by ascending keys. For unblocked records, the user's key becomes the physical key on the disc; for blocked records, the key of the last logical record of the block is stored as the physical key of the record.

Each record is of fixed length and is written onto the disc with count, key, and data fields. The count field contains the disc ID of the record, the key-length (from DTFIS keyword parameter), and the data length (unblocked = record size; blocked = record size times the blocking factor).

For example, a breakdown of the fields for a record with a user key of 13 gives:



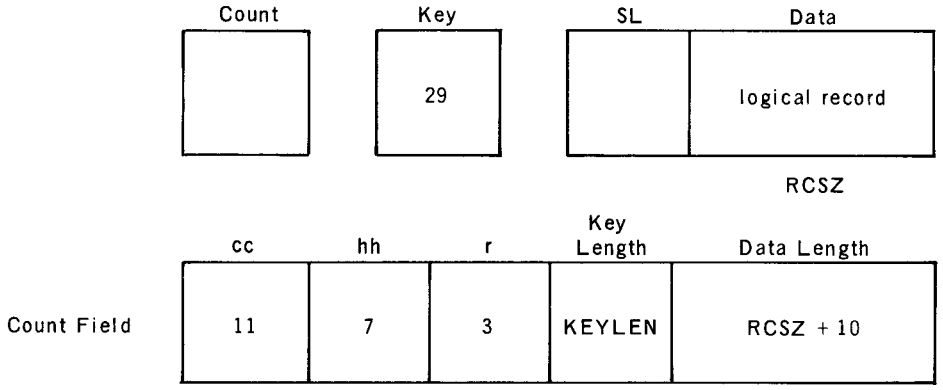


G.2.3. Overflow Records

After a file has been loaded, records with new keys may be inserted sequentially into the file. Areas for containing overflow records may be set aside by the user by reserving tracks associated with a particular cylinder (cylinder overflow) or by reserving an area associated with the entire file (independent overflow area).

Overflow records are always stored in unblocked fixed-length format. Each record is written on the disc with count, key, sequence link, and data fields. The count field contains the disc ID of the record, the key-length (from DTFIS), and the data length (always = RCSZ+10). The key field contains the physical key of the logical record, while the data field contains a 10-byte sequence link field to chain overflow records (described in G.3.2) and the logical record.

For example, a breakdown of the fields for an overflow record with key = 29 gives:



G.3. INDEXES

For an ISAM file, two levels of indexes are maintained (a track index and a cylinder index) as well as an optional higher-level third index called a master index.

The track index is the lowest level of index and appears at the beginning of each prime data cylinder. The number of track index entries needed for each cylinder and the format of the track index are determined by the loading procedure which is based on file specifications. At this time, it is also determined whether or not there is enough room on the track after the track index to allow for data records to share the track.

The cylinder index occupies the space set aside by the user as the index extent. If a master index is requested, it immediately precedes the cylinder index. Each entry in the master index contains the highest key on a cylinder index track.

All index entries are of the same format, and contain count, key, and data areas. The count field of eight bytes is structured in the same way as that of a prime data record. The key field (length determined by the KEYLEN parameter in DTFIS) is used for searching the indexes, while the data field is always 10 bytes. The data field contains a full eight-byte disc address of either the next lower level index, the prime data track, or the first record of an overflow chain. The ninth byte of the data field, called the "f" flag, specifies the type of index entry, the validity of the entry, and the type of search required next. The tenth byte is unused and is merely maintained for compatibility with other systems.

The indexes are terminated with "dummy" entries (shown as D in the examples), with keys of all 1 bits (highest key in the collating sequence) and data areas of all zeros. These entries terminate the index searching and isolate "no find" conditions in the indexes.

Chaining entries (in the case of cylinder and master indexes only) are used to link cylinders of index entries. They appear in the last record of the last track of each cylinder of index entries. A chaining entry has a key of all 1 bits and contains the address of the next cylinder of index entries. When index scanning takes place in multiple track mode, these entries eliminate a premature "no find" condition. The entries are not illustrated in the examples because they are of no value to the user.

G.3.1. Track Index

For each track containing prime data records within the cylinder, two entries are provided in the track index: the track normal entry and the track overflow entry. A dummy entry terminates the track index. As the file is loaded, the highest key of each prime data track is stored in both its normal and overflow track index entries. The data areas are set up to point to the prime data track as illustrated in Example 1 at the end of this appendix.

■ Track Index Normal Entry (for a track shared with data records):

Key	Data Field					
	m		cc	hh	r	f
4	1		11	0	16	X'08'

NOTE: r is the first prime data record number on a shared track. The f flag indicates a shared track condition.

■ Track Index Normal Entry (for a track not shared with data records):

Key	Data Field					
	m		cc	hh	r	f
8000	2		5	4	0	X'00'

NOTE: r=0 and the f flag indicate that the track contains only data records.

■ Track Index Overflow Entry (with no active overflow area):

Key	Data Field					
	m		cc	hh	r	f
4	1		11	0	X'FF'	X'10'

NOTE: r=X'FF' and f flag indicate no overflow.

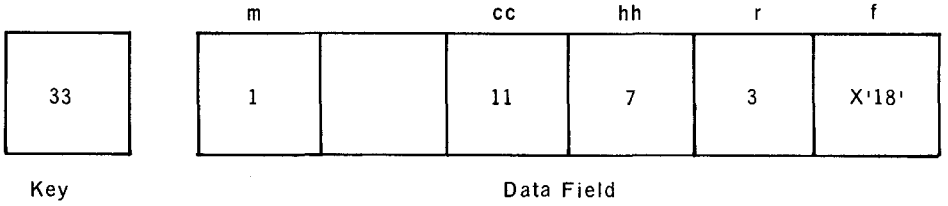
As records are inserted into the file in the overflow areas, the address in the track index overflow data fields is altered to point to the first record in the overflow area. The key of the overflow entry still reflects the highest key originally associated with the prime data track, but the key of the track index normal entry is altered to reflect the highest key remaining on the prime data track. For example:

■ Track Index Normal Entry (after adding records):

Key	Data Field					
	m		cc	hh	r	f
27	1		11	1	0	X'00'

NOTE: Key=27 is the highest key on prime data track 1, after adding three records to the file.

- Track Index Overflow Entry (with an overflow area):

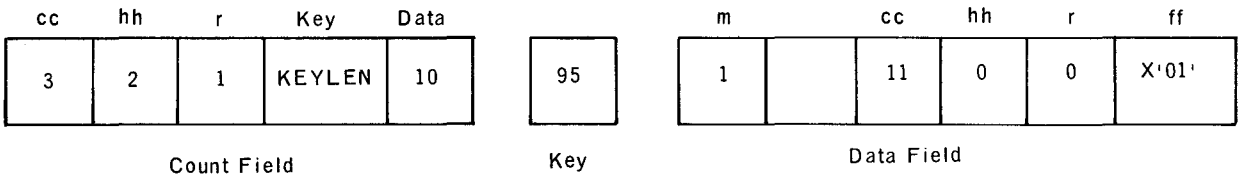


NOTE: Key=33, which was the highest key on prime data track 1, is now the highest key in the overflow area. The address (cchhr) points to the disc address of the first record in the overflow area, which in this case is the third record placed there. The f flag is changed to indicate that the TI entry is now active. Refer to Example 2 for further explanation of overflow record entries.

G.3.2. Cylinder Index

As the file is being loaded, an entry is made into the cylinder index for each cylinder of prime data records. The key of the cylinder index entry is the highest prime data key within the cylinder. The data field of the cylinder index entry gives the address of the corresponding prime data cylinder.

The cylinder index entry for the first prime data cylinder (Example 1) would be:

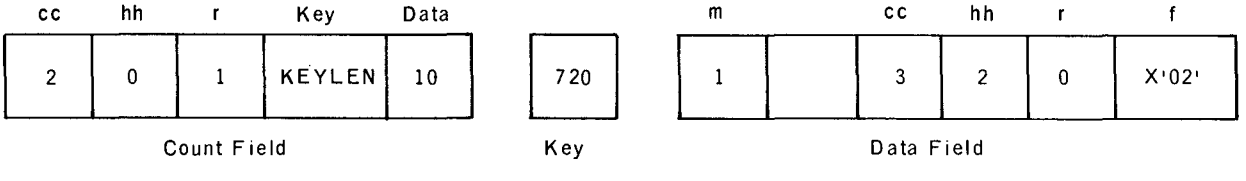


NOTE: The count field address (cchhr) is an approximation. Key=95 is the highest prime data key in the cylinder, and the f flag indicates an active cylinder entry.

G.3.3. Master Index

If a master index is requested, an entry is made for each track containing cylinder index entries. The key of the master index entry is the highest key on the cylinder index track, while the data field gives the address of the corresponding cylinder index track.

The master index entry for the first cylinder index track (Example 1) would be:



NOTE: The count field address (cchhr) is an approximation. Key=720 is the highest key on the cylinder index track, and the f flag indicates an active master index entry.

G.4. OVERFLOW CONTROL

When a record is inserted into an ISAM file, it is placed in the prime data track or overflow area sequentially by key. Thus, if a record with key=18 is inserted into the file (see Examples 1 and 2 at the end of this appendix), it would be placed between keys 16 and 19. All succeeding records would be moved down the track and the displaced record (key=33) would become an overflow record.

G.4.1. Cylinder Overflow Control Record

If the user has specified cylinder overflow, a cylinder overflow control record is written in the first record area (record 0, head 0) of each cylinder during loading. This record maintains the address at which a cylinder overflow record can be written. The cylinder overflow control record has no key field, but it is eight bytes long and has the form hhrbbtxx, where

hh = two bytes indicating the last track containing overflow records.

r = one byte indicating the last overflow record number.

t = one byte indicating the remaining number of available tracks in the cylinder overflow area.

bb and xx = 0's.

An unused cylinder overflow control record (Example 1) would be:

hh	r	bb	t	xx
7	0	0	3	0

The current cylinder overflow control record (Example 2) would be:

hh	r	bb	t	xx
7	3	0	3	0

If the cylinder overflow area is full or if only independent overflow was requested, the overflow record is written in a separate area. Fields in the DTFIS are maintained with the last independent overflow record address and the remaining number of available independent overflow tracks.

G.4.2. Sequence Link Field

Each of the records in an overflow area originally moved from the same prime data track are chained together by means of a sequence link field. The data field of the track index overflow entry contains the address of the first overflow record. The first 10 bytes of the data area of this record (the sequence link field) contains the address of the next overflow record. The last record in the chain contains the home address of the original prime data track.

The sequence link field is formatted in the same way as an index data entry; that is, it contains an eight-byte disc address, an f flag, and an ignored 10th byte.

The sequence link field for key=29 contains the address of key=33, and the f flag shows that this entry is not the last in the chain (Example 2).

m		cc	h	r	f
1		11	7	1	X'18'

The sequence link field (see Example 2) for key=33 contains the address of track 1, and the f flag indicates that this is the end of the chain.

m		cc	hh	r	f
1		11	1	X'FF'	X'10'

G.5. ISAM FILE PROCEDURES

Various file procedures for adding records and retrieving records (either randomly or sequentially) are covered in the following sections.

G.5.1. Adding Records to File

Add to the ISAM file, illustrated in Example 1, three records with the following keys: key=18, key=1, and key=24 respectively. The final results are shown in Example 2.

- Key=18

  (1) Search indexes (see G.5.2) for a track index entry with key > 18. Result: a track index normal entry with key=33 is found which points to prime data track 1.

  (2) Scan track for first key > 18. Result: the record with key=18 is inserted between records with key=16 and key=19.

  (3) Enter record with key=18 onto prime data track and move all succeeding records down the track one logical record position. Result: the record with key=33 is pushed off the track and must be written in an overflow area.

  (4) The cylinder overflow control record specifies that track 7, record 1 is available. Write the record with key=33 in overflow record format, with a sequence link field which points to the home address of track 1. Alter the cylinder overflow control record to reflect that record 1 has been used.

  (5) Alter the track index overflow entry data field to point to the disc address of record key=33 (first record in overflow chain).

  (6) Alter the track index normal entry key field to record key=29 (highest key on prime data track).

- Key=1

  (1) Follow procedure for key=18 as previously described. Result: a record with key=4 is now written in the overflow area. The track index overflow entry data field points to record key=4. The track index normal entry key is set to key=2.

- Key=24

  (1) Follow procedure for key=18 (steps 1 through 3). Result: a record with key=29 must be written in the overflow area.

  (2) Get the ID from track index overflow entry data field and put it in the sequence link field of the record with key=29.

  (3) Write record key=29 to overflow area based on the cylinder overflow control record. Alter the cylinder overflow control record.

  (4) Alter the track index overflow entry data field to point to a record with key=29.

  (5) Alter the track index normal entry key field to key=27.

G.5.2. Random Retrieval on Prime Data Track

Random retrieval of record with key=2; see Examples 1 and 2.

(1) Search the master index for an entry with key $\geq$ 2. Result: a record with key=720 is found which points to a cylinder index track address.

(2) Search the cylinder index track for an entry with key $\geq$ 2. Result: a record with key=95 is found which points to a prime data cylinder.

(3) Search the track index of the prime data cylinder for an index entry with key $\geq$ 2. Result: a record with key=2 is found from a track index normal entry which points to a prime data track address.

(4) Search the prime data track for a record with key=2. Result: read the record with key=2.

G.5.3. Random Retrieval From an Overflow Area

Retrieve a record with key=33; see Examples 1 and 2.

(1 and 2) Same as for random retrieval on prime data track; that is, a record with a key=33 would yield the same master and cylinder index entries.

(3) Search the track index for an entry with key $\geq$ 33. Result: a record with key=33 is found in a track index overflow entry and its data address yields an overflow record ID.

(4) Compare key of this overflow record (key=29) to the record with key=33. Result: no match can be found; therefore, only read the sequence link field, which points to the next record in the chain.

(5) Compare the key of the record whose address is in the sequence link field (key=33) to the record having key=33. Result: match can be found; therefore, read the record.

For random retrieval and add operations, all or part of the cylinder index may reside in main storage. This will eliminate disc accesses to the higher-level indexes.

G.5.4. Sequential Retrieval Starting With Specified Key

Start sequential retrieval with key=2; see Example 2.

- Issue a SETL macro instruction to retrieve a record with key=2.

  (1...3) Same as for random retrieval on a prime data track.

  (4) Search prime data track for a record with key=2 and save the address of the record.

- Issue GET macro instructions to retrieve the records. For each GET instruction issued, the results are:

  (1) First GET instruction: the record with key=2 is retrieved from track 0.

  (2) Second GET instruction: the prime data record is at the maximum value (physical end) for a shared track; therefore, read the next overflow track index entry. Since it is marked "active", start the retrieval process from the overflow area using the disc address in the track index overflow entry. Result: the record with key=4 is retrieved from track 7; also, save the sequence link field for this record.

  (3) Third GET instruction: examine the sequence link field. Since "f" flag indicates an end-of-chain condition, update the prime data track address. Result: the record with key=5 from track 1 is retrieved.

When the end of prime data track 1 is reached, the next track index overflow entry is examined, and a GET macro instruction would then retrieve the record with key=29. For a subsequent GET instruction, an active sequence link field would retrieve the record with key=33.

The GET, PUT and ESETL macro instructions do not require access to the master and cylinder index areas, nor to the track index normal entries.

G.5.5. Sequential Retrieval Starting With Specified ID

Start sequential retrieval with the eight-byte ID (the ID of the record with key=2).

m		cc	hh	r
1		11	0	17

- Issue SETL Instruction.

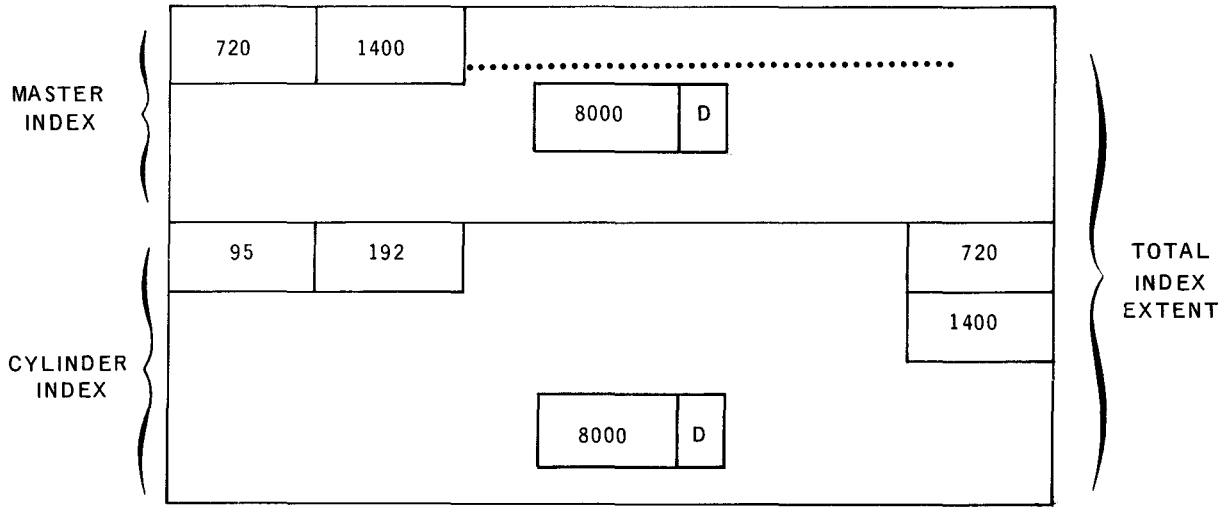
  (1) Determine if the ID is on a prime data track, in a cylinder overflow area, or in an independent overflow area. Result: the record is found to be on a prime data track.

  (2) Calculate the associated track index entry address based on the input ID. No index searching is required for this example.

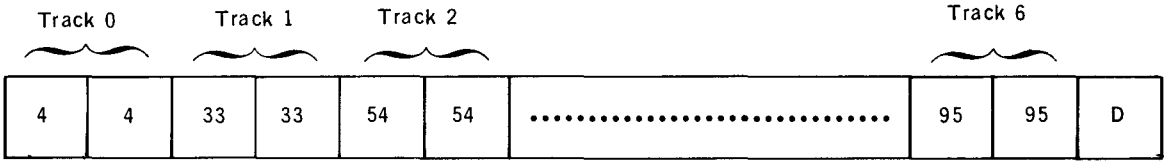
- Issue GET instructions.

Example 1: Initial Load of an ISAM File, Showing Key Structure

- Master and cylinder indexes share the user-specified index extent. Assume cylinders 2 through 10 on volume 1 have been assigned as the index extent.



- For the track index of the first prime data cylinder, assume cylinder 11 on volume 1. Three tracks have been set aside for cylinder overflow, and track 0 is shared with data records. Therefore, the track index contains 15 entries, including the dummy entry.

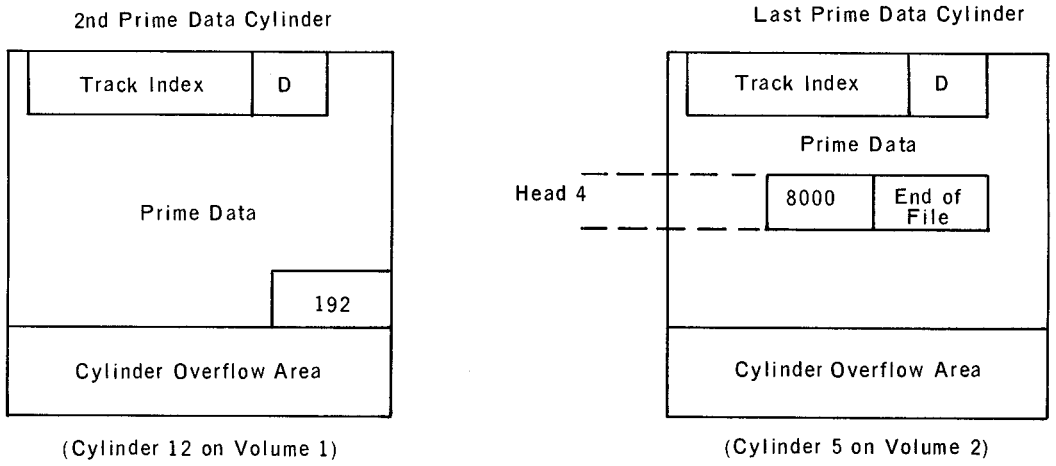


- The first prime data cylinder is loaded with unblocked records, as illustrated. Since cylinder overflow has been specified, a cylinder overflow control record (COCR) is written on track 0, record 0; and track 0 is shared with data records.

First Prime Data Cylinder

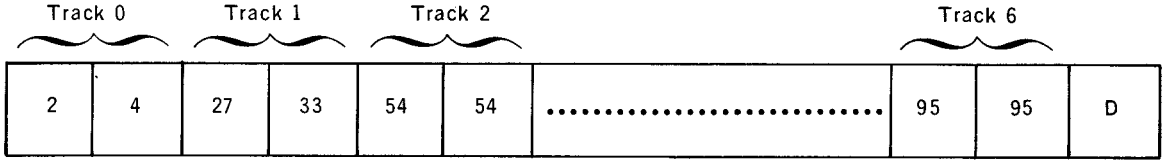
Track										
0	COCR	Track Index							2	4
1	5	7	13	16	19	22	26	27	29	33
2	36	37	40	43	46	47	49	51	53	54
3										
4										
5										
6										95
7	Cylinder									
8										
9	Overflow Area									

- The second and last prime data cylinders are shown in order to relate entries to the higher-level indexes.

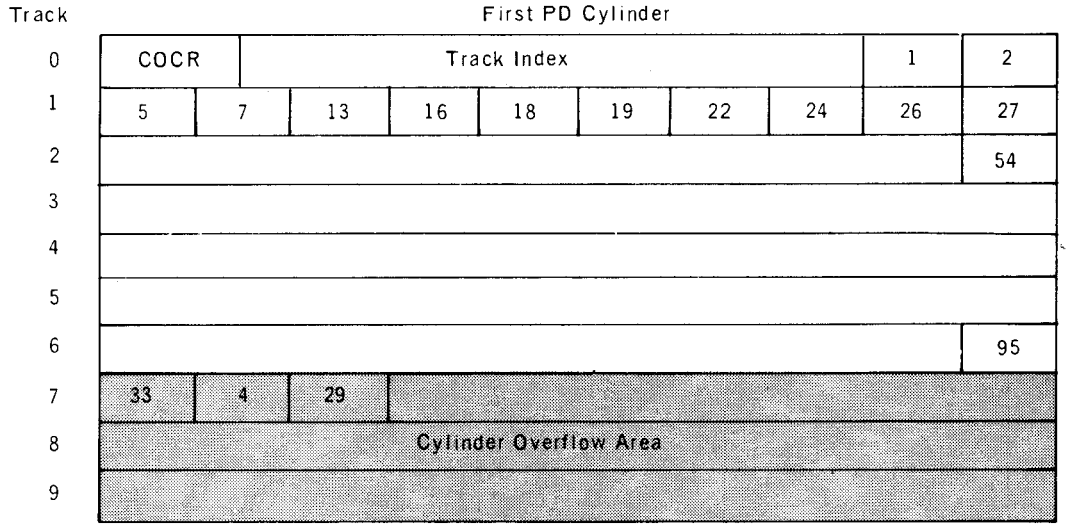


Example 2: ISAM File Structure with Overflow Records

- After the addition of three records into the file, with keys=18, 1, and 24 respectively, the track index entries for tracks 0 and 1 are altered as follows:



- The first prime data cylinder now shows three entries in the cylinder overflow area.



APPENDIX H. DISC SPACE REQUIREMENTS FOR ISAM FILES

H.1. GENERAL

This appendix describes the space requirements for the UNIVAC 8411, 8414, or 8424 Disc Subsystem for the various elements of ISAM files. Constraints on the disc extents are detailed, and the formulas (including examples of their use) for computing the disc space requirements are given.

H.2. CONSTRAINTS ON ISAM DISC EXTENTS

A maximum of eight volumes can be used for the file. There can be only one prime data extent per volume, but the index or independent overflow extents, or both, can be on the same volume as a prime data extent. A prime data extent must begin and end on cylinder boundaries, but can be located anywhere on the volume. If there is more than one prime data extent for the file, the extents need not be continuously addressable.

There is only one index extent for the file, which always contains the cylinder index. If the file is to have a master index, the required space is allocated in the low-order portion of the index extent. The index extent must be in the first volume of the file. If there is another extent for the file in the same volume, the index extent must come first; that is, it must appear in an EXT Job Control statement before the other extent, but it does not have to occupy the low-order portion of the disc. Also, the index extent need only be an integral number of tracks, and it does not have to begin or end on cylinder boundaries.

The use of an independent overflow area is optional; if an overflow area is used, the required extent can appear on any volume of the file. It must be an integral number of cylinders and must begin and end on cylinder boundaries.

H.3. FORMULAS FOR DISC SPACE REQUIREMENTS

Disc space requirements are derived by using the proper formulas. Information concerning these formulas is presented under the following subheadings:

- Notes and conventions
- Basic calculations
- Prime data cylinder capacity
- Higher level index requirements
- Overflow area requirements

H.3.1. Notes and Conventions

(1) The formulas which follow do not explicitly refer to the track descriptor record (R0) on each track or to the count area of each physical record, but they do allow space for them.

(2) The notation used is standard, but the following conventions are submitted for clarification:

(a) Multiplication is denoted by an asterisk (*) because of typing considerations.

(b) Division is normally real; however, where the numbers are in brackets, the real number quotient is to be truncated to an integer (by ignoring its fractional portion and keeping only the integer). For example:

3625/250 = 14.5, and 3/4 = 0.75, but
[3625/250] = 14, and [3/4] = 0.

(3) Differences in formulas with respect to the UNIVAC 8411 Disc Subsystem, UNIVAC 8414 Disc Subsystem, and the UNIVAC 8424 Disc Subsystem are represented by four factors expressed in lowercase letters. These factors are constants whose values for each subsystem can be found in Table H–1.

DESCRIPTION	CONSTANT	8411 DISC	8414/8424 DISC
Number of tracks per cylinder	ntrk	10	20
Adjusted number of bytes per track	trksz	3605	7249
Tolerance factor	tol	1.05	1.044
Record overhead	ohd	81	146

Table H–1. UNIVAC 8411 and 8414/8424 Formula Differences

H.3.2. Basic Calculations

B1. Index record length (IRL)

IRL = ((KEYLEN +10)*tol)+ohd

B2. Maximum number of index records per track (MXIR)

MXIR = [(trksz–(KEYLEN+10))/IRL]+1

B3. Number of bytes in a data block (BDB)

BDB = KEYLEN+(RCSZ*NRECDS)

where: NRECDS is taken as 1 for unblocked records

B4. Data block length (DBL)

DBL = (BDB*tol)+ohd

B5. Maximum number of data blocks per track (MXDB)

MXDB = [(trksz–BDB)/DBL]+1

B6. Number of data blocks on a shared track (SHDB)

SHDB = [(trksz–BDB–(n*IRL))/DBL]+1

where: n is the number of TI entries on the shared track (see H.3.3.1 and H.3.3.3).

B7. Overflow record length (OFRL)

OFRL = [(KEYLEN+RCSZ+10)*tol]+ohd

B8. Number of overflow records per track (NOFR)

NOFR = [(trksz–(KEYLEN+RCSZ+10))/OFRL]+1

H.3.3. Prime Data Cylinder Capacity

The capacity of a prime data cylinder is defined as the number of logical records that it can contain. This number does not include any logical records in overflow areas associated with the cylinder. Records in a prime data area are referred to as prime data records, while those in an overflow area are referred to as overflow records.

The capacity of a prime data cylinder depends on the number of tracks available for prime data records and the number of records that can fit on each track. The number of tracks available depends on the number of tracks set aside for cylinder overflow (from a minimum of 0 to a maximum of 8) and the amount of space required by the track index (from a minimum of a fraction of track 0 to a maximum of tracks 0 and 1 for the UNIVAC 8411 disc, or a maximum of tracks 0 and 1 and a fraction of track 2 for the UNIVAC 8414 disc and the UNIVAC 8424 disc). There are six possible variations with respect to the size of the track index (TI) as follows:

- Track 0 Shared

  The track index resides on track 0 with enough space left over for at least one data block. There are 2n+1 TI entries, where n equals ntrk minus the number of cylinder overflow tracks, with two entries for each track from 0 through n−1, and a dummy entry.

- Track 0 Nonshared

  The track index resides on track 0. No space is left over for data blocks. There are 2n−1 track index entries, two entries for each track from 1 through n−1, and a dummy entry.

- Tracks 0 and 1 Shared

  The track index is too large to fit on track 0, and the extra entries are placed on track 1. There is enough space left over on track 1 for at least one data block. There are 2n−1 track index entries, two for each track from 1 through n−1, and a dummy entry.

- Tracks 0 and 1 Nonshared

  The track index is too large to fit on track 0, and the extra entries are placed on track 1. No space is left over on track 1 for data blocks. There are 2n−3 track index entries, two for each track from 2 through n−1, and a dummy entry.

- Tracks 0, 1, and 2 Shared

  The track index is too large to fit on tracks 0 and 1. The extra entries are placed on track 2. There is enough space left over on track 2 for at least one data block. There are 2n−3 entries, two entries for each track from 2 through n−1, and a dummy entry.

- Tracks 0, 1, and 2 Nonshared

  The track index fits on tracks 0, 1, and 2. No space is left over on track 2 for data blocks. There are 2n−5 entries, two for each track from 3 through n−1, and a dummy entry.

- Track Index Size Formulas

  The formulas and procedures that follow show how to compute the size of the track index and the capacity of a prime data cylinder. Many of the factors that enter into these calculations are derived from the basic calculations given in H.3.2. Current track (CURTRK) can be track 0, 1, or 2. Initially, the current track is track 0; therefore, CURTRK=0.

  S1. Number of prime data tracks (NTPD)

  NTPD = ntrk−(number of cylinder overflow tracks)

  S2. Number of track index entries (NTI)

  NTI = (2*NTPD)+1

  S3. Does track index fit on current track? Test (NTI:MXIR).

  NTI>MXIR − track index does not fit, go to S4.

  NTI = MXIR − track index fits, but current track is nonshared, go to S6.

  NTI<MXIR − track index fits, and current track is possibly shareable, go to S5.

  S4. Decrement NTPD by 1 and recompute NTI as NTI = (2*NTPD)+1 − (MXIR*CURTRK).

  Does track index now fit on current track? Test (NTI:MXIR).

  NTI>MXIR − track index still does not fit, go to S7.

  NTI≤MXIR − track index now fits. Set SHDB to 0, go to S8.

  S5. Test shareability of current track.

  (a) Compute space available for data (SAD)

  SAD = trksz−(NTI*IRL)

  (b) Is there space for at least one data block? Test (BDB:SAD).

  BDB>SAD − No, the track is not shareable. Go to S6.

  BDB<SAD − Yes, the track is shareable. Compute the number of data blocks on the shared track, SHDB.

(c) SHDB = [(SAD−BDB)/DBL] +1

(d) Go to S8.

S6. Decrement NTPD by 1 and set SHDB to 0. Go to S8.

S7. Compute the number of track index entries on current track + 1.

CURTRK = CURTRK+1.
NTI = NTI−MXIR

Go to S3.

S8. Compute cylinder capacity in blocks (CBC) and records (CRC).

(a) If SHDB = 0, CBC = NTPD*MXDB.

If SHDB ≠ 0, CBC = ((NTPD−1)*MXDB)+SHDB.

(b) CRC = CBC*NRECDS,(NRECDS is taken as 1 for unblocked records).

S9. Compute the total file capacity in blocks (FBC) and records (FRC).

(a) FBC = CBC*(number of prime data cylinders)−MXDB

(b) FRC = FBC*NRECDS,(NRECDS is taken as 1 for unblocked records).

(c) To find the number of cylinders required to hold a given number of records, add MXDB*NRECDS to the actual number of records, divide this sum by the cylinder capacity in records (CRC), and round upwards.

H.3.4. Higher Level Index Requirements

There is a cylinder index entry for each prime data cylinder. If there is a master index, it has an entry for each track occupied by the cylinder index. For each type of index, there is a dummy entry marking the end of the index, and a chaining entry which is required each time the index crosses a cylinder boundary. Thus, if an index has entries on n cylinders, there are n−1 chaining entries.

In the control stream, one extent (the first entry on the EXT statement for the first volume) should be assigned as the higher level (master index, MI, or cylinder index, CI) extent, but it need not begin or end on cylinder boundaries. The program subdivides the space between the CI and the MI, allowing the maximum possible space for each. (Of course, if an MI is not used, space is not allocated for one.) The CI is checked for adequate space against the number of prime data cylinders; if the CI space is adequate, the MI space is also adequate.

The MI, if present, occupies one or more tracks at the beginning of the extent, while the CI begins with the track immediately following the last MI track.

Depending on the actual disc addresses involved, there can be some variation in the size of the MI and CI because of the presence or absence of chaining entries. The formulas which follow (I1 through I5) allow for the maximum size. Quotients are to be rounded to the next higher integer.

I1. Minimum number of CI entries (MNCI)

MNCI = (# Prime Data cylinders) + 1

I2. Number of CI tracks (NCIT)

NCIT = (MNCI+((MNCI/MXIR)−1)/ntrk)/MXIR

I3. Minimum number of MI entries (MNMI)

MNMI = NCIT+1

I4. Number of MI tracks (NMIT)

NMIT = (MNMI+((MNMI/MXIR)−1)/ntrk)/MXIR

I5. Number of higher level index tracks (NHLT)

NHLT = NMIT+NCIT

H.3.5. Overflow Area Requirements

Formulas B7 and B8 (see H.3.2), which give overflow record length (OFRL) and the number of overflow records per track (NOFR), respectively, are all that are required to compute the disc space requirements for a given number of overflow records. They apply to both the cylinder and the independent overflow areas.

H.4. SAMPLE DISC SPACE CALCULATIONS

The following two sample calculations are for unblocked and blocked records, respectively.

H.4.1. Unblocked Records, Cylinder Overflow, Track 0 Shared

Initial volume is 90,000 records, each record is 80 bytes long with a 10-byte key, and one track per cylinder is allowed for overflow. DEVICE = 8411.

(1) Basic Calculations

B1. IRL = (10+10)*1.05+81 = 102

B2. MXIR = [(3605-(10+10))/102] = 35

B3. BDB = 10+(80*1) = 90

B4. DBL = (90*1.05)+81 = 175.51

B5. MXDB = [(3605-90)/175.51] +1 = 21

(2) Prime Data Cylinder Capacity

S1. NTPD = 10-1 = 9

S2. NTI = (2*9)+1 = 19

S3. (19<35)

S5. (a) SAD = 3605-(19*102)=1667
(b) (90<1667)
(c) SHDB = [(1667-90)/175.51] +1 = 9

S8. (a) CBC = (9-1)*21+9 = 177
(b) CRC = 177*1 = 177

S9. # Prime Data cylinders required = (90000+21)/177 = 509

(3) MI/CI Requirements

I1. MNCI = 509+1 = 510

I2. NCIT = (510+((510/35)-1)/10)/35 = 15

I3. MNMI = 15+1 = 16

I4. NMIT = (16+((16/35)-1)/10)/35 =1

I5. NHLT = 1+15 = 16

(4) Overflow Area Requirements

B7. OFRL = ((10+80+10)*1.05) +81 = 186

B8. NOFR = [(3605-(10+80+10))/186] +1 = 19

At one overflow track per cylinder, there are 9671 possible overflow records, or approximately 10 to 11 percent of the prime data capacity.

H.4.2. Blocked Records, No Cylinder Overflow, Track 0 Nonshared

Initial volume is 2,500 records, each record is 200 bytes long with 50-byte embedded keys, records are blocked at 3-per-block and cylinder overflow is not specified. DEVICE = 8411.

(1) Basic Calculations

B1. IRL = ((50+10)*1.05) +81 = 144

B2. MXIR = [(3605-(50+10))/144] +1 = 25

B3. BDB = 50+(200*3) = 650

B4. DBL = (650*1.05)+81 = 682.5

B5. MXDB = [(3605-650)/682.5] +1 = 5

(2) Prime Data Cylinder Capacity

S1. NTPD = 10-0 = 10

S2. NTI = (2*10)+1 = 21

S3. (21<25)

S5. (a) SAD = 3605-(21*144) = 581
(b) (650>581)

S6. NTPD = 10-1 = 9

SHDB = 0

S8. (a) CBC = 9*5 = 45
(b) CRC = 45*3 = 135

S9. # Prime Data cylinders required = 2500+(5)*(3)/135 = 19

(3) MI/CI Requirements

I2. MNCI = 19+1 = 20

I2. NCIT = (20+((20/25)-1)/10)/25 = 1

NOTE: In this case, with just one track for cylinder index, a master index would be pointless.

H.4.3. Unblocked Records, No Cylinder Overflow, Track 2 Shared

Initial volume specifies a key length of 250 and record size of 270. DEVICE = 8414.

(1) Basic Calculations:

B1. IRL = [(250+10)*1.044] +146 = 417.44

B2. MXIR = [(7249-(250+10))/417.44] +1 = 17

B3. BDB = 250+(270*1) = 520

B4. DBL = (520*1.044)+146 = 688.88

B5. MXDB = [(7249-520)/688.88] +1 = 10

(2) Prime Data Cylinder Capacity

S1. NTPD = 20

S2. NTI = (2*20)+1 = 41

S3. 41>17 Track index does not fit on track 0.

S4. NTPD = 20-1 = 19
NTI = (2*19) +1–(17*0) = 39
39>17 Track index still does not fit on track 0.

S7. CURTRK = 0+1 = 1
NTI = 39-17 = 22

S3. 22>17 Track index does not fit on track 1.

S4. NTPD = 19-1 = 18
NTI = (2*18) +1–(17*1) = 20
20>17 Track index still does not fit on track 1.

S7. CURTRK = 1+1 = 2
NTI = 20-17 = 3

S3. 3<17 Track index now fits on track 1 and current track (track 2) is possibly shareable.

S5. (a) SAD = 7249-(3*417.44) = 5997
(b) 52<5997 Track 2 is shareable.
(c) SHDB = [(5997-520)/688.88] +1 =9

S8. (a) CBC = ((18-1)*10) +9 = 179
(b) CRC = 179

S9. (a) FBC = 179*(number of prime data cylinder)-10
(b) FRC = FBC
(c) To hold a file of 100,000 records
100010/179 = 559 cylinders required for prime data extent.

(3) Master Index and Cylinder Index Requirements

I1. MNCI = 559+1 = 560

I2. NCIT = $(560+(\frac{560}{17}-1)/20)/17 = 34$

I3. MNMI = 34+1 = 37

I4. NMIT = (35+((35/17)-1)/20)/17 = 3

I5. NHLT = 3+34 = 37

(4) Overflow requirements

B7. OFRL = [(250+270+10)*1.044]+146 = 699

B8. NOFR = [(7249-530)/699]+1 = 10

APPENDIX I. PRINTER FORMS HANDLING INFORMATION

The printer software prints a maximum of two normally spaced lines (depending on number of I/O areas) whenever forms overflow is recognized. The user must, therefore, be careful not to place forms overflow punches in the carriage tape too close to the home paper punch, as the skip to home position may have already passed the position needed to recognize the home position, and the overflow routine may skip a page.

The UNIVAC 0768 (Model B) Printer no longer has a switch to control 6–8 lines-per-inch spacing. The setting of this option is accomplished in the following manner:

If six-lines-per-inch spacing is desired, punch 14 (2,4,8) in the paper tape loop 'home' position.

If eight-lines-per-inch spacing is desired, punch 15 (1,2,4,8) in the paper tape loop for 'home' position.

Place the carriage tape in the printer and home the paper twice while in off-line mode. The printer will set itself to six or eight line spacing according to the punches for home position. When the printer is placed online, spacing will be according to the setting thus determined. A control command skip-to-14 (or skip-to-15) will then be recognized as skip-to-home, regardless of the line spacing.

Using a UNIVAC 0768 (Model B) Printer, the maximum number of lines spaced after printing is three. Users of earlier model printers (Model A with the manual 6–8 switch) may find it necessary to punch both a 14 and a 15 in adjacent positions on the paper tape loop, in order to accomplish both the skip-to-15 command used by Data Management and the skip-to-14 command which may be in a user's printer overflow routine.

APPENDIX J. DATA MANAGEMENT COMMON CODE

J.1. GENERAL

In a multiprogramming environment, many of the Data Management modules are used by more than one program. Frequently used Data Management modules can be incorporated into the Supervisor during system generation. These modules, which are resident in the Supervisor, are referred to as common code.

The Data Management modules are re-entrant; therefore, those modules resident in common code may be shared by all programs running in that environment. This saves a considerable amount of space in the system. Modules need not be linked into every program being executed. Only those modules frequently used should be included in common code. Less frequently used modules should be linked into those programs which need them.

J.2. DATA MANAGEMENT COMMON CODE UTILITY PROGRAM

The Data Management Common Code Generator (DMCC) is a utility program. The user must specify the Data Management routines to be included in common code by means of PARAM statements (see J.2.7). The output from the DMCC utility program is an optional source deck, a listing of defined labels, and a library tape having the object module of equates in the reserve file and the source code module in the source file (see Figure J–1). A sample control stream for a utility run is shown in J.2.9.

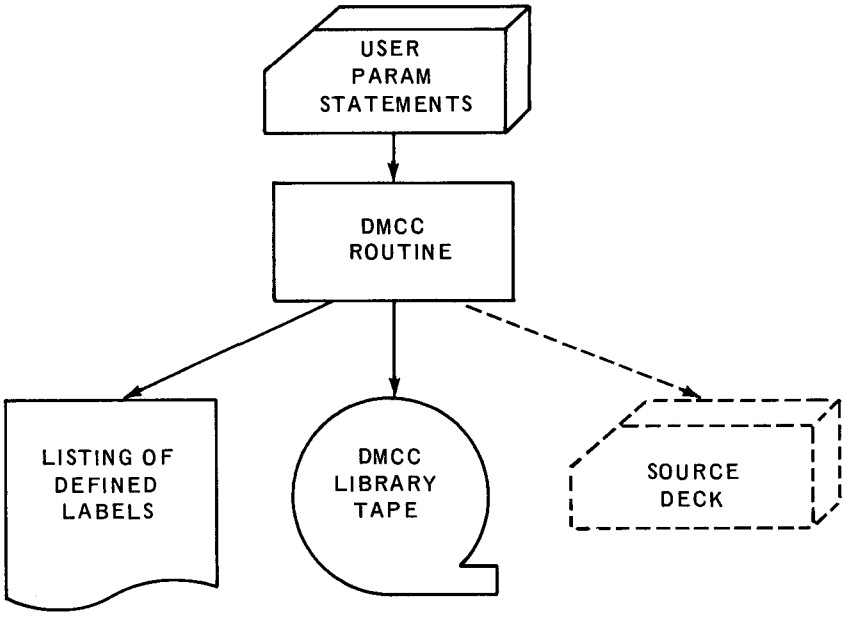


Figure J–1. Data Management Common Code Generator (DMCC)

J.2.1. Listing of Defined Labels

Labels defined for the routines included in common code are listed on the printer. Some Data Management routines provide multiple functions which are found in smaller modules. These larger modules are called supersets. The smaller, more specific modules are called subsets. The superset modules may be used to replace one or more subset routines. If a superset is included in common code, the label of the superset, as well as the labels of all its subsets, appears on the printer listing; however, only the superset appears in the common code.

J.2.2. Object Module of Equates

The object module of equates produced by the DMCC utility program is used by the Linkage Editor to satisfy external references (EXTRNs) in the user program's DTF address table. These DTF addresses are external references to labels defined in common code. The equates contain information allowing the OPEN transient routines to complete the linkage to common code.

The object module containing equates should be placed in the user's Reserve library; the module is then readily available to the Linkage Editor. Programs which use common code and the auto include feature of the Linkage Editor should not specify an auto include reserve library containing Data Management commond code modules. Common code modules should reside in a reserve library other than the library designated to be searched for automatic inclusion modules. The module is placed in the library by using the Librarian routine. The Reserve library may be kept on tape or disc. Figure 3—2 shows how data from the system library tape is merged with the object module containing equates.

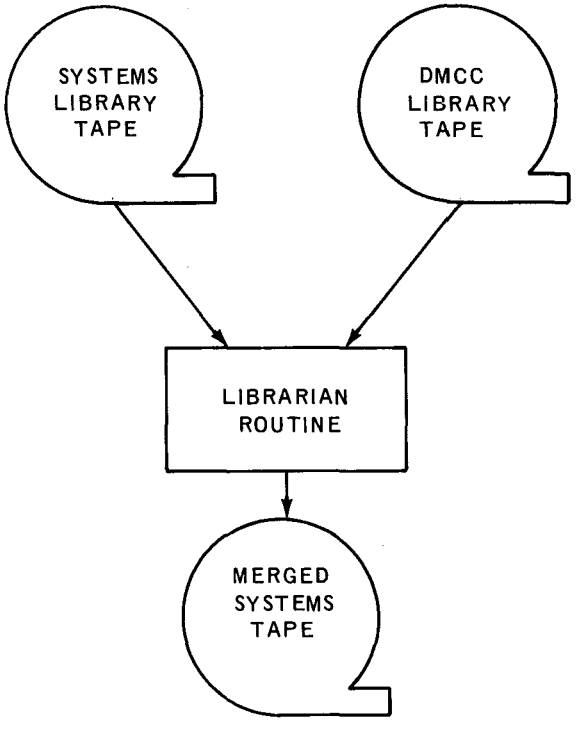
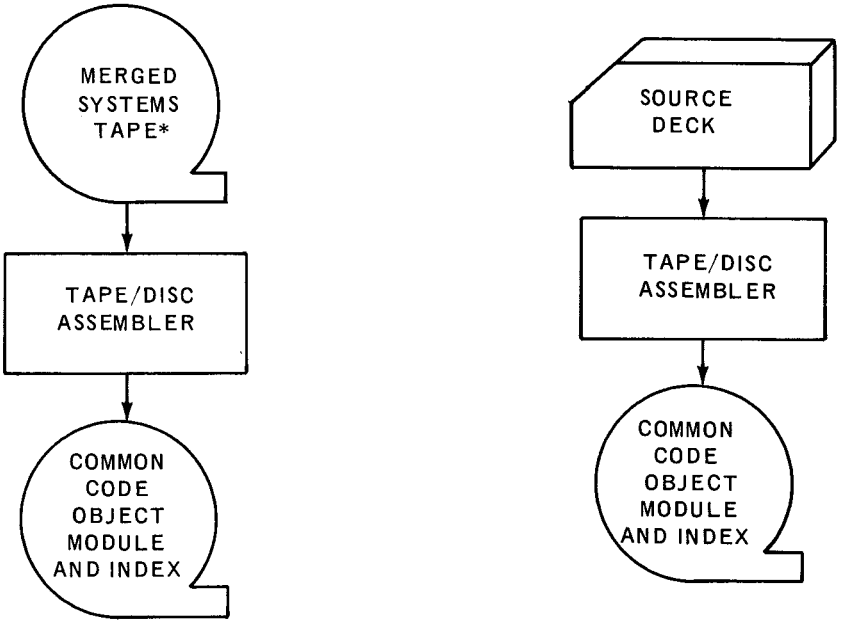


Figure J-2. Placing Object Module in Tape Reserve Library

For more explicit instructions, see *UNIVAC OS/4 Tape Librarian Programmer Reference, UP-7667* (current version) or *UNIVAC OS/4 Disc Librarian Programmer Reference, UP-7745* (current version).

J.2.2.1. Source Deck Assembly

The source module of the DMCC library tape or the source deck produced by the DMCC must be assembled to produce an object module containing the common code and its index. Since the common code itself is generated by a series of Proc calls, use of the tape assembler is not recommended, nor should a Proc tape be specified to the tape/disc assembler. Figure J–3 shows use of the tape/disc assembler.



*The DMCC library tape may also be used as tape input to the assembler.

Figure J–3. Assembly of Source Modules

J.2.2.2. Source Module Assembly Linkage

The object module produced by the assembler has several control sections. These control sections are linked separately to produce the common code module and indexes associated with the module.

- Common Code Module

  Common code must be linked separately as a load module. The module is then placed in the YCTRL file of the system disc (see J.2.4). The Supervisor loads the module into storage from the disc at IPL time. Figure J–4 shows the use of the tape Linkage Editor.

- Common Code Indexes

  The other control sections are indexes describing the displacement from the start of the common code load module of each label that could be defined in common code. The index entries are in fixed order. If the routine defining a certain label is not included in common code, its corresponding index entry is set to 0. These control sections are linked separately. The indexes are inserted into the transient routine area which is then placed in the YTRAN file of the system disc (see J.2.3).

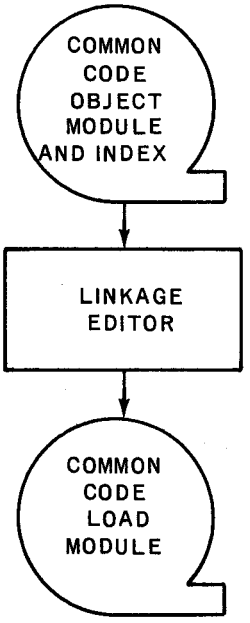


Figure J–4. Common Code Module Linkage

The Data Management OPEN transient routines access the index to complete the user program linkage to common code. Figure J–5 illustrates the linking of the common code index.

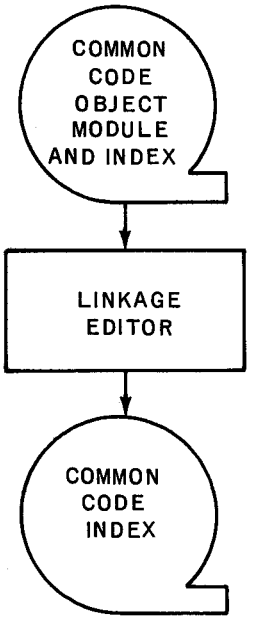


Figure J–5. Common Code Index Linkage

J.2.3. Setting Up the System Disc

To provide the Supervisor with the facility for handling Data Management common code, the following options must be included in the specifications to the SYSTEM macro instruction (see *UNIVAC 9400 System Supervisor Programmer Reference, UP-7689* (current version).

LABEL	♭ OPERATION ♭	OPERAND
	SYSTEM	{DISC / COMB} ,DMCC

The parameter DMCC generates the Supervisor facility for loading common code as part of the Supervisor.

After the Supervisor is loaded into storage and the DMCC parameter has been specified in the SYSTEM macro instruction, the following typeout occurs:

09 SV80 DMCC PKG?(A,B,....OR NONE)

The following response typein is required:

09R a

where a is the unique characteristic letter (see J.2.6) of the common code module to be loaded as part of the Supervisor. If a common code module is not to be loaded, the operator may type in NONE in place of a.

J.2.4. Placing Common Code Load Module and Index on System Disc

Before the Disc Mapping Program can place the common code module and its index on the disc system, they must be placed in the Load library of a library tape.

Figure J–6 shows the merged systems tape (see J.2.2) being merged with the common code index by the Tape Librarian. The common code index is in transient form. The common code load module and its index are both on the same OBJFIL tape as the Linkage Editor runs were stacked rather than placed on separate tapes.

The Disc Mapping Program (DACMAP) places the common code load module and index in the appropriate files on the system disc. When the user requests the Supervisor to load a common code module, the Supervisor searches the YCTRL file of the system disc for the module. When a program to be linked to common code is executed, the OPEN transient routine searches the YTRAN file of the system disc for the common code index.

Figure J–7 represents DACMAP placing the common code load module and the common code index in the$Y$CTRL and YTRAN files of the disc system, respectively.

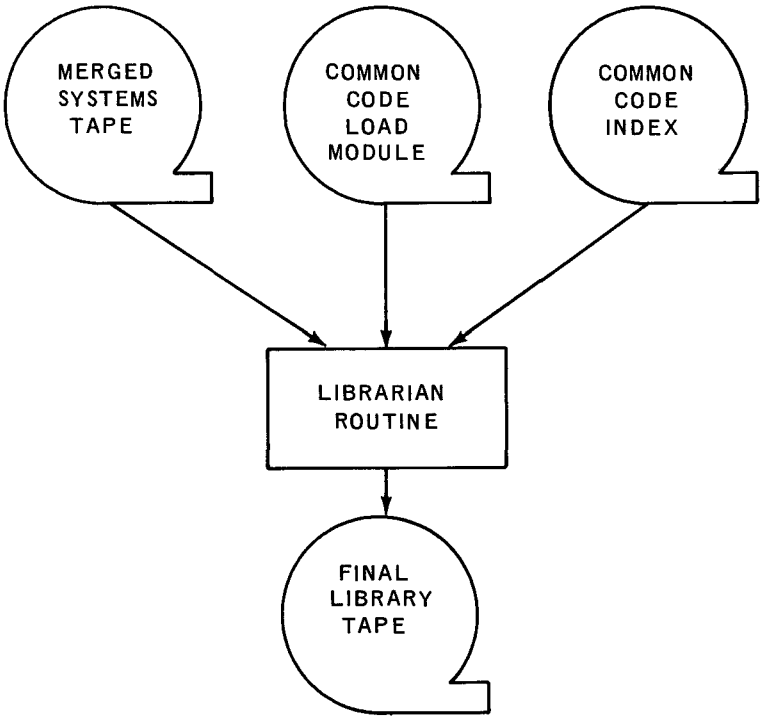


Figure J–6. Placing Common Code and Index on Library Tape

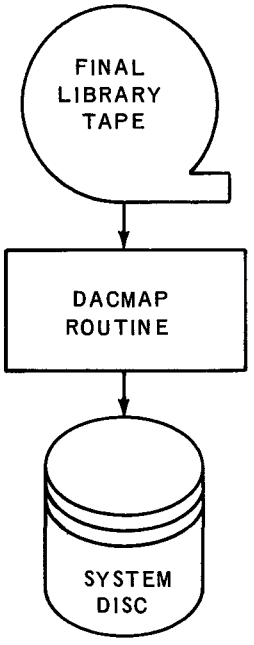


Figure J–7. Placing Common Code and Index on System Disc

J.2.5. Linking User Programs to Common Code

The user programs which are to be linked to common code must go through the Linkage Editor. The name of the object module of equates (see J.2.2), associated with the common code to which the program is linked, must be specified to the Linkage Editor by means of the following control statement:

INCLUDE object-module-name

The naming conventions for the object module are discussed in J.2.6. See J.2.10 for a sample control stream for linking a user program with the object module of equates.

The following error messages, which may be produced during the linking procedure, may be ignored:

Tape link message – K734
Disc link message – K834

J.2.6. Naming Conventions and Multiple Common Code Modules

It is possible to have more than one common code module in the YCTRL area of the systems disc. The user then has the option of selecting the common code module loaded at IPL time.

The various common code modules are distinguished by a unique characteristic letter. This characteristic letter must be supplied to DMCC by the following PARAM statement:

1	10
// PARAM	NAME␣a

POSITIONAL PARAMETER

a – a unique alphabetic character which is associated with the common code module, common code indexes, and the reserved library object module to be created.

This statement must be the first PARAM statement specified to DMCC. If it is omitted, DMCC assumes A as the characteristic letter.

The format of the name of the first control section of the source module (as well as the name of the source module itself), which in turn becomes the name of the common code module itself is:

DMAa00

where a is the unique characteristic letter of the common code module. (Default case: DMAA00.)

The format of the names of the other control sections of the source module which in turn become the names of the indexes associated with that common code module is:

YMAa0m

where a is the unique characteristic letter of the common code load module and m is the number of the index. (Default case of first index: YMAA00)

The format of the name of the object module is absolute equates used to link the user program to the common code module is:

D9$MAa00

where a is the unique characteristic letter of the common code module to which the user program is to be linked. (Default case: D9$MAA00.)

This is the object module name to be specified to the Linker Editor on an INCLUDE statement.

J.2.7. PARAM Statement For Selection of Routines

PARAM statements are used to select routines to be included in common code. The general format of the PARAM statement is:

1	10
// PARAM	codeword␣[option-1][,option-2]...

where codeword defines the Data Management file type being described and the options describe the characteristics of the file processing to be provided for in common code. The options for a particular file type parallel the keyword parameters of that file type's DTF macro instruction.

Options may be listed in any order and are separated by commas. Scanning of options is terminated by the first blank encountered.

Continuation statements are not recognized; that is, column 72 must be blank. If all options for a file cannot be contained on one PARAM statement, additional PARAM statements with the same codeword may be used.

J.2.7.1. Printer Modules

The following PARAM statement specifies which Data Management printer file options are to be provided in common code:

1	10
// PARAM	PR [{IOA1[(WORK)] / IOA2[(WORK)] / SUPER}][,CNTRL][,PRTOV]

CODEWORD:

PR – indicates that options refer to Data Management printer modules and automatically includes DP$CHK1 in common code.

OPTIONS:

IOA1 – includes DP$10E1 in common code. This module handles printer files having one I/O area and no work area.

IOA1(WORK) – includes DP$11E1 in common code. This module handles printer files having one I/O area and a work area.

IOA2 – includes DP$20E1 in common code. This module handles printer files having two I/O areas and no work area.

IOA2(WORK) – including DP$21E1 in common code. This module handles printer files having two I/O areas and a work area.

SUPER – includes DP$62E1 in common code. This module handles one or two I/O areas with or without a work area. This module should be included unless only one combination of I/O areas and work area will be used.

CNTRL – includes DP$CNT1 in common code. This module handles the CNTRL imperative macro instruction for printer files.

PRTOV – includes DP$PRT1 in common code. This module is needed when a printer overflow option is specified.

J.2.7.2. Reader Modules

The following PARAM statement includes the Data Management card reader module (DR$GET1) in common code:

1	10
// PARAM	READ

J.2.7.3. Punch Modules

The following PARAM statement specifies which Data Management punch file options are to be provided in common code:

1	10
// PARAM	PUNCH [{IOA1(WORK) / IOA2(WORK) / SUPER}][,CNTRL][,PUNR]

CODEWORD:

PUNCH – indicates that options refer to Data Management punch modules and automatically includes DW$CHK1 in common code.

OPTIONS:

IOA1 – indicates DW$10E1 in common code. This module handles punch files having one I/O area and no work area.

IOA1(WORK) – including DW$11E1 in common code. This module handles punch files having one I/O area and a work area.

IOA2 – includes DW$20E1 in common code. This module handles punch files having two I/O areas and no work area.

IOA2(WORK) – includes DW$21E1 in common code. This module handles punch files having two I/O areas and a work area.

SUPER – includes DW$62E1 in common code. This module handles one or two I/O areas with or without a work area. This module should be included unless only one combination of I/O areas and work area will be used.

CNTRL – includes DW$CNT1 in common code. This module handles the CNTRL imperative macro instruction for punch files.

PUNR – includes DW$RTY1 in common code. This module is needed when card punch error recovery has been specified.

J.2.7.4. Readpunch Modules

The following PARAM statement specifies which Data Management read/punch file options are to be included in common code:

1	10
// PARAM	READPUNCH [CNTRL]

CODEWORD:

READPUNCH – indicates that the option (if present) refers to the Data Management read/punch modules and automatically includes DQ$GET1, DQ$PUT1 and DW$CHK1 in common code.

OPTION:

CNTRL – includes DW$CNT1 in common code. This module handles the CNTRL imperative macro instruction for read/punch files.

NOTE: DW$CHK1 and DW$CNT1 are modules shared by punch and read/punch files. They appear in common code only once even if specified more than once.

J.2.7.5. Indexed Sequential Modules

The following PARAM statement specifies which Data Management indexed sequential modules are to be included in common code.

1	10
// PARAM	IS [LOAD][,ADD] [,{SEQNTL / RANDOM / RANSEQ}] [,INDAREA][,IOSIZE][,BLK][,UNB]

CODEWORD:

IS – indicates that the options refer to Data Management indexed sequential modules.

OPTIONS:

LOAD – includes DU$LOAD1 in common code. This module is used when creating or extending files.

ADD – includes DU$ADD1 and DU$WTF1 in common code. These modules are used when inserting records in a file.

SEQNTL – includes DU$RESL1, DU$RSEQ1, DU$PUT1, and DU$RGET1 in common code. These modules are used when sequentially retrieving records from the file.

RANDOM – includes DU$RRAN1, DU$RWRT1 and DU$WTF1 in common code. These modules are used when randomly retrieving records from the file.

RANSEQ – includes DU$RRAN1, DU$WRT1, DU$WTF1, DU$RESL1, DU$RSEQ1, DU$PUT1 and DU$RGET1 in common code. These modules will allow both sequential and random retrieval of records from a file.

INDAREA – includes DU$RXCI1 in common code. This module is used when the resident cylinder index option is used.

IOSIZE – includes DU$AMIO1 in common code. This module is used when the when the option of reading and writing an entire track at a time is specified.

BLK – includes DU$ABLK1 in common code. This module is used when inserting blocked records in a file.

UNB – includes DU$AUNB1 in common code. This module is used when inserting unblocked records in a file.

J.2.7.6. Direct Access Modules

The following PARAM statement specifies which Data Management direct access modules are to be included in common code:

1	10
// PARAM	DA [READ] [,WRITE] [,AFTER] [,RELATIVE[(IDLOC)] [,ABSOLUTE[(IDLOC)]]

CODEWORD:

DA – indicates that the options refer to Data Management direct access modules and automatically includes DD$CT1, DD$WT1 and DD$AC1 in common code (see the AFTER specification for DD$AC1 restriction).

OPTIONS:

READ – includes DD$RD1 in common code. This module handles the READ imperative macro instruction for direct access files.

WRITE – includes DD$WR1 in common code. This module handles update writes (that is, WRITE imperative macro instructions with either the ID or KEY positional parameter) for direct access files.

AFTER – includes DD$FW1 and DD$BC1 in common code. DD$FW1 handles format writes (that is, WRITE imperative macro instructions with either the RZERO or AFTER positional parameter) for direct access files. When AFTER is specified, DD$AC1 is not included in common code because it is a subset of DD$BC1.

CNTRL – includes DD$SK1 in common code. This module handles the CNTRL imperative macro instructions for direct access files.

RELATIVE – includes DD$RL1 in common code. This module is needed whenever relative addressing is used in direct access files.

RELATIVE(IDLOC) – includes DD$RL1 and DD$RW1 in common code. DD$RW1 is needed if relative addressing is used and the IDLOC option is requested.

ABSOLUTE – includes DD$ID1 in common code. DD$ID1 is needed if absolute addressing is used in direct access files.

ABSOLUTE(IDLOC) – includes DD$ID1 and DP$AW1 in common code. DD$AW1is needed if absolute addressing is used and the IDLOC option is requested.

J.2.7.7. Sequential Disc Modules

The following PARAM statement specifies which Data Management sequential disc file options are to be provided in common code:

1	10
// PARAM	SD [UPDATE[(WORK)]][,CNTRL] [,INPUT {(BLK[,WORK]) / (UNB[,WORK]) / (SUPER)}] [,OUTPUT {(BLK[,WORK]) / (UNB[,WORK]) / (SUPER)}]

CODEWORD:

SD – indicates that the options refer to Data Management sequential disc modules and automatically includes DS$ISS1 in common code unless UPDATE [(WORK)] is the only optional parameter.

OPTIONS:

UPDATE – includes DS$ISU1, DS$GTF1, DS$PTF1, DX$ISS1, DS$WCI1 and DS$RLU1 in common code. These modules are needed when sequential disc input files are to be read, modified and then written back into the same storage location.

UPDATE(WORK) – includes DS$ISU1, DS$GTF1, DS$PTF1, DS$WCI1, DS$RLU1 and DS$MVR1 in common code. This parameter is needed if sequential disc input files are to be updated and records in the input file are to be transferred from the input I/O area to a work area.

CNTRL – includes DS$CNT1 in common code. This module handles the CNTRL imperative macro instruction for sequential disc files.

INPUT(BLK) – includes DS$GTC1, DS$WCI1 and DS$RSE1 in common code. DS$GTC1 handles blocked sequential input files not using a work area. DS$WCI1 is needed with sequential disc input files and DS$RSE1 is needed with blocked sequential input files.

INPUT(BLK,WORK) – includes DS$GTD1, DS$WCI1, DS$RSE1 and DS$MVR1 in common code. DS$GTD1 handles blocked sequential input files using a work area. DS$WCI1 is needed with sequential disc input files and DS$RSE1 is needed with blocked sequential input files. DS$MVR1 is needed with sequential files using a work area.

INPUT(UNB) – includes DS$GTA1 and DS$WCI1 in common code. DS$GTA1 handles unblocked sequential input files, not using a work area. DS$WCI1 is needed for sequential disc input files.

INPUT(UNB,WORK) – includes DS$GTB1, DS$WCI1 and DS$MVR1 in common code. DS$GTB1 handles unblocked sequential input files using a work area. DS$WCI1 is needed with sequential disc input files and DS$MVR1 is needed for sequential files using a work area.

INPUT(SUPER) – includes DS$GTE1, DS$WCI1, DS$RSE1 and DS$MVR1 in common code. DS$GTE1 handles blocked or unblocked sequential input files, using or not using a work area. DS$WCI1 is needed with sequential disc input files and DS$RSE1 is needed with blocked sequential input files. DS$MVR1 is needed for sequential files using a work area.

OUTPUT(BLK) – includes DS$PTC1, DS$WCO1 and DS$TNC1 in common code. DS$PTC1 handles blocked sequential output files not using a work area. DS$WCO1 is needed with sequential disc output files and DS$TNC1 is needed for blocked sequential output files.

OUTPUT(BLK,WORK) – includes DS$PTD1, DS$WCO1, DS$TNC1 and DS$MVR1 in common code. DS$PTD1 handles blocked sequential output files using a work area. DS$WCO1 is needed with sequential disc output files and DS$TNC1 is needed with blocked sequential output files. DS$MVR1 is needed for sequential files using a work area.

OUTPUT(UNB) – includes DS$PTA1 and DS$WCO1 in common code. DS$PTA1 handles unblocked sequential output files, not using a work area. DS$WCO1 is needed with sequential disc output files.

OUTPUT(UNB,WORK) – includes DS$PTB1, DS$WCO1 and DS$MVR1 in common code. DS$PTB1 handles unblocked sequential output files using a work area. DS$WCO1 is needed with sequential disc output files and DS$MVR1 is needed for sequential files using a work area.

OUTPUT(SUPER) – includes DS$PTE1, DS$WCO1, DS$TNC1 and DS$MVR1 in common code. DS$PTE1 handles blocked or unblocked sequential output files, using or not using a work area. DS$WCO1 is needed with sequential disc output files and DS$TNC1 is needed with blocked sequential output files. DS$MVR1 is needed for sequential files using a work area.

J.2.7.8. Sequential Tape Modules

The following PARAM statement specifies which Data Management sequential tape file options are to be provided in common code.

1	10
// PARAM	MT [BACK][,INPUT{(BLK[,WORK]) / (UNB[,WORK]) / (SUPER)}][,OUTPUT{(BLK[,WORK]) / (UNB[,WORK]) / (SUPER)}]

CODEWORD:

MT – indicates that the options refer to Data Management sequential tape modules and automatically includes DS$ISS1 and DT$CTL1 in common code.

OPTIONS:

BACK – includes DT$GTG1, DT$WKB1 and DS$RSE1 in common code. DT$GTG1 handles blocked or unblocked sequential input files, using or not using a work area, whether they are being read forward or backward. DT$WKB1 is needed with sequential tape input files being read backward but will also handle these files if they are being read forward. DS$RSE1 is needed for blocked sequential input files.

INPUT(BLK) – includes DS$GTC1, DT$WKI1 and DS$RSE1 in common code. DS$GTC1 handles blocked sequential input files not using a work area. DT$WKI1 is needed with sequential tape input files and DS$RSE1 is needed for blocked sequential input files.

INPUT(BLK,WORK) – includes DS$GTD1, DT$WKI1, DS$RSE1 and DS$MVR1 in common code. DS$GTD1 handles blocked sequential input files using a work area. DT$WKI1 is needed with sequential tape input files and DS$RSE1 is needed with blocked sequential input files. DS$MVR1 is needed for sequential files using a work area.

INPUT(UNB) – includes DS$GTA1 and DT$WKI1 in common code. DS$GTA1 handles unblocked sequential input files, not using a work area. DT$WKI1 is needed for sequential tape input files.

INPUT(UNB,WORK) – includes DS$GTB1, DT$WKI1 and DS$MVR1 in common code. DS$GTB1 handles unblocked sequential input files using a work area. DT$WKI1 is needed with sequential tape input files and DS$MVR1 is needed for sequential files using a work area.

INPUT(SUPER) – includes DS$GTE1, DT$WKI1, DS$RSE1 and DS$MVR1 in common code. DS$GTE1 handles blocked or unblocked sequential input files, using or not using a work area. DT$WKI1 is needed with sequential tape input files and DS$RSE1 is needed with blocked sequential input files. DS$MVR1 is needed for sequential files using a work area.

OUTPUT(BLK) – includes DS$PTC1, DT$WKO1 and DS$TNC1 in common code. DS$PTC1 handles blocked sequential output files not using a work area. DT$WKO1 is needed with sequential tape output files and DS$TNC1 is needed for blocked sequential output files.

OUTPUT(BLK,WORK) – includes DS$PTD1, DT$WKO1, DS$TNC1 and DS$MVR1 in common code. DS$PTD1 handles blocked sequential output files using a work area. DT$WKO1 is needed sequential tape output files and DS$TNC1 is needed with blocked sequential output files. DS$MVR1 is needed for sequential files using a work area.

OUTPUT(UNB) – includes DS$PTA1 and DT$WKO1 in common code. DS$PTA1 handles unblocked sequential output files, not using a work area. DT$WKO1 is needed for sequential tape output files.

OUTPUT(UNB,WORK) – includes DS$PTB1, DT$WKO1 and DS$MVR1 in common code. DS$PTB1 handles unblocked sequential output files using a work area. DT$WKO1 is needed with sequential tape output files and DS$MVR1 is needed for sequential files using a work area.

OUTPUT(SUPER) – includes DS$PTE1, DT$WKO1, DS$TNC1 and DS$MVR1 in common code. DS$PTE1 handles blocked or unblocked sequential output files, using or not using a work area. DT$WKO1 is needed with sequential tape output files and DS$TNC1 is needed with blocked sequential output files and DS$TNC1 is needed with blocked sequential output files. DS$MVR1 is needed for sequential files using a work area.

J.2.7.9. Shared Sequential Modules

Sequential disc and tape Data Management share many modules. Even if certain shared modules are included by the MT and the SD positional parameters, these modules are included only once in common code. As mentioned in J.2.1, some Data Management modules include the function of other modules. This is true of many of the shared modules. After all PARAM statements have been interpreted, DMCC determines whether more than one subset module has been requested. If so, the superset module is placed in common code instead of the subset modules.

Tables J–1 and J–2 show modules generated by INPUT, OUTPUT, and BACK optional parameters.

CODEWORD	OPTION	DS$GTA1 (1) (2)	DS$GTB1 (1) (2)	DS$GTC1 (1) (2)	DS$GTD1 (1) (2)	DS$GTE1 (2)	DT$GTG1	DS$MVR1	DS$RSE1	DS$WCI1	DT$WKI1 (3)	DT$WKB1
SD	INPUT(BLK)			X					X	X		
	INPUT(BLK,WORK)				X			X	X	X		
	INPUT(UNB)	X								X		
	INPUT(UNB,WORK)		X					X		X		
	INPUT(SUPER)					X		X	X	X		
MT	INPUT(BLK			X					X		X	
	INPUT(BLK,WORK)				X			X	X		X	
	INPUT(UNB)	X									X	
	INPUT(UNB,WORK)		X					X			X	
	INPUT(SUPER)					X		X	X		X	
	BACK						X		X			X

NOTES: (1) Subsets of DS$GTE1

(2) Subsets of DT$GTG1

(3) Subset of DT$WKB1

Table J–1. Modules Included by Input Options of SD and MT Codewords and BACK Option of MT Codeword

CODEWORD	OPTION	DS$PTA1*	DS$PTB1*	DS$PTC1*	DS$PTD1*	DS$PTE1	DS$MVR1	DS$TNC1	DS$WCO1	DT$WKO1
SD	OUTPUT(BLK)			X				X	X	
	OUTPUT(BLK,WORK)				X		X	X	X	
	OUTPUT(UNB)	X							X	
	OUTPUT(UNB,WORK)		X				X		X	
	OUTPUT(SUPER)					X	X	X	X	
MT	OUTPUT(BLK)			X				X		X
	OUTPUT(BLK,WORK)				X		X	X		X
	OUTPUT(UNB)	X								X
	OUTPUT(UNB,WORK)		X				X			X
	OUTPUT(SUPER)					X	X	X		X

*Subsets of DS$PTE1

Table J–2. Modules Included by Output Options of SD and MT Codewords

J.2.7.10. Summary of PARAM Statements

Table J–3 summarizes the PARAM statements recognized by the DMCC.

PREFIX	CODEWORD	OPTIONS
// PARAM	NAME	a
// PARAM	PR	[{IOA1 [(WORK)] / IOA2 [(WORK)] / SUPER}] [,CNTRL] [,PRTOV]
// PARAM	READ	
// PARAM	PUNCH	[{IOA1(WORK) / IOA2(WORK) / SUPER}] [,CNTRL] [,PUNR]
// PARAM	READPUNCH	[CNTRL]
// PARAM	IS	[LOAD] [,ADD] [,{SEQNTL / RANDOM / RANSEQ}] [,INDAREA] [,IOSIZE] [,BLK] [,UNB]
// PARAM	DA	[READ] [,WRITE] [,AFTER] [,CNTRL] [,RELATIVE [(IDLOC)]] [,ABSOLUTE [(IDLOC)]]
// PARAM	SD	[UPDATE [(WORK)]] [,CNTRL] [,INPUT {(BLK [,WORK]) / (UNB [,WORK]) / (SUPER)}] [,OUTPUT {(BLK [,WORK]) / (UNB [,WORK]) / (SUPER)}]
// PARAM	MT	[BACK] [,INPUT {(BLK [,WORK]) / (UNB [,WORK]) / (SUPER)}] [,OUTPUT {(BLK [,WORK]) / (UNB [,WORK]) / (SUPER)}]

Table J–3. Summary of PARAM Statements

J.2.8. DMCC Routine Control Stream Requirements

A sample control stream for the DMCC follows:

```
LABEL      OPERATION    OPERAND
1          10           16
//JOB      DMCCRUN      Job Step 1: Execute DMCC
//OPTION   NOVOL
//DVC      2            These cards may be omitted if punched card
//LFD      PUNCH        output is not wanted.
//DVC      3
//LFD      PRNTR
//DVC      5
//VOL      ABC345
//LFD      LIBOUT
//EXEC     DMCC,LOAD$LIB,,REL
//PARAM    NAME D
//PARAM
.
.
.
.
/&                      End of DMCC job run.
```

This sample control stream is the first job step for the control stream examples given in J.2.9.and J.2.10.

ABC345 is the volume serial number of an output tape, and D is the unique characteristic letter of the common code module to be generated. The NAME statement must be the first PARAM statement. If it is omitted, A is assumed as the characteristic letter.

If the punch is not specified, the following message appears on the console

DM43 UUU PUNCH

Ignore the message and allow the DMCC routine to terminate normally.

J.2.9. Control Streams for Setting Up Common Code Disc System

Two sample control streams for setting up a common code disc system are presented as follows:

- The following sample control stream merges a systems tape and the DMCC library tape and assembles and links the common code module and the index transients created by DMCC, placing them on the disc in the appropriate libraries.

  Refer to Table J–4 for a description of the mounting scheme.

```
LABEL          OPERATION      OPERAND
1              10             16
// JOB         SETUP
// OPTION      NOVOL
// DVC         3
// LFD         PRNTR
// DVC         5
// VOL         ABC345
// LFD         ALTLIB
// DVC         6
// VOL         ABC246
                                        } Systems tape. The object module and source
                                          module are placed in the Reserve library and
                                          the Source library respectively.
// LFD         LIBIN
// DVC         7
// VOL         ABC537
// LFD         LIBOUT
// EXEC        LIBS,LOAD$LIB,,REL
/$
   LIB       IPL,NOBJ
   ADDR      D9$MADOO(ALT)
   ADD$      $D$MADOO(ALT)
/*
```

Job Step 2: Build system library tape

Library output tape of DMCC.

This module should be kept in the Reserve library of a library tape for use when linking.

Job Step 3: Assemble source modules

```
// OPR       DISMOUNT TAPE ABC246
// OPR       REPLACE WITH TAPE ASM26
// OPR       *MOUNT TAPE ABC468 ON DVC 8
// OPTION    NOVOL
// DVC       3
// LFD       PRNTR
// DVC       5
// VOL       ABC345
// LFD       SCR1
// DVC       6
// VOL       ASM26
                                        } Since volume ABC537 is a copy of volume
                                          ABC246, this tape need not be changed if
                                          the user wishes to reuse volume ABC246
                                          as a scratch tape. It is recommended to
                                          save it and mount another tape. Volume
                                          ASM26 is used as a scratch tape.
// LFD       SCR2
```

The DMCC library tape is no longer needed as it was merged onto volume ABC537.

LABEL	OPERATION	OPERAND	
// DVC	7		This tape is specified only if input to DTASM is from tape.
// VOL	ABC537		
// LFD	SYSRES		
// DVC	8		
// VOL	ABC468		
// LFD	OBJFIL		
// DVC	21		
// VOL	ABC021		
// DVC	22		
// VOL	ABC022		
// LFD	SYSPOOL		
// DVC	21		The control stream assumes that the Proc library is on disc. It is recommended that a disc Proc library be used. Use of a tape Proc library is time consuming.
// VOL	ABC021		
// LBL	PROC$LIB,ABC026		
// LFD	PROC$		
// EXEC	DTASM,LOAD$LIB,,REL		Use of a tape assembler would be inefficient.
// PARAM	IN=DMAD00/SYSRES		If user is assembling source deck, the deck should be inserted here preceded by a /$ card and succeeded by a /* card. The PARAM statement is then omitted.

```
LABEL      OPERATION      OPERAND
1          10             16
//  DVC    3              Job Step 4: Link object modules
//  LFD    PRNTR
//  DVC    5
//  VOL    ABC345
//  LFD    SCR1
//  DVC    6
//  VOL    ASM26
//  LFD    LDMFIL
//  DVC    7
//  VOL    ABC537
//  LFD    SYSRES
//  DVC    8
//  VOL    ABC468
//  LFD    OBJFIL
//  EXEC   LINK,LOAD$LIB,,REL     The tape Linkage Editor
/$                                has been used here.
   LOADM  $D$MAD00,X'00'          These LOADM, INCLUDE
   INCLUDE $D$MAD00($D$MAD00),*   and ENTER statements
   ENTER  $D$MAD00                link the common code
/*                                module itself.
```

```
//  OPTION NOVOL                  Job Step 5: Link common code index
//  EXEC   LINK,LOAD$LIB,,REL
/$
   LOADM  $Y$MAD00,X'00'          The LOADM, INCLUDE,
   INCLUDE $D$MAD00($Y$MAD00),*   and ENTER statements
   ENTER  $Y$MAD00                link the common code
/*                                index.
```

```
LABEL        OPERATION        OPERAND
1            10               16
//  OPTION   NOVOL
//  DVC      3
//  LFD      PRNTR
//  DVC      5
//  VOL      ABC345
//  LFD      LIBOUT
//  DVC      6
//  VOL      ASM26
//  LFD      OBJFIL
//  DVC      7
//  VOL      ABC537
//  LFD      LIBIN
//  EXEC     LIBS,LOAD$LIB,,REL
```

Job Step 6: Common code and index to tape

Volume ABC345 becomes the final library tape.

The common code module and its index must be placed on a library tape before mapping them to the the disc.

```
/$
   LIB      IPL,NALT
   CPYL     TO,$J$CONOO
   ADDL     $D$MADOO
   ADDL     TR,$Y$MADOO
/*
```

LABEL 1	OPERATION 10	OPERAND 16	

```
// OPTION   NOVOL                           Job Step 7:  Common code and index on disc
// DVC      3
// LFD      PRNTR
// DVC      5
// VOL      ABC345
// LFD      LIBIN
// DVC      20
// VOL      ABC020
// LFD      $YSMAP,SQ,1
// DVC      20
// VOL      ABC020
// LBL      $Y$TRAN,ABC020
// LFD      SYSTRAN
// DVC      20
// VOL      ABC020
// LBL      $Y$CTRL,ABC020
// LFD      SYSCTRL
// EXEC     DACMAP00,LOAD$LIB,,REL          DACMAP places the
                                            common code module
                                            and its index in the
                                            appropriate libraries
                                            on the disc.

// PARAM    NCON,LIST
/$
   MAP      CTRL,MOD
   ADD      $D$MAD00
   MAP      TRAN,MOD
   ADD      $Y$MAD00
/*
/&
```

JOB STEPS	DVC 5	DVC 6	DVC 7	DVC 8	DVC 20	DVC 21	DVC 22
1. DMCC	VOL ABC345 LIBOUT						
2. LIBS	ABC345 ALTLIB	VOL ABC246 LIBIN	VOL ABC537 LIBOUT				
3. DTASM	SCR1	VOL ASM26 SCR2	SYSRES	VOL ABC468 OBJFIL		VOL ABC021 SYSPOOL PROC$	VOL ABC022 SYSPOOL
4. (TAPE) LINK	SCR1	LDMFIL	SYSRES	OBJFIL			
5. (TAPE) LINK	SCR1	LDMFIL	SYSRES	OBJFIL			
6. LIBS	LIBOUT	OBJFIL	LIBIN				
7. DACMAP	LIBIN				VOL ABC020 SYSMAP SYSCNTRL SYSTRAN		

Table J–4. Control Stream Mounting Scheme

- The following sample control stream utilizes the disc assembler and disc linker for the common code module and the index transients created by DMCC when the procs and reserve modules reside in a disc library. DASM4 is used here for the UNIVAC OS/4 program utilization with the UNIVAC 9700 System. DASM4 is replaced by DASM for the UNIVAC 9400 System.

```
LABEL        OPERATION    OPERAND                          
1            10           16
// JOB       DMCCRUN,,,,AOOO,COOO      Job step 1: Execute DMCC
// OPTION    NOVOL,MAYIDUMP
// DVC       20,SYM
// LFD       PRNTR
// DVC       10,H                      These cards may be omitted if
// LFD       PUNCH                     punched card output is not
                                       desired.
// DVC       40                        AAAAAA is the volume serial
// VOL       AAAAAA                    number of an output tape.
// LFD       LIBOUT                    The user may mount a scratch
                                       tape.
// EXEC      DMCC,LOAD$LIB,,REL
// PARAM     NAME Z                    Z is the unique characteristic
                                       letter of the common code
                                       module to be generated.
// PARAM
  .                                    Insert desired parameter cards
  .                                    here.
  .
// OPTION    NOVOL,MAYIDUMP            Job step 2: Assemble source.
// DVC       20,SYM
// LFD       PRNTR
// DVC       40
// VOL       AAAAAA
// LFD       ALT1
// DVC       70
// VOL       SYSCR1                    SYSCR1 is a scratch disc con-
                                       taining a SYSPOOL area.
// DVC       71
// VOL       OS4LBS                    OS4LBS contains the standard
                                       disc proc library and a SYSPOOL
                                       area.
// LFD       SYSPOOL
// DVC       71
// VOL       OS4LBS
// LBL       PROC$LIB,OS4LBS
// LFD       PROC$
```

```
LABEL      OPERATION      OPERAND
1          10             16
// EXEC     DASM4,LOAD$LIB,,REL
// PARAM    IN=$D$MAZ00/ALT1
```

If assembling from source deck, deck is placed here, preceded by a /$ card and succeeded by a /* card. The PARAM statement is then omitted.

```
// OPTION   NOVOL,MAYIDUMP
```

Job step 3: Link common code module

```
// DVC      20,SYM
// LFD      PRNTR
// DVC      41
// VOL      BBBBBB
```

BBBBBB is the volume serial number of an output tape. The user may mount a scratch tape.

```
// LFD      LDMFIL
// DVC      70
// VOL      SYSCRI
// DVC      71
// VOL      OS4LBS
// LFD      SYSPOOL
// EXEC     DLINK,LOAD$LIB,,REL

// PARAM    OUT=T
/$
   LOADM    $D$MAZ
   INCLUDE  $D$MAZ00($D$MAZ00),*
   ENTER    $D$MAZ00
/*
```

```
// OPTION   NOVOL,MAYIDUMP
```

Job step 4: Link common code index

```
// EXEC     DLINK,LOAD$LIB,,REL
// PARAM    OUT=T
/$
   LOADM    $Y$MAZ
   INCLUDE  $D$MAZ00($Y$MAZ00),*
   ENTER    $Y$MAZ00
```

```
LABEL      OPERATION      OPERAND
1          10             16
// OPTION  NOVOL,MAYIDUMP
// DVC     20,SYM
// LFD     PRNTR
// DVC     40
// VOL     AAAAAA
// LFD     ALTLIB
// DVC     41
// VOL     BBBBBB
// LFD     OBJFIL
// DVC     42
// VOL     CCCCCC
// LFD     LIBIN
// DVC     43
// VOL     DDDDDD
// LFD     LIBOUT
// EXEC    LIBS,LOAD$LIB,,REL
/$
  LIB     IPL
  ADDL    $D$MAZOO
  ADDL    TR,$Y$MAZOO
  ADDR    D9$MAZOO(ALT)
/*
```

Job step 5: Merge common code on system tape

CCCCCC is the volume serial number of a tape containing a copy of OS/4 in standard tape library format.

DDDDDD is the volume serial number of a tape containing the merged system tape and data management common code.

LABEL	OPERATION	OPERAND	
// OPTION	NOVOL,MAYIDUMP		Job step 6: DACMAP common code module and index on disc
// DVC	20,SYM		
// LFD	PRNTR		
// DVC	43		
// VOL	DDDDDD		
// LFD	LIBIN		
// DVC	72		
// VOL	XXXXXX		XXXXXX is the volume serial number of the systems disc onto which common code is to be mapped.
// LFD	SYSMAP		
// DVC	72		
// VOL	XXXXXX		
// LBL	YTRAN,XXXXXX		
// DVC	72		
// VOL	XXXXXX		
// LBL	YCTRL,XXXXXX		
// LFD	SYSCTRL		
// EXEC	DACMAP00,LOAD$LIB,,REL		
// PARAM	NCON,LIST		
/$			
MAP	CTRL,MOD		
ADD	DMAZOO		
MAP	TRAN,MOD		
ADD	YMAZOO		
/*			
/&			

J.2.10. Control Stream for Linking User Program to Common Code

Two sample control streams for linking the user program to common code are presented as follows:

- The following is a sample control steam for linking a user program to the common code module produced in J.2.8.

```
LABEL      OPERATION      OPERAND
1          10             16
// JOB      LINKUP
//
// OPTION   NOVOL
// DVC      3
// LFD      PRNTR
// DVC      5                          Volume ABC345 is the final
// VOL      ABC345                     library output tape from the
// LFD      SYSRES                     first control stream in J.2.9. It
                                       has the object module, D9$MAD00,
                                       in the Reserve library.
// DVC      6
// VOL      ABC123                     Output tape for user's linked program.
// LFD      LDMFIL
// DVC      7                          Assumes user's object module is on
// VOL      DEF789                     an assembler OBJFIL tape.
// LFD      OBJFIL
// DVC      8
// LFD      SCR1
// EXEC     LINK,LOAD$LIB,1,REL
/$
   LOADM    name[,addr]
   INCLUDE  [module name][(s1,s2,...,sn)][,filename]
   INCLUDE  D9$MAD00                   This INCLUDE statement is required
                                       when linking a program to common
                                       code.
   ENTER    expression                 Refer to UNIVAC OS/4 Linkage Editor
                                       Programmer Reference, UP-7703 (current
                                       version) for instruction on completing
                                       other control statements.
/*
/&
```

- The following sample job control stream illustrates the linking of a user test program named DMTEST with common code modules created in J.2.9.

```
LABEL      OPERATION      OPERAND
1          10             16
//  JOB        LKTEST
//  OPTION     NOVOL,MAYIDUMP        Job step 1: Build a temporary
                                     user reserve library
//  DVC        20,SYM
//  LFD        PRNTR
//  DVC        40
//  VOL        DDDDDD                DDDDDD is the volume serial
                                     number of a tape which is the
                                     output of the fifth job step of
                                     the preceding sample job stream.
//  LFD        ALT1
//  DVC        41
//  VOL        EEEEEE                EEEEEE is the volume serial
                                     number of a tape which contains
                                     the assembled output of the
                                     user's text program, DMTEST,
                                     in the reserve library.
//  LFD        ALT2
//  DVC        70
//  VOL        OS4LBS                OS4LBS is the volume serial
                                     number of a disc containing
                                     the standard system reserve
                                     library (RESV$LIB) and file
                                     in which a temporary reserve
                                     library may be built (USERLIB).
//  LBL        USERLIB,OS4LBS
//  LFD        LIB1
//  DVC        70
//  VOL        OS4LBS
//  LBL        RESV$LIB,OS4LBS
//  LFD        LIB2
//  EXEC       LIBUPS,LOAD$LIB,,REL
//  PARAM      LIN=(2,NMCL,NLIB)
```

```
LABEL      OPERATION      OPERAND
1          10             16
/$
           FILR     LIB1,0,2
           INPR     LIB2
           CPYR
           INPR     LIB1
           ADDR     DMTEST(ALT2)
           ADDR     D9$MAZOO(ALT1)
/*
// OPTION  NOVOL,MAYIDUMP
```

Job step 2: Link test program with common code.

```
// DVC     20,SYM
// LFD     PRNTR
// DVC     42
// VOL     FFFFFF
```

FFFFFF is the volume serial number of the output tape onto which the user's program is linked.

```
// LFD     LDMFIL
// DVC     70
// VOL     OS4LBS
```

OS4LBS is assumed to have a SYSPOOL area.

```
// DVC     71
// VOL     SYSCR1
```

SYSCR1 is a scratch disc containing a SYSPOOL area.

```
// LFD     SYSPOOL
// DVC     70
// VOL     OS4LBS
// LBL     USERLIB
// LFD     RESV$
// EXEC    DLINK,LOAD$LIB,,,REL
// PARAM   OUT=T
/$
           LOADM    DMTEST
           INCLUDE  D9$MAZOO
           INCLUDE  DMTEST
           ENTER    DMTEST
/*
/&
```

J.3. ERROR MESSAGES

■ DM43 UUU PUNCH

Explanation:

This informational message appears on the console if the user chooses not to produce punched card output.

Action:

None required. Processing continues.

■ ERROR – INVALID CARD IN JOB STREAM – SCAN TERMINATED HERE

Explanation:

An invalid statement has been encountered while scanning PARAM statements. The scan ends at this point; however, DMCC processing continues.

Action:

Correct the statement in error and resubmit the job.

■ ERROR – invalid parameter

Explanation:

An invalid parameter has been encountered while interpreting a PARAM statement. DMCC discontinues scanning the statement and prints out the statement starting with the erroneous parameter.

Action:

Correct the statement in error and resubmit the job.

■ ERROR – PARAM MISSPELLED

Explanation:

A statement begins with two slashes, but the slashes are not followed by ␣PARAM␣. The statement is ignored and the next statement is read.

■ DM46 uuu filename

Explanation:

When using common code, the OPEN transient routines make this typeout if an error is encountered as the user program is being linked to common code.

uuu represents the logical unit number of the device; filename is the label of the user's DTF macro instruction.

The error conditions causing this typeout are:

- A DTF macro instruction slot contains a label which must be defined by common code and no common code was loaded at IPL time.
- The bit masks of the common code module and the common code index do not match. The bit mask is a 32-byte area whose setting represents the modules generated in common code. This error condition would occur when more than one common code has been generated and the characteristic letters used were not unique.
- A DTF macro instruction contains a label which must be defined by common code and the common code loaded at IPL time does not define this label.
- The OPEN transient routine is unable to retrieve the index. This may be caused by an I/O error or by a missing index.

Action:

Resubmit the job after correcting the error.

INDEX

R

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

CUT

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT