HONEYWELL

LEVEL 66 UNIFIED FILE ACCESS SYSTEM (UFAS)

SOFTWARE

SERIES 60 (LEVEL 66)/6000

UNIFIED FILE ACCESS SYSTEM (UFAS) ADDENDUM A

SUBJECT

Additions and Changes to the Manual

SPECIAL INSTRUCTIONS

This is the first addendum to DC89-03 dated August 1979. Change bars in the margins indicate technical changes or new material; asterisks denote deletions. Sections XI and XII are completely new; therefore change bars have not been used. Insert the attached pages into the manual according to the collating instructions on the back of this cover.

Note:

Insert this cover after the manual cover to indicate the updating of the document with Addendum A.

SOFTWARE SUPPORTED

Series 60 Level 66 Software Releases 4S3 and DPS1.3
Series 6000 Software Release JS3

ORDER NUMBER

DC89-03A

July 1980

30561
1.5281
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update this manual, remove old pages and insert new pages as follows:

Remove	Insert
iii thru viii	iii thru viii
1-5, 1-6	1-5, 1-6
1-9 thru 1-12	1-9 thru 1-12
1-17 thru 1-20	1-17 thru 1-20.1, blank
1-23 thru 1-26	1-23 thru 1-26
1-29 thru 1-32	1-29, 1-30
	1-30.1, blank
	1-31, blank
	1-31.1, 1-32
	1-32.1, blank
1-37, 1-38	1-37, 1-37.1
	1-37.2, 1-38
1-45 thru 1-52	1-45 thru 1-52
2-1 thru 2-6	2-1 thru 2-6
2-11, 2-12	2-11, 2-12
2-17, 2-18	2-17, 2-18
2-25, 2-26	2-25, blank
	2-25.1, 2-26
2-29, 2-30	2-29, blank
	2-29.1, 2-30
2-39, 2-40	2-39, 2-40
3-23, 3-24	3-23, 3-24
4-13 thru 4-16	4-13 thru 4-16
5-11, 5-12	5-11, blank
	5-11.1, 5-12
5-17, 5-18	5-17, 5-18
5-31, 5-32	5-31, 5-32
6-23 thru 6-26	6-23 thru 6-26
7-1, blank	7-1, 7-2
	7-3, blank
8-1, 8-2	8-1, 8-2
9-3 thru 9-5, blank	9-3 thru 9-5, blank
10-3 thru 10-12	10-3, 10-4
	10-5, blank
	10-5.1, 10-6
	10-7, blank
	10-7.1, 10-8
	10-9 thru 10-12
10-15 thru 10-18	10-15, 10-16
	10-16.1, blank
	10-17, blank
	10-17.1, 10-18
10-21, 10-22	10-21, 10-22
10-25, 10-26	10-24.1, blank
	10-25, 10-26
	10-26.1, blank
	11-1 thru 11-10
	12-1 thru 12-8
	12-9, blank
A-1 thru A-8	A-1 thru A-4
	A-4.1, blank
	A-5, blank
	A-5.1, A-6
	A-7, A-8
B-5 thru B-8	B-5, B-6
	B-6.1, blank
	B-7, blank
	B-7.1, B-8
E-1, E-2	E-1, E-2
	E-2.1, blank
i-1 thru i-8	i-1 thru i-8
	i-9, blank

The information and specifications in this document are subject to change without notice. This document contains information about Honeywell products or services that may not be available outside the United States. Consult your Honeywell Marketing Representative.

SERIES 60 (LEVEL 66)/6000

UNIFIED FILE ACCESS SYSTEM (UFAS)

SUBJECT

Revision to the Unified File Acess System (UFAS) Reference Manual

SPECIAL INSTRUCTIONS

This revision of DC89 supersedes Revision 2, dated April 1978. Because of the extensive changes in both format and technical content, change bars and asterisks are not used to indicate new, revised and deleted material.

SOFTWARE SUPPORTED

Series 60 Level 66 Software Release 4S2 and DPS1.2
Series 6000 Software Release JS2

ORDER NUMBER

DC89-03

August 1979

Honeywell

PREFACE

The Unified File Access System (UFAS) provides the necessary compatibility and versatility to access different file organizations and to process varied file formats. Label processing and peripheral unit switching capabilities are also provided. The system flexibility enables the user to make use of the whole system or just those portions of the system required for a particular file format and efficient file utilization.

The Series 60 Level 66 is hereafter referred to as the Series 60.

The information and specifications in this document are subject to change without notice. This document contains information about Honeywell products or services that may not be available outside the United States. Consult your Honeywell Marketing Representative.

File No.: 1753, 1P53

This Page Intentionally Left Blank.

This Page Intentionally Left Blank.

CONTENTS

Page

Section I Introduction 1-1
File Organizations 1-2
Sequential 1-2
Relative 1-2
Indexed 1-2
Integrated 1-2
Access Modes 1-2
Sequential 1-2
Random 1-3
Dynamic 1-3
Processing Modes 1-3
Input 1-3
Output 1-3
Input/Output 1-3
Extend 1-3
User Label Processing 1-4
File Formats 1-5
Unified File Formats 1-5
Data Region 1-7
Inventory Region 1-17
ISP File Format 1-18
UFF And American National Standard Tape File Format 1-20
Physical Attributes 1-20
General Format 1-20
Data Format 1-20.1
Label Formats And Processing 1-22
Multi-Volume Output Processing 1-31.1
Multi-Volume Input Processing 1-31.1
Unlabeled UFF Tape Files 1-32
Unlabeled UFF File Processing 1-32
Single File Input Processing 1-32.1
Multi-File Input Processing 1-32.1
Output Processing 1-33
Extend Processing 1-33
System Input And Output Files 1-33
Physical Attributes 1-33
GFRC File Format 1-34
Physical Attributes (Tape Files) . . . 1-35
General Format (Labeled Tape Files) . 1-35
Data Format 1-35
Page Printing System (PPS) Format And Processing 1-37.1
Control Word Format 1-37.1
Label Formats And Processing 1-38
Partial Tape Label (PTL) 1-43
Unlabeled GFRC Tape Files 1-44
Unlabeled GFRC File Processing 1-45
Single File Input Processing 1-45
Multi-File Input Processing 1-45
Output Processing 1-45
UNLT Tape File Format 1-45
Physical Attributes 1-46
Unlabeled UNLT File Processing 1-46
Single File Input Processing 1-46
IBM Tape File Format 1-46

Page

Physical Attributes 1-47
General Format - Labeled Tapes 1-47
General Format - Unlabeled Tapes . . . 1-47
Data Format 1-48
Label Formats And Processing 1-48
UFAS Unlabeled IBM Tape Processing 1-49
File Formats 1-50
Unlabeled File Processing Procedure . . . 1-50
Single-File Input Processing 1-50
Multi-file Input Processing 1-50
Output Processing 1-50
Unlabeled File Data Format 1-51
UFAS Open Procedures 1-51
Single-File Input Open Procedure . . . 1-51
Multi-file Input Open Procedure . . . 1-51
Output Open Procedure 1-52
Extend Open Procedure 1-52
H2000 Tape File Format 1-52
H2000/60 Magnetic Tape Interchange Facility 1-52
Physical Attributes 1-52
General Format (Labeled Tape Files) . 1-53
Data Format 1-53
Label Formats And Processing 1-53
Unlabeled Tape Files 1-58
Processing Functions 1-58
File Protection And Concurrent Access Control 1-61
Buffer Pooling 1-62
Transliteration Tables 1-62

Section II UFF, ANS, GFRC, And IBM Sequential File Organization 2-1
File Information Block Macro 2-3
Functional Macros 2-15
Open Macro - DOPEN 2-16
Routine Package SYMREF Name 2-17
Put Macro - DLPUT 2-24
Rewrite Macro - DLRWR 2-27
Get Next Macro - DLGXT 2-29
Position Macro - DLPOS 2-31
Read User Labels - DLRLB 2-34
Write User Labels - DLWLB 2-35
Return Macro - UFSRET 2-37
Information Macro - DINFO 2-38
Checkpoint Files Macro - DCKPF 2-40
Checkpoint Files And Program Macro - DCKPT 2-41
Rollback Files Macro - DROLF 2-42
Rollback Files And Program Macro - DROLP 2-43
Wrap-up Macro - DWRAP 2-43
Close Macro - DCLOS 2-44

Section III Series 2000 Sequential File Organization . . . 3-1
2000/60 Magnetic Tape Interchange Facility 3-1
File Information Block Macro 3-2
Functional Macros 3-7
Open Macro - DOPEN 3-8
Routine Package SYMREF Name 3-10
Put Macro - DLPUT 3-15
Get Next Macro - DLGXT 3-17
Return Macro - UFSRET 3-19

CONTENTS (cont)

Page

Wrap-up Macro - DWRAP 3-20
Information Macro - DINFO 3-21
Close Macro - DCLOS 3-23

Section IV Relative File Organization 4-1
File Information Block Macro 4-3
Functional Macros 4-9
Open Macro - DOPEN 4-10
Routine Package SYMREF Name 4-12
Put Macro - DLPUT 4-16
Rewrite Macro - DLRWR 4-18
Delete Macro - DLDEL 4-20
Position Macro - DLPOS 4-22
Get Next Macro - DLGXT 4-27
Get Macro - DLGET 4-29
Return Macro - UFSRET 4-31
Information Macro - DINFO 4-32
Checkpoint Files Macro - DCKPF 4-35
Checkpoint Files And Program Macro - DCKPT 4-36
Rollback Files Macro - DROLF 4-37
Rollback Files And Program Macro - DROLP 4-38
Wrap-up Macro - DWRAP 4-39
Close Macro - DCLOS 4-40

Section V Indexed File Organization 5-1
File Information Block Macro 5-3
Functional Macros 5-9
Open Macro - DOPEN 5-10
Routine Package SYMREF Name 5-11
Put Macro - DLPUT 5-16
Rewrite Macro - DLRWR 5-17
Delete Macro - DLDEL 5-19
Position Macro - DLPOS 5-20
Get Next Macro - DLGXT 5-25
Get Macro - DLGET 5-26
Return Macro - UFSRET 5-29
Information Macro - DINFO 5-30
Checkpoint Files Macro - DCKPF 5-33
Checkpoint Files And Program Macro - DCKPT 5-34
Rollback Files Macro - DROLF 5-35
Rollback Files And Program Macro - DROLP 5-35
Wrap-up Macro - DWRAP 5-36
Close Macro - DCLOS 5-36

Section VI Indexed Sequential Processor File Organization 6-1
File Information Block Macro 6-3
Functional Macros 6-7
Open Macro - DOPEN 6-8
Routine Package SYMREF Name 6-9
Put Macro - DLPUT 6-14
Rewrite Macro - DLRWR 6-15
Delete Macro - DLDEL 6-17
Position Macro - DLPOS 6-18
Get Next Macro - DLGXT 6-19
Get Macro - DLGET 6-21
Return Macro - UFSRET 6-22
Checkpoint Files Macro - DCKPF 6-23
Checkpoint Files And Program Macro - DCKPT 6-23
Information Macro - DINFO 6-24
Rollback Files Macro - DROLF 6-27

Page

Rollback Files And Program Macro - DROLP 6-28
Wrap-up Macro - DWRAP 6-28
Close Macro - DCLOS 6-28

Section VII DM-IV Data Base File Organizations 7-1
Integrated Format 7-1
Indexed Format 7-1
Dual Data Environment 7-2
Directory Selection 7-2

Section VIII File Generation And Concatenation 8-1
Restrictions 8-1
File Generations 8-2
File Concatenation 8-3

Section IX Time Sharing Environment 9-1
File Specification 9-1
File Accessing 9-2
Terminal Interface 9-3

Section X Exception And Error Conditions 10-1
Exception Conditions And Status Codes . . . 10-1
Label Processing Messages 10-18
Console Messages 10-18

Section XI Dedicated Printer Capability 11-1
Allocating A Printer 11-1
Procedures 11-2
Printer Status 11-2
File Information Block Macro 11-3
Functional Macros 11-5
Open Macro - DOPEN 11-6
Put Macro - DLPUT 11-7
Close Macro - DCLOS 11-9

Section XII Rerun And Tape Utilities 12-1
Adjust Density 12-2
File Name Identified Tape (FNIT) 12-2
Rerun Functions 12-2
Defective Volume Replacement 12-2
Activity Restart 12-4
Print Function 12-5
Volume Size Function 12-6
User External Label Function 12-6
Checkpoint Record Tracking 12-8
Dedicated Tape 12-8

Appendix A Programming Aids A-1

Appendix B File Information Block B-1

Appendix C Programming Examples C-1

Appendix D Glossary D-1

Appendix E Indexed File Overflow E-1
Inventoried Overflow E-2

Index i-1

Page

ILLUSTRATIONS

Figure 1-1 General File Format 1-5
Figure 1-2 General Control Interval Format 1-7
Figure 1-3 Sequential File Control Interval Format 1-8
Figure 1-4 Relative Control Interval Format 1-9
Figure 1-5 Indexed Data Control Interval Format 1-10
Figure 1-6 Indexed Index Control Interval Format 1-11
Figure 1-7 Alternate Key Fine Level Index Format 1-13
Figure 1-8 Alternate Key Coarse Level Index Format 1-14
Figure 1-9 Alternate Key Fine Level Overflow Index Format 1-15
Figure 1-10 Integrated Control Interval Format 1-17
Figure 1-11 ISP Data Control Interval (Page) Format 1-19
Figure 1-12 ISP Index Control Interval (Page) Format . . . 1-19
Figure 1-13 UFF And American National Standard Tape File Format 1-24
Figure 1-14 Volume-Header Label (VOL1) 1-24
Figure 1-15 First File-Header Label (HDR1) 1-26
Figure 1-16 Second File-Header Label (HDR2) 1-28
Figure 1-17 First End-Of-File/Volume Label (EOF1/EOV1) . . 1-30
Figure 1-18 Second End-Of-File/Volume Label (EOF2/EOV2) . . 1-31
Figure 1-19 Unlabeled UFF File Arrangements 1-32
Figure 1-19.1 PPS Block Control Word 1-37.1
Figure 1-19.2 PPS Record Control Word 1-37.2
Figure 1-20 GFRC Tape File Format 1-38
Figure 1-21 GFRC Beginning Of Tape Label (GFRC BTL) 1-39
Figure 1-22 GFRC End-Of-Reel/File Label (EOR/EOF) 1-42
Figure 1-23 GFRC Partial Tape Label (PTL) 1-43
Figure 1-24 Unlabeled GFRC Tape Arrangements 1-44
Figure 1-25 UNLT Tape File Arrangement 1-46
Figure 1-26 H2000 Beginning Of Tape Label (BTL) - 80 Characters 1-54
Figure 1-27 H2000 Beginning Of Tape Label (BTL) - 120 Characters 1-54
Figure 1-28 End-Of-File/Reel Label (EOF/EOR) - 80 Characters 1-56
Figure 1-29 End-Of-File/Reel Label (EOF/EOR) - 120 Characters 1-56
Figure 1-30 End-Of-File/Reel Label Structure 1-57
Figure 9-1 Batch Activity To Create H* File For Execution In Time Sharing Environment 9-4
Figure B-1 File Information Block Format B-2
Figure C-1 GFRC Sequential File Deck Setup For COBOL-74 Program C-1
Figure C-2 COBOL-74 Program To Build A GFRC Sequential File C-2
Figure C-3 GMAP Program To Build A GFRC File C-6
Figure C-4 GMAP Program To Access An ISP File C-13
Figure C-5 GMAP Programs To Build And Position UFF Relative Files C-16
Figure C-6 GMAP Program To Process An Indexed File C-32
Figure C-7 Free-Standing SORT Of A UFF Sequential File . . C-34

Page

TABLES

Table 1-1 Sequential File Functions 1-58
Table 1-2 Relative File Functions 1-59
Table 1-3 Indexed File Functions 1-60
Table 2-1 Sequential File Functions And Processing Modes 2-2
Table 2-2 Sequential File Format Directories 2-18
Table 3-1 H2000 Sequential Tape File Functions And Processing Modes 3-1
Table 3-2 Series 2000 Sequential File Format Directories 3-10
Table 4-1 Relative File Functions And Processing Modes . 4-2
Table 4-2 Relative File Format Directories 4-12
Table 5-1 Indexed File Functions And Processing Modes . . 5-2
Table 5-2 UFF Indexed File Format Directories 5-11
Table 6-1 ISP File Functions And Processing Modes 6-2
Table 6-2 ISP File Format Directories 6-9
Table 7-1 UFAS Directories For DM-IV-TP 7-3
Table 7-2 UFAS Directories For DM-IV (Batch Or Timesharing) 7-3
Table B-1 File Information Block Map B-4

SECTION I

INTRODUCTION

The Unified File Access System (UFAS) provides functional compatibility to all users of the Series 60 and Series 6000 Information Processing Systems. These systems operate under control of the General Comprehensive Operating Supervisor (GCOS) through the Input/Output Supervisor (IOS) and the File Management Supervisor (FMS). UFAS is composed of a set of routines that provide automatic management for file processing.

User input/output operations are implemented through a File Information Block (FIB) macro and by specific macro requests that initiate the sequence of events necessary for file processing.

The Unified File Access System provides the user with the following capabilities:

- Logical input/output processing on files with sequential, relative, indexed, and integrated file organizations in either batch or time sharing environment.
- Support for File and Record Control (GFRC) files on magnetic tape and linked mass storage.
- Support for Series 2000 (H2000), American National Standard, and IBM magnetic tape files.
- Support for Indexed Sequential Processor (ISP) files on mass storage.
- Space and buffer management, and physical input/output management for the sequential, random, and dynamic file access modes.
- Label processing for GFRC, H2000, American National Standard (ANS), and IBM magnetic tape files.
- Complete error checking and initiation of proper error processing as defined by the American National Standard for COBOL-74.
- File integrity protection for both normal and abort processing.

In this discussion of the Unified File Access System the files are arranged by type and structure to enable proper selectivity and efficient use of the system. The major types or categories of files are designated as file organizations and are divided into structures, which are called file formats. The format of mass storage files allocated as random is called Unified File Format (UFF). Tables 1-1, 1-2, and 1-3 list the UFAS record processing functions by file organization and file format. These tables also list the processing mode and access mode with the applicable parameters.

FILE ORGANIZATIONS

Four types of file organizations are supported by the Unified File Access System. These four types are designated sequential, relative, indexed, and integrated.

Sequential

In a sequential file organization the files can only be accessed sequentially and may reside on magnetic tape or mass storage.

Relative

A relative file organization must be allocated as random to a mass storage device, but may be accessed sequentially, randomly, and dynamically. All records are assigned a fixed amount of space and can be referenced by record number starting with 1.

Indexed

An indexed file organization must be allocated as random to a mass storage device, but may be accessed sequentially, randomly, and dynamically. Each record is identified by a unique key, called prime key, and/or one or more subordinate keys, called alternate keys. Both are part of data within the record and are also held in an index.

Integrated

In an integrated file organization the file must be allocated as random to a mass storage device. The records can be of variable length and are identified by a data base key. An integrated file is accessed only through I-D-S/II.

ACCESS MODES

The records in a file can be accessed or retrieved sequentially, randomly, or dynamically. Only mass storage files allocated as random can be accessed in all three modes. The dynamic mode allows file record access in both sequential and random modes through the full duration of an Open function.

Sequential

In the sequential access mode, records are entered in, or retrieved from, a file in a consecutive predecessor-to-successor logical record sequence. All file organizations can be accessed sequentially; however, sequentially ordered files can only be accessed sequentially.

Random

In the random access mode each value of a key is specified by the user to identify the records to be retrieved from, deleted from, or entered in the file. The key value identifies a record independently of any record previously accessed. Random access is restricted to ISP files and UFF files with relative, indexed, and integrated organizations.

Dynamic

The dynamic access mode allows records in a file to be accessed first sequentially then randomly (or vice versa) without closing and reopening the file. This access mode can be used only with ISP files and UFF files having relative, indexed, and integrated organizations. The routines necessary for sequential and random access are loaded in the system whenever the dynamic access mode is specified.

PROCESSING MODES

The processing modes available with UFAS are input, output, input/output (I/O), and extend. A file can be opened for any of these processing modes, depending on whether the function to be performed is Read, Write, Read then Write, or Write after the last record in the file.

Input

If records are to be retrieved only, they can be retrieved from a file that was opened for input.

Output

If records are to be entered only, they can be entered in a file that was opened for output.

Input/Output

To retrieve and alter, replace, insert, or delete records from a file, the file must be opened for input/output.

Extend

To append records to a sequential file on magnetic tape or mass storage, the file must be opened for extend. This processing mode is equivalent to the output mode, except that the file is positioned when opened, immediately following the last logical record currently in the file. Succeeding writes add records to the file. GFRC files residing on mass storage and H2000 files cannot be opened for extend.

USER LABEL PROCESSING (for ANS, UFF and IBM files)

UFAS provides for the processing of User volume, header, and trailer records on ANS, UFF, and IBM tape files. In order for UFAS to process these user labels the following conditions must be met in the FIB:

1. File format must be ANS, UFF, or IBM.

2. File organization must be sequential.

3. Standards file labels must be present on input and created on output.

4. File status code must be specified.

5. A label area must be specified.

6. Label exit table address must be specified.

In addition to the required FIB information, a user label exit table must be constructed. The table is 12 words in length and the format is defined in Section II under "File Information Block Macro".

It should be noted that in EXTEND mode processing, the extend exits are taken during positioning and the output exits are used after positioning. It is the user's responsibility to ensure that the proper read or write user label functions are issued. If a read function is issued during the output exit, the program will be aborted. The same is true of the write function during an input or extend exit.

Unified File Formats

All UFF file space must be allocated as random to a mass storage device (random mass storage). The UFF file space consists of the attribute region, which contains all pertinent information about the file, an optional label region, the data or logical record region, which is divided into control intervals and, for integrated files, an inventory region. Formats of UFF mass storage files are identical, except where noted. Figure 1-1 illustrates the general file format.

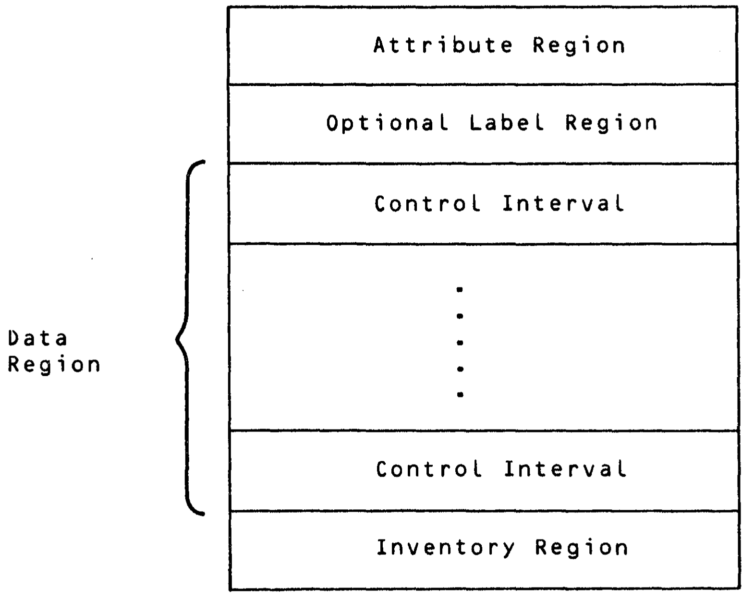


Figure 1-1. General File Format

The file attribute region is of variable length and starts in sector 0 of the mass storage space allocated to the file. This region contains the characteristics (attributes) of the file.

The label region is optional, depending on whether or not the file is labeled. This region is of variable length and starts in the next sector following the attribute region. Currently, the label region contains only the file name but can accommodate future labeling requirements. Together the attribute and label regions occupy at least one control interval at the beginning of the file.

The data region is composed of control intervals in a format that facilitates interchange of files among various systems. Files developed for a particular system can be transferred to a different system with a minimum need of file reformatting.

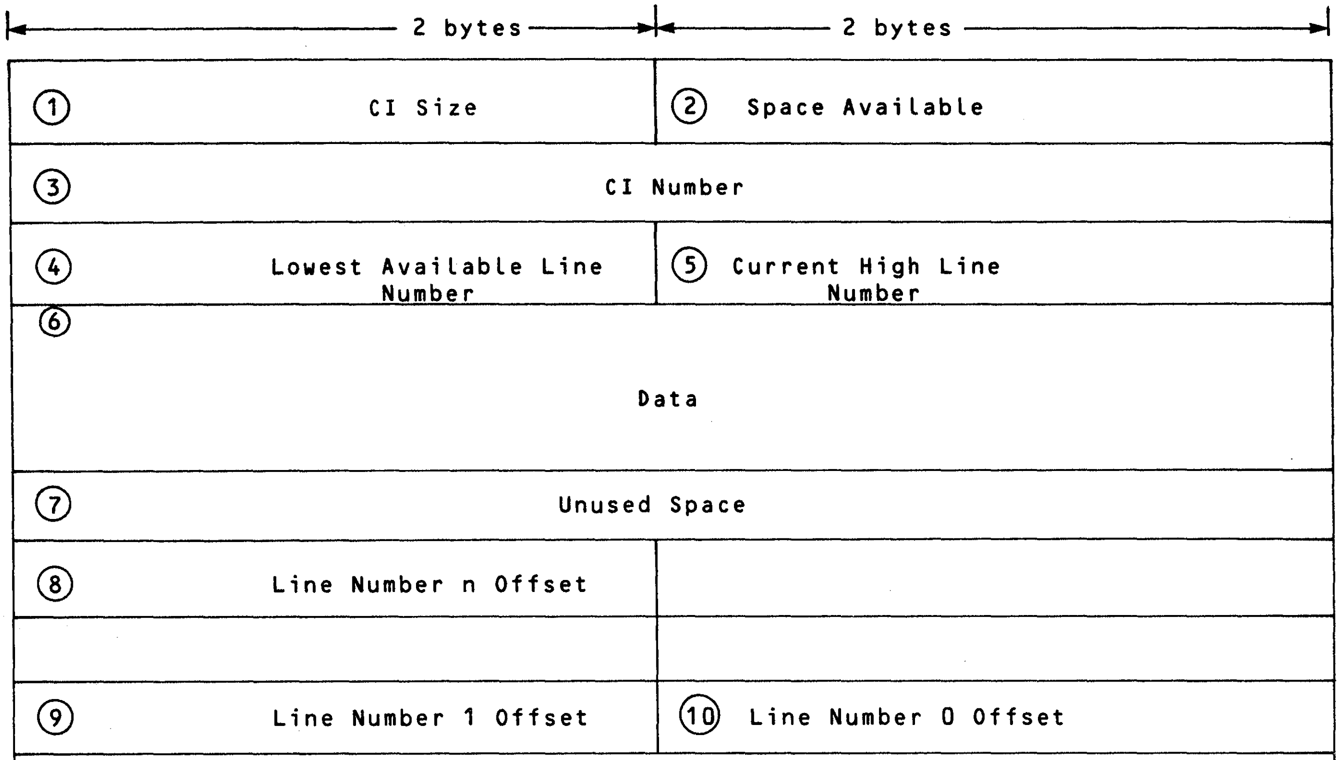
Each control interval has a header, which contains control information, followed by record data. Control intervals of a file are formatted according to their organization (sequential, relative, indexed, and integrated). Indexed and integrated file control intervals contain line offsets and may contain unused space.

Each offset at the end of the indexed and integrated file control intervals points to a data record. Both the indexed and integrated records are chained together with pointers to preserve the logical order within the file. A pointer specifies the control interval and a particular offset at the end of the control interval. The offset points to the record that the pointer specifies. With this procedure records can be moved within the control interval by changing the appropriate offsets in the control interval, instead of having to change the pointers that may be scattered over many control intervals. Thus, with the offsets it is possible to reclaim deleted record space without changing the pointers of all records that may be moved in the processing of a file. The first bit of an offset is set to indicate that the corresponding record was physically deleted and that the offset may be reused.

The inventory region is applicable only to integrated files. This region has the same format as the data control interval.

DATA REGION

The data region contains only control intervals of identical format. Figure 1-2 illustrates the general control interval format for UFF files.



NOTES:

① Active size of this control interval in bytes.

② Total space available in bytes in the control interval.

③ Number assigned to the control interval.

④ First position available in the line offset array.

⑤ Number of positions in the line offset array minus one.

⑥ Data records.

⑦ Space available for line offsets.

⑧, ⑨, ⑩ An array of two-byte fields, each containing a record offset in the control interval.

Figure 1-2. General Control Interval Format

Sequential Format

The control intervals of sequential files on mass storage (see Figure 1-3) contain a two-word header followed by data records. Each data record is composed of a four-byte header followed by data. Bits 18 through 35 of the record header contain the size of the record expressed in bytes.

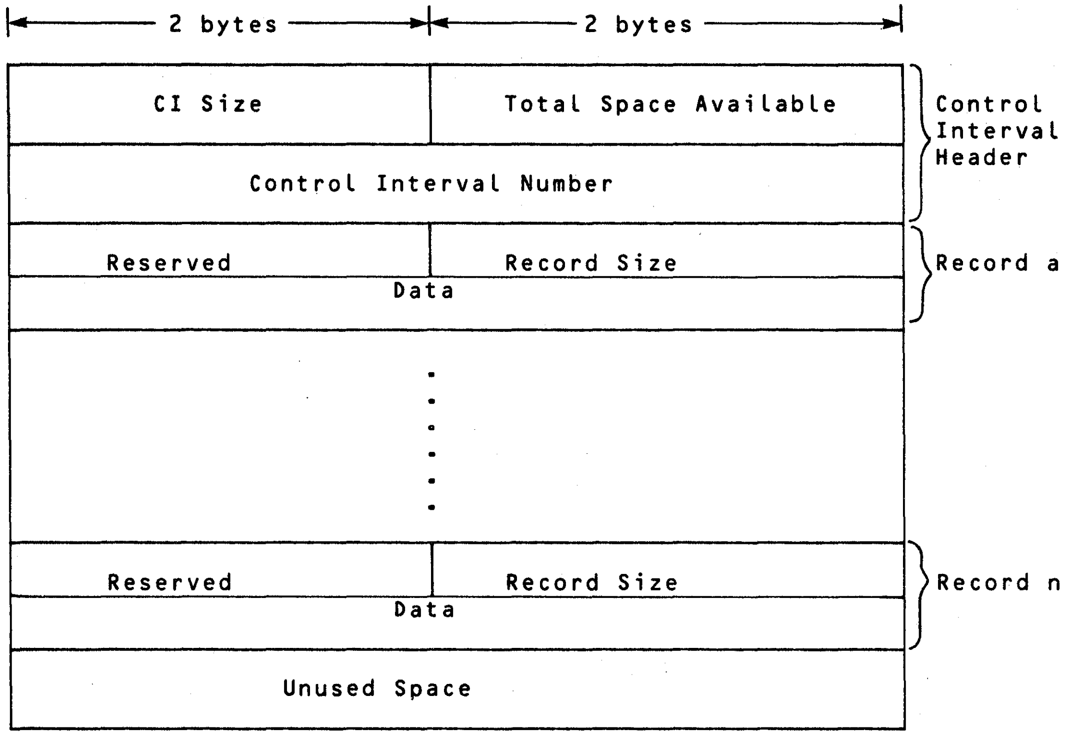


Figure 1-3. Sequential File Control Interval Format

Relative Format

Like control intervals of sequential files on random mass storage, relative file control intervals contain a two-word header followed by data records. A four-byte record header containing the record size precedes the data in each record.

In relative files, the first six bits denote the status. The status of bit one is initialized to zero before the file is built. If a record is entered in a relative slot, the status bit one is set ON, but the bit is reset if the record is subsequently deleted. Figure 1-4 shows the format for the relative control interval.

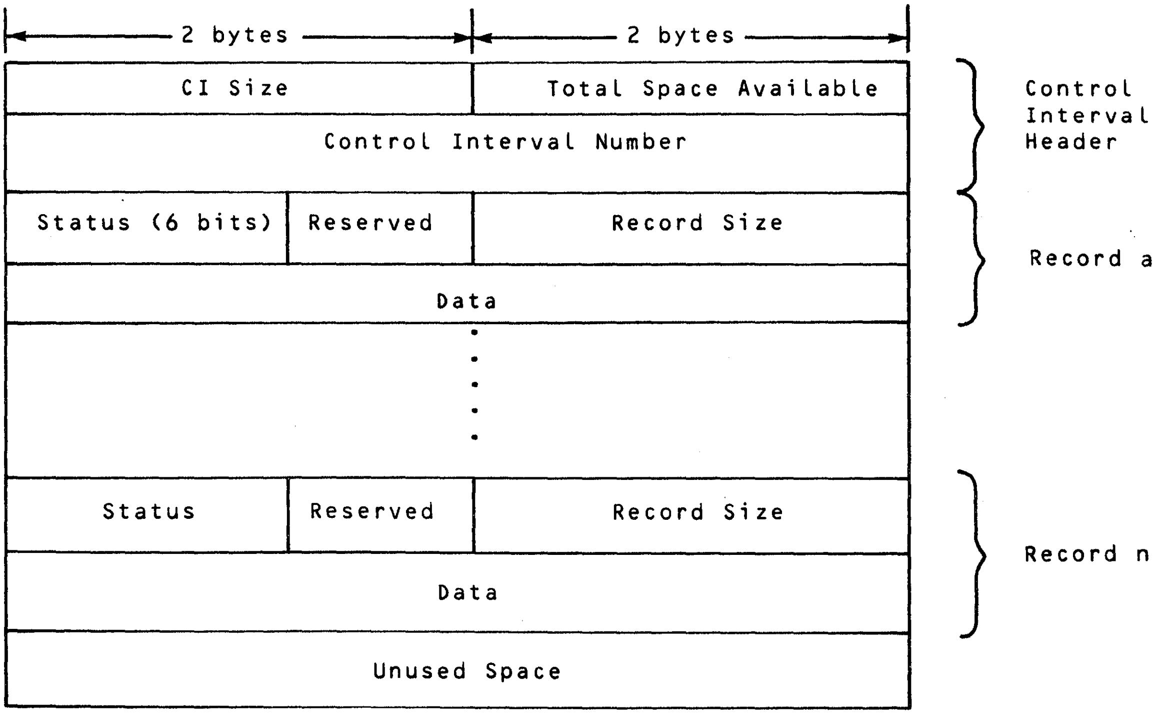


Figure 1-4. Relative Control Interval Format

Indexed Format

In an indexed file, the control interval contains a three-word header, and a two-word record, followed by data records. Each record has a two-word header. The first word of the record header contains a status code that is used by UFAS to optimize record retrieval and the record size.

The second word is the pointer to the next record in sequential order (the indexed file is ordered by prime key). The pointer specifies the control interval number of the next record and the line number details the specific offset that points to the record at the bottom of the control interval. Figure 1-5 illustrates the format of an indexed data control interval.

Each indexed data control interval has a corresponding index entry in a control interval in the index file. Control intervals in the index file contain a three-word header preceding the index entries. These entries consist of a key value and the control interval number.

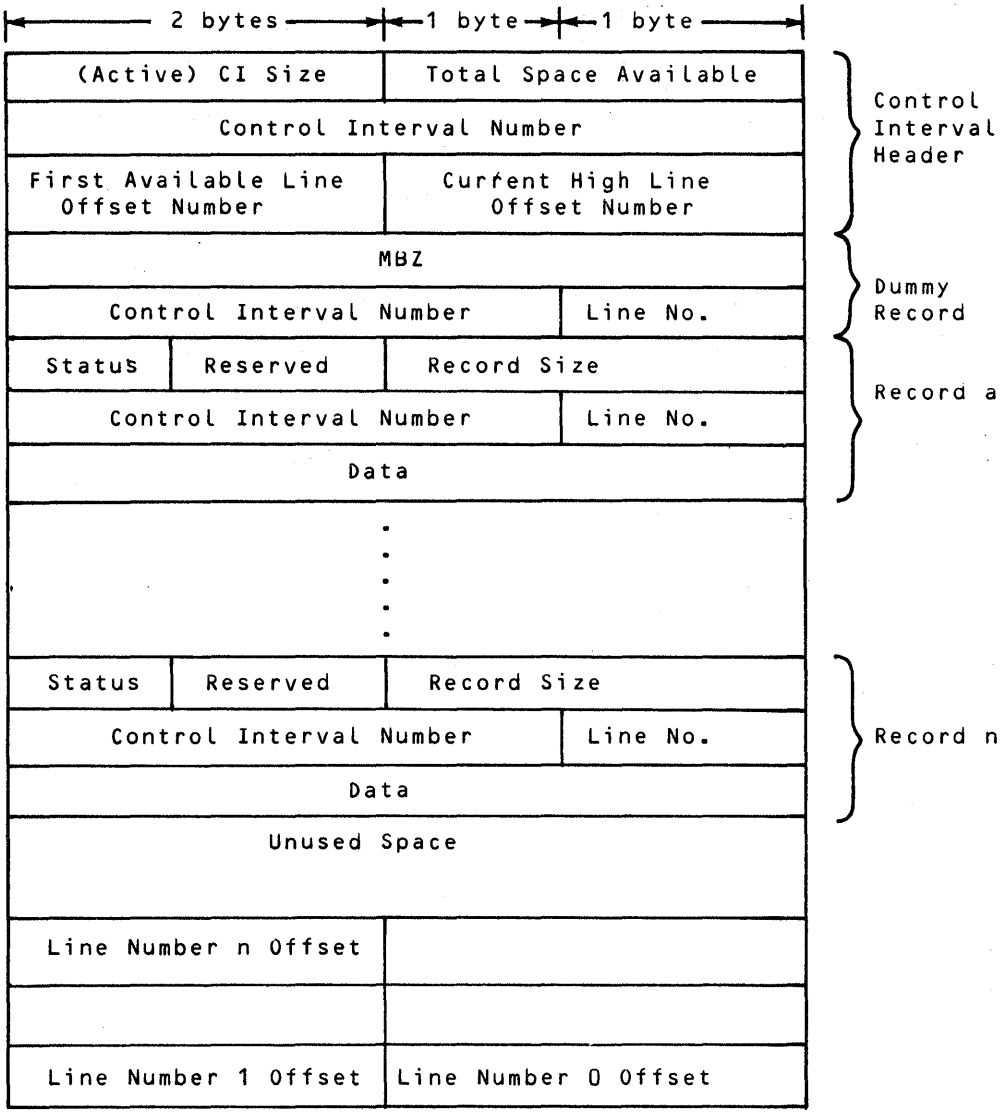
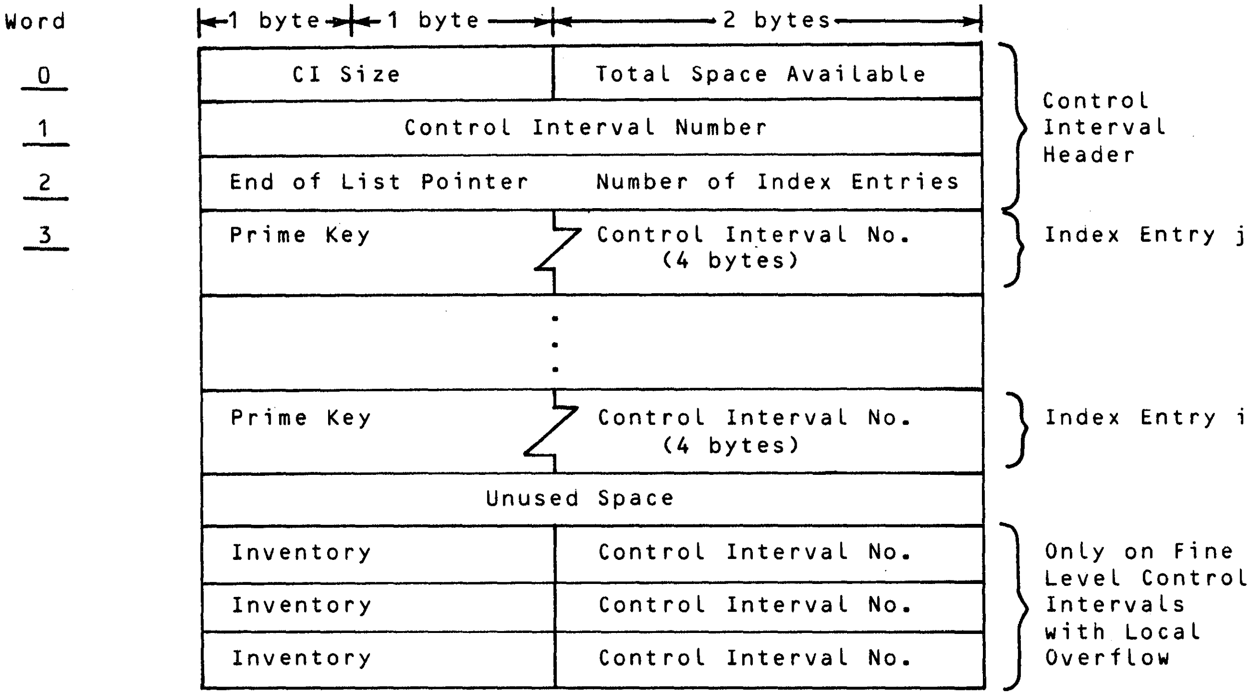


Figure 1-5. Indexed Data Control Interval Format

One prime key index is entered on an index control interval for each data control interval that exists at the time the file is created. The key is the lowest prime key from the succeeding data control interval. When an index control interval becomes full, an entry is made in a higher level index with the same entry format. The key is the highest key in the full control interval. The prime key may have many levels of index. The highest level index has only one control interval. The format of an indexed index control interval is shown in Figure 1-6.



Word	1 byte	1 byte	2 bytes	
0	CI Size		Total Space Available	Control Interval Header
1	Control Interval Number			
2	End of List Pointer		Number of Index Entries	
3	Prime Key		Control Interval No. (4 bytes)	Index Entry j
	. . .			
	Prime Key		Control Interval No. (4 bytes)	Index Entry i
	Unused Space			
	Inventory		Control Interval No.	Only on Fine Level Control Intervals with Local Overflow
	Inventory		Control Interval No.	
	Inventory		Control Interval No.	

Figure 1-6. Indexed Index Control Interval Format

In addition to the prime key index, the UFF indexed index file may contain one or more alternate key indexes. The alternate keys are provided mainly to allow retrieval of records through several paths. An alternate key structure is created for each alternate key index. Each alternate key structure is composed of a fine level, several coarse levels as needed and, if required, a fine level overflow for the fine level. A pointer is kept to the most coarse level control interval and is stored in the file attribute region.

Key entries consisting of a key value and a pointer are placed in the fine level alternate key structure in physical order. This order must be preserved to allow sequential access.

Overflow control intervals are spread throughout the alternate index structure. The overflow control interval handles overflow entries from any fine index control interval in the area. An unused space is available at the end of the file for additional fine level control interval overflow. Fine level index control intervals have pointers to link all fine level control intervals of a particular key. Coarse levels are not changed as records are deleted or inserted.

For files with local overflow, an inventory of the space available in the local overflow control intervals is included on fine level index control intervals. An inventory entry is made for each local overflow control interval in the region of the file indexed by the fine level control interval. For additional information on overflow, see Appendix E.

The control interval of an alternate key fine level index contains a five-word header followed by unique key values and duplicate key values, if any are present. The first word of the header contains the control interval size in bytes and the space available for additional entries expressed in words.

The second word contains the level number and the control interval number. In the third word, the number of key values is retained to be used in the binary search of key values, and an entry size is used to specify size of key value and record address (one word long).

The fourth word has a bit indicator and a next fine level control interval number to provide sequential order of alternate keys. To indicate a next fine level control interval for a fine level overflow, the bit is set to 1; to indicate no fine level overflow, the bit remains unchanged.

The fifth word contains an offset, to indicate the next slot available for a record address when duplicate keys are used, and a space reserved for UFAS.

Every unique key value has a pointer that consists of a control interval number and an offset number.

If a duplicate key exists, bit 0 of the duplicate key pointer is set to 1 to indicate that the record address is at the offset specified. If bit 0 remains unchanged this denotes that the record address is included in this word. Bits 1-17 contain the word offset for the last record address of the duplicate key value. The number of duplicate keys is expressed in bits 19-35 of the pointer. Bit 18 of the pointer is set to 1 if more duplicate keys are present in the next fine level control interval. If duplicate keys exist the record addresses are entered at the end of the control interval. Figure 1-7 shows the format of an alternate key fine level index.

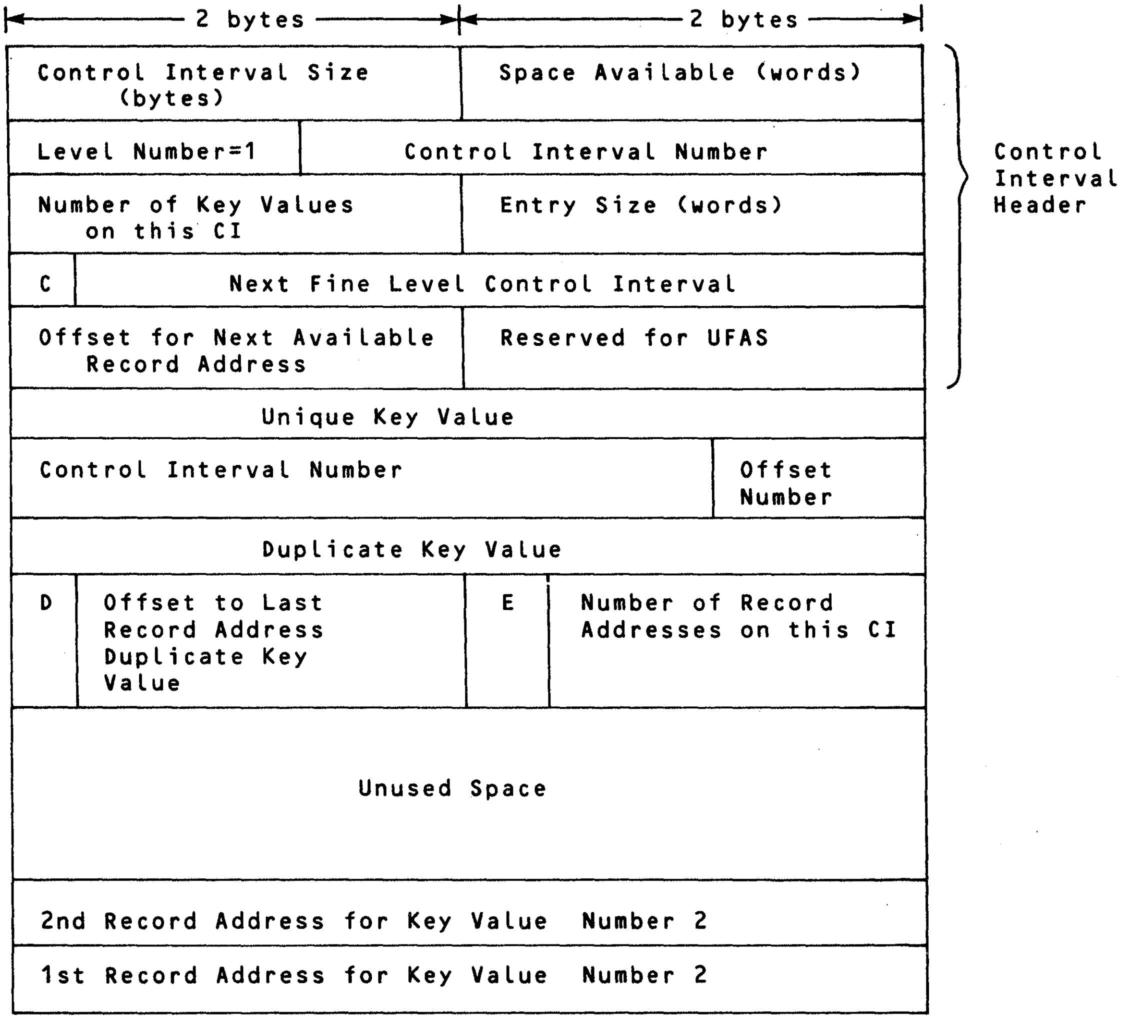

2 bytes
2 bytes
Control Interval Size (bytes)
Space Available (words)
Level Number=1
Control Interval Number
Number of Key Values on this CI
Entry Size (words)
C
Next Fine Level Control Interval
Offset for Next Available Record Address
Reserved for UFAS
Control Interval Header
Unique Key Value
Control Interval Number
Offset Number
Duplicate Key Value
D
Offset to Last Record Address Duplicate Key Value
E
Number of Record Addresses on this CI
Unused Space
2nd Record Address for Key Value Number 2
1st Record Address for Key Value Number 2

Figure 1-7. Alternate Key Fine Level Index Format

In an alternate key coarse level index control interval, the control interval contains a three-word header followed by unique key values. No duplicate key values are entered in the coarse level control interval.

The first word in the header contains the control interval size and space for additional entries; the second word contains the level number and the control interval number; and the third word specifies the number of key values in the control interval and the entry size.

Entries consist of a unique key value and a 36-bit control interval number as shown in Figure 1-8.

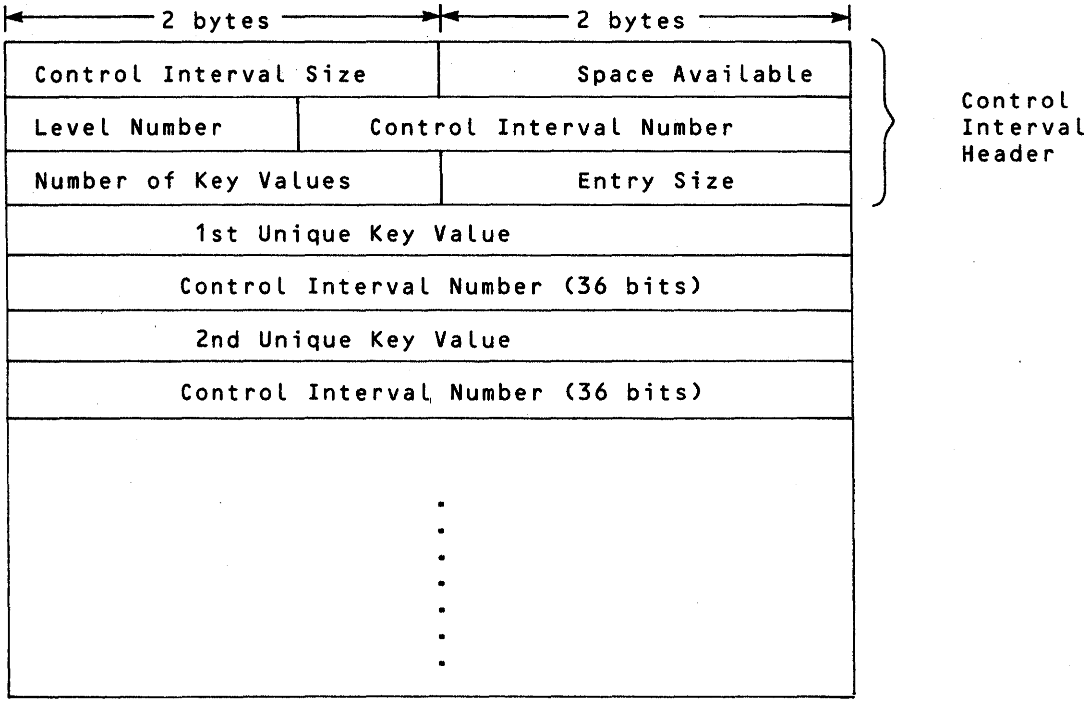


Figure 1-8. Alternate Key Coarse Level Index Format

Alternate key fine level overflow index control intervals are interspersed in the fine level index and are grouped at the end of the file. The fine level overflow control interval is divided into sections; that is, one section for each fine control interval that has overflow entries.

The overflow index control interval has a five-word header that is identical with that of the alternate key fine level index control interval, except for the second word. The second word of the overflow index consists of a bit that indicates an overflow control interval when set to 1 and a control interval number. This header is followed by section descriptions listed in physical order. For duplicate key values, the record addresses are pooled at the end of the control interval. Figure 1-9 illustrates the format of the overflow index control interval.

<table>
<tr><th colspan="2">2 bytes</th><th>2 bytes</th><th></th></tr>
<tr><td colspan="2">Control Interval Size (bytes)</td><td>Space Available (words)</td><td rowspan="5">Control Interval Header</td></tr>
<tr><td>F</td><td colspan="2">Control Interval Number</td></tr>
<tr><td colspan="2">Number of Key Values</td><td>Entry Size</td></tr>
<tr><td>C</td><td colspan="2">Next Control Interval Number</td></tr>
<tr><td colspan="2">Offset for Next Available Record Address</td><td>Reserved for UFAS</td></tr>
<tr><td colspan="2">Offset Section 1 (words)</td><td>Number of Entries</td><td rowspan="9"></td></tr>
<tr><td>C</td><td colspan="2">Next Control Interval Number</td></tr>
<tr><td colspan="2">Offset Section 2 (words)</td><td>Number of Entries</td></tr>
<tr><td>C</td><td colspan="2">Next Control Interval Number</td></tr>
<tr><td colspan="3">.
.
.
.</td></tr>
<tr><td colspan="3">Section 1 Entries</td></tr>
<tr><td colspan="3">Section 2 Entries</td></tr>
<tr><td colspan="3">Unused Space</td></tr>
<tr><td colspan="3">Record Addresses</td></tr>
</table>

Figure 1-9. Alternate Key Fine Level Overflow Index Format

Integrated Format

The control intervals of an integrated file are composed of a header, logical records, and line offsets. A control interval header is three words long and contains information necessary to manage the space within the control interval.

In addition to the record size and status, each header in the control interval contains the record type. This field is used to identify a specific record occurrence as one of several types declared in the external control table (schema).

Records in an integrated file may contain set pointers, which are also declared in the schema. Set pointers are required for records that participate in sets and any number of set pointers are allowed. Records that are members in the calc set contain a next and prior pointer for the calc set. All other set members require next, prior, and owner pointers. Set owners require next and prior pointers. Set pointers are 2, 3 or 4 bytes long. Sets that are contained within one area have pointers large enough to address the records within that area. Sets that may span areas have pointers large enough to address the records in the data base (see the following diagrams).

Size of Intra-Area Pointers (bits)

Number of Data Base Keys in Data Base
(NA = not applicable)

		$< 2^{18}$	$\geq 2^{18}$	$\geq 2^{27}$
			$< 2^{27}$	
Number of Data Base Keys in Area	$< 2^{18}$	18	18	18
	$\geq 2^{18}$ $< 2^{27}$	NA	27	27
	$\geq 2^{27}$	NA	NA	36

Size of Inter-Area Pointers (bits)

Number of Data Base Keys in Data Base

		$< 2^{18}$	$\geq 2^{18}$	$\geq 2^{27}$
			$< 2^{27}$	
Number of Data Base Keys in Area	$< 2^{18}$	18	27	36
	$\geq 2^{18}$ $\leq 2^{27}$	NA	27	36
	$\geq 2^{27}$	NA	NA	36

Figure 1-10 illustrates the format for the integrated control interval.

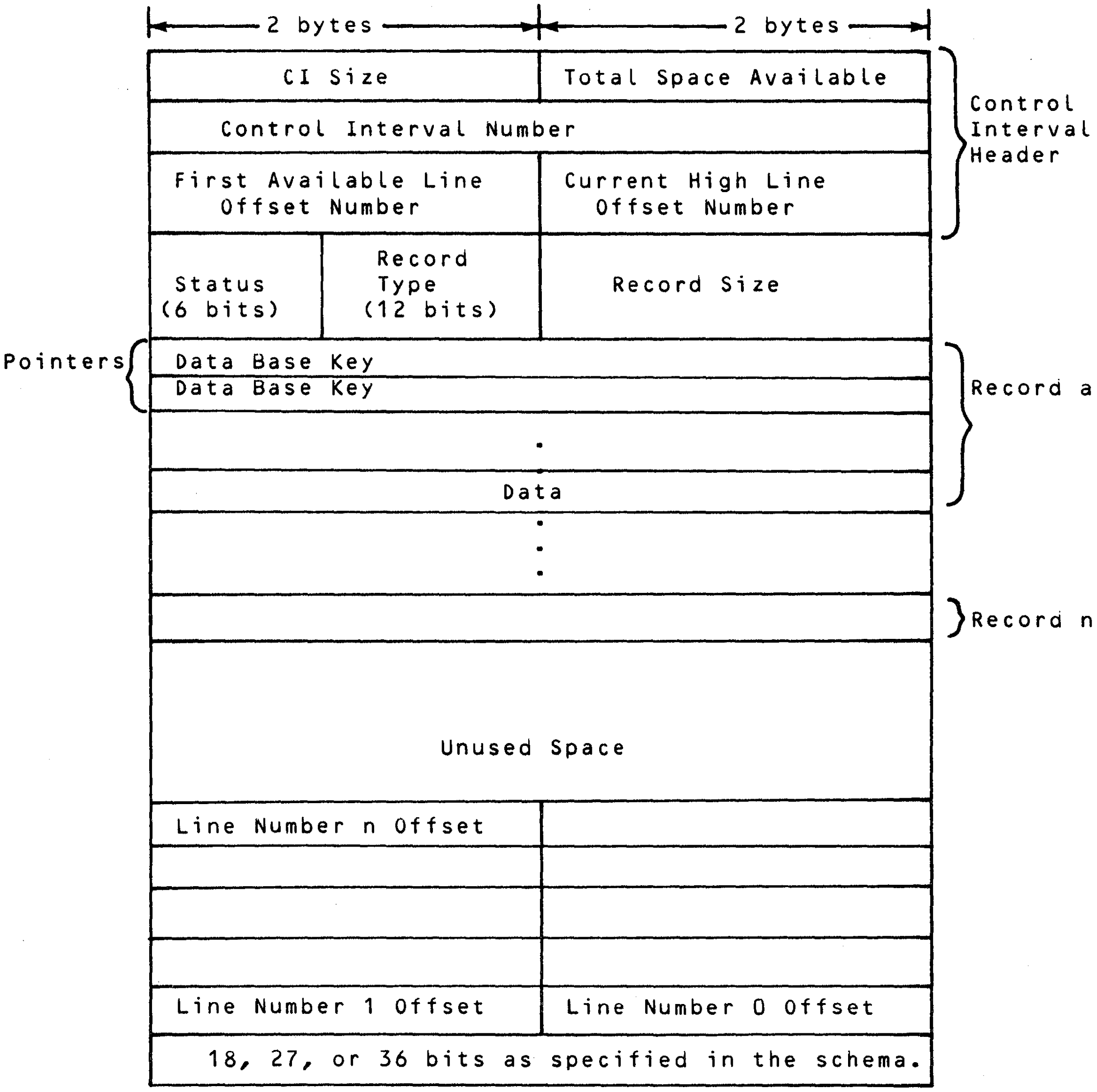


Figure 1-10. Integrated Control Interval Format

INVENTORY REGION

Integrated files contain an inventory region following the data control intervals. Each inventory control interval is identical in format with a data control interval and contains only one record. This record has no header word and no pointer fields. The inventory records have a one-byte slot for each data control interval that describes the space available in that control interval. Inventory control intervals have the same size, are read and written into the same buffers, and are handled by the same physical I/O routines as data control intervals.

ISP File Format

Like the UFF file space, the space for ISP files must be allocated as random to a mass storage device (random mass storage). The ISP file space consists of a data file and an index file. Each of these files is divided into fixed-length control intervals, also referred to as pages. The control interval size of the data file may vary from that of the index file and is specified when the file is created (see Index Sequential Processor manual).

The first page of both the data file and the index file always begins in sector 0 of the allocated file space. Only the first page of the data and index files has a control record called the utilization record which contains the file characteristics. The size of the utilization record in the data file is 62 words and in the index file is 64 words. Utilization records are created when the file is initialized and only the utilization record for the data file is updated when the file is updated. The index utilization record is preceded by a control word that is identical with the record control word for the utilization record in the data file.

In the ISP data files, each control interval begins with a control word. A record word always precedes the data in a record. The data is followed by a pointer word.

The pointer word identifies the location of a record with the smallest key value that is greater than that of the record the pointer word follows. The last record in the file, that is, the record with the greatest key value is followed by a pointer to an end-of-file record. A record may vary in size but must reside within one control interval and the key value must begin always in the same position and have the same size for every record of the file. Figure 1-11 illustrates the format of an ISP data control interval.

The index file is divided into a coarse index and a fine index. The fine index is composed of the control intervals that contain an entry for each control interval in the data file. The coarse index control intervals contain an entry for each fine index control interval. A coarse index is created only if more than one fine index is present.

Each control interval of the index file also begins with a control word. An index entry consists of a key value followed by a page number. The page number for an entry in the coarse index is that of a fine index page whose largest key value is equal to the key value in the coarse index entry. The page number for a fine index entry is that of the data page which immediately precedes the data page whose lowest key value at the time of file initialization was equal to the key value in the fine index entry. The character offset in an index entry is the same as in a data record. Figure 1-12 depicts the ISP index control interval format.

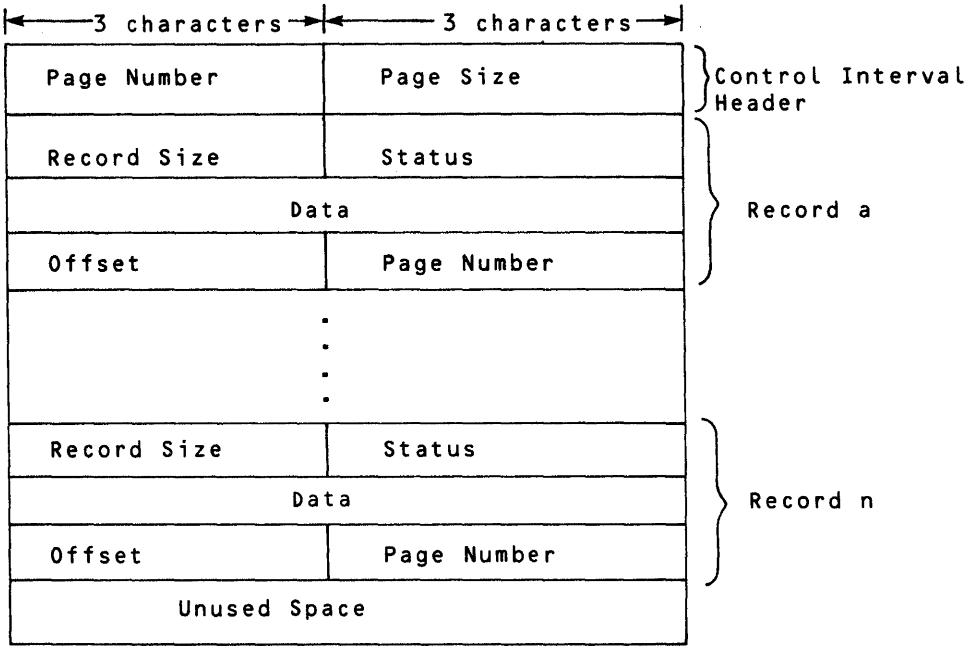


Figure 1-11. ISP Data Control Interval (Page) Format

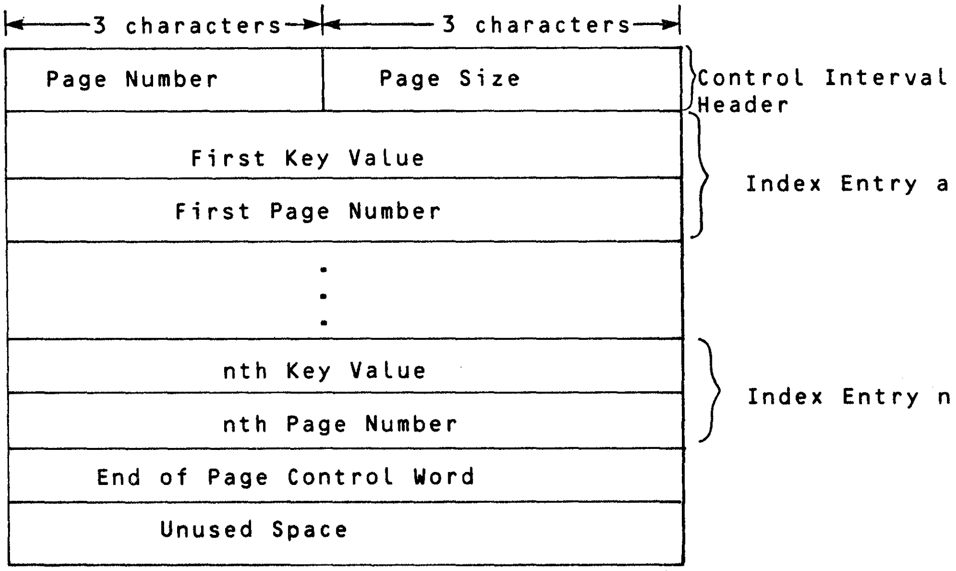


Figure 1-12. ISP Index Control Interval (Page) Format

UFF And American National Standard Tape File Format

These tape file formats are discussed together because they are almost identical. The two formats differ only in the entry in column 80 of VOL1 label and the data content. UFF sequential files are identified by an ASCII H in the mentioned field, which denotes the Honeywell version of the American National Standard. The data content for UFF sequential files may be any 8-bit plus parity character set; however, all control fields and labels are expressed in ASCII characters. The UFF sequential file is non-interchangeable except across Honeywell lines for UFF.

PHYSICAL ATTRIBUTES

9-track tape

- Variable- or fixed-length records (If variable length, each logical record is preceded by a 4-byte record-length field - record length is in ASCII.)
- Block serial number (BSN) - Optional (6 characters ASCII)
- Byte recording mode (write and read 9-track - four 8-bit bytes)
- Labels (contain ASCII characters)- Optional for UFF; required for ANSI.
- Only ASCII character set for American National Standard interchange tapes
- Any 8-bit plus parity character set is permissible for data content of UFF sequential files; however, all control fields and labels are expressed in ASCII characters.

GENERAL FORMAT

```
          ┌         Volume     VOL1                Required
          │                  ┌ UVL1 ┐
          │    User Labels   └ UVLn ┘              Optional
 Header   │                  ┌ HDR1 ┐              Required
 Label   <   Header Labels   │ HDR2 ┘
 Group    │                  └ HDRn                Optional
          │    User Labels   ┌ UHLa ┐
          │                  └ UHLz ┘              Optional
          └      Tape Mark     TM                  Required
 File                        ┌ Data Block 1 ┐
 Data                        └ Data Block n ┘      Optional
          ┌      Tape Mark     TM                  Required
          │                  ┌ EOF1 ┐              Required
 Trailer  │ Trailer Labels   │ EOF2 ┘
 Label   <                   └ EOFn                Optional
 Group    │                  ┌ UTLa ┐
          │    User Labels   └ UTLz ┘              Optional
          │      Tape Mark     TM                  Required
          └      Tape Mark     TM                  Required
```

DATA FORMAT

The space on which a file is stored is subdivided into blocks. A block may contain an integral number of records or a record may span several blocks.

A file may contain either fixed-length records or variable-length records. In a file with variable-length records the record-length field, which consists of the first four bytes of each record, contains the record size in ASCII decimal-digits, right justified.

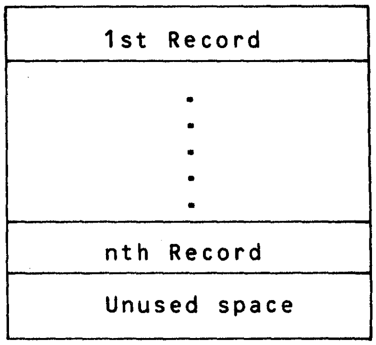
Fixed-length records do not contain the record-length field. In this instance, the block contains nothing but records and possibly unused space.

Padding is used to ensure an integral number of words in the block. Records, including the record-length field, begin and end on byte boundaries which do not necessarily coincide with word boundaries. The padding character is the circumflex accent (^ - 136 octal). Records can be segmented (spanned). A one-byte segment control field precedes each record-length field. The combined fields are referred to as a record control field. For a non-spanned variable-length record, the record control field is the record-length field.

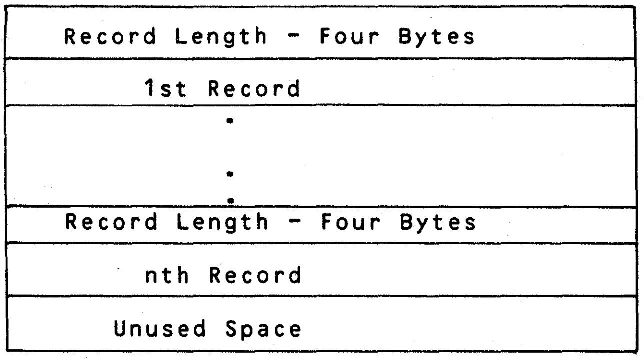
The spanned record control field values are as follows:

0 - segment contained entirely within this block
1 - initial segment
2 - intermediate segment
3 - last segment

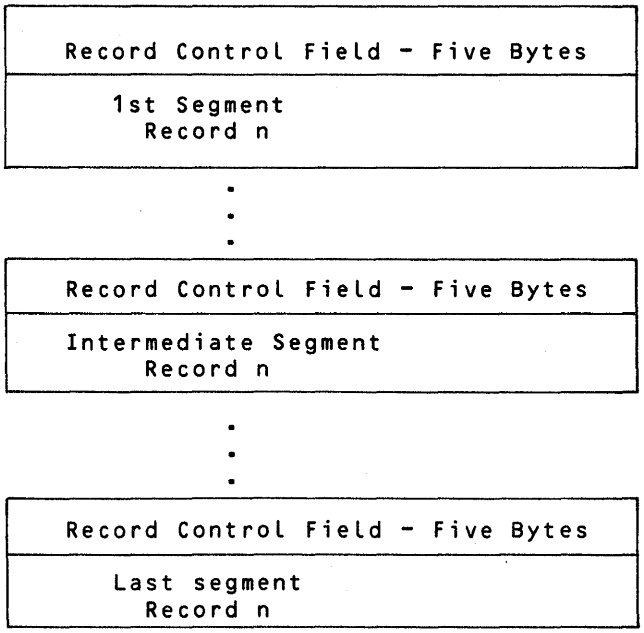
The following diagrams illustrate the block formats for fixed-length, variable-length, and spanned records.



Fixed-Length Records



Variable-Length Records


Record Control Field - Five Bytes
1st Segment
Record n
Record Control Field - Five Bytes
Intermediate Segment
Record n
Record Control Field - Five Bytes
Last segment
Record n

Spanned Record

Only relevant data blocks are written on a 9-track tape reel used for interchange. Checkpoint records are not allowed.

If block serial numbers are specified by the user, characters 51 and 52 of the HDR2 label for the file give the length of the block serial number field. The block serial number must precede immediately the first data record of each block.

LABEL FORMATS AND PROCESSING

The Labeled UFF and American National Standard format conform to one of the following arrangements: single-file single-volume, single-file multi-volume, multifile single-volume, or multifile multi-volume. If a file is contained on one or more volumes, the volume or volumes containing a file are considered a volume set. For tape files the terms reel and volume are used interchangeably.

*

Both types of file formats contain the following labels:

1. Volume-Header Label (VOL1)
2. First File-Header Label (HDR1)
3. Second File-Header Label (HDR2) - Optional
4. First End-of-Volume Label (EOV1)
5. Second End-of-Volume Label (EOV2) - Required if HDR2 is present; otherwise, must be omitted.
6. First End-of-File Label (EOF1)
7. Second End-of-File Label (EOF2) - Required if HDR2 is present; otherwise, must be omitted.
8. Other optional labels (HDR3-HDR9, EOV3-EOV9, EOF3-EOF9)
9. User Labels (UVLn, UHLa, UTLa) - Optional

These formats are applicable only to 9-track tapes, that are processed with RT9/WT9 commands and can be recorded in NRZI or be phase encoded. The end of file mark is octal 23 and the recording mode is the byte.

If a file is multi-volume, items 1 through 5 appear on all volumes, except the last one. The last volume contains items 1 through 3, and items 6 and 7. If a given volume contains more than one file, items 2, 3, 6, and 7 are repeated for each file. Multi-file multi-volume combinations are permitted.

The various file arrangements available with these formats are illustrated in Figure 1-13.

Single-File Single-Volume

```
VOL1 HDR1*---File A---*EOF1**
(VOL1 UVLn HDRn UHLa*---File A---*EOFn UTLa**) (see note)
```

Single-File Multi-Volume

```
VOL1 HDR1*---First Section of File A---*EOV1**
VOL1 HDR1*---Last Section of File A---*EOF1**
```

Multifile Single-Volume

```
VOL1 HDR1*---File A---*EOF1*HDR1*---File B---*EOF1**
```

Multifile Multi-Volume

```
VOL1 HDR1*---File A---*EOF*HDR1*---First Section of File B---*EOV1**
VOL1 HDR1*---Last Section of File B---*EOF1*HDR1*---File C---*EOF1**
```

NOTE:

Asterisk (*) represents tape mark

This is the general format of a label tape file. Optional labels may exist on input for any of the file arrangements listed.

Figure 1-13. UFF And American National Standard Tape File Format

The name and characteristics of the label fields for the Volume-Header Labels, File-Header Label, and End-of-File/Volume Labels are listed and described in Figures 1-14 through 1-18.

Character Position	Field Name	Contents
1-4	Label Identifier and Number	VOL1
5-10	Volume Serial Number	6 alphanumeric characters
11	Accessibility	1 alphanumeric character indicating restrictions
12-37	Reserved for Future Use	Spaces
38-51	Owner Identifier	14 alphanumeric ASCII characters
52-73	Reserved for Future Use	Spaces
74-79	Next Volume Serial Number	6 alphanumeric ASCII characters
80	Label Standard Version	3 or H

Figure 1-14. Volume-Header Label (VOL1)

Volume-Header Label (VOL1)

LABEL IDENTIFIER AND NUMBER: When an input volume is opened, UFAS examines the first four bytes of the first block on that volume for the ASCII literal string VOL1. If VOL1 is present, it is accepted as a legitimate volume-header label; otherwise, an error condition exists and an operator message is generated. The operator can accept the reel and all subsequent reels as valid, mount a different reel to be processed, or abort the job.

When an output volume is opened, UFAS enters the ASCII literal string VOL1 in the first four bytes of the first label block on the output volume.

Up to nine user volume labels, UVL1 through UVL9, may follow the volume-header label on input. User volume labels are not written on output.

VOLUME SERIAL NUMBER: The volume serial number consists of six alphanumeric ASCII characters. This number is left-justified with blanks filled to the right if the actual serial number is less than six characters.

On output volumes, the volume serial number is placed in this field, left-justified with blanks filled if necessary.

ACCESSIBILITY: On input volumes, this field is ignored. On output volumes, this field is initialized to an ASCII space.

OWNER IDENTIFIER: This field provides identification for the owner or assignee to whom this volume belongs.

On input, this field is ignored. On output, it is initialized to spaces.

RESERVED: On input volumes, this field is ignored by UFAS. On output volumes, this field is initialized to ASCII spaces.

NEXT VOLUME SERIAL NUMBER: This field contains the serial number of the next volume to be used in volume swap label checking for UFF file types. This field contains blanks for all other file types.

LABEL STANDARD VERSION: On input, this field is ignored. On output, a 3 or H is placed in this field. The 3 refers to version numbers of the International Standard for Magnetic Tape Labels (X3L5/419T). UFAS also accepts versions 1 and 2 of the label standards.

An H in column 80 denotes the Honeywell version (UFF sequential file on tape) of the American National Standard. The tape is non-interchangeable except across Honeywell lines for unified file format.

Character Position	Field Name	Contents
1-4	Label Identifier and Number	HDR1
5-21	File Identifier	Any alphanumeric characters
22-27	File Set Identifier	The file serial number of the first volume of this file set
28-31	File Section Number	0001 on first volume
32-35	File Sequence Number	4 numeric characters
36-39	Generation Number	4 numeric characters
40-41	Generation Version Number	2 numeric characters
42-47	Creation Date	A space followed by 5 numeric characters
48-53	Expiration Date	A space followed by 5 numeric characters
54	Accessibility	1 alphanumeric character
55-60	Block Count	Must be zero
61-73	System Code	13 alphanumeric characters
74-80	Reserved for Future Use	Spaces

Figure 1-15. First File-Header Label (HDR1)

First File-Header Label (HDR1)

LABEL IDENTIFIER AND NUMBER: On input volumes UFAS searches for a HDR1 label following the VOL1 label. There may be any number of user header labels identified by UVLn, where "n" can be any alphanumeric character, in any order following the VOL1 label. If present, they are bypassed until the HDR1 label block is found. Any other label identifier is considered an error and an operator message is generated. The operator can accept the reel and all subsequent reels as valid, mount the correct reel to be processed, or abort the job. If the volume is a multifile volume and this is not the first file on the volume, UFAS expects the first label in the file-header label group to be a HDR1 label block. If not, the preceding error notification/response is issued. Whenever a volume ends within a file, the continuation of the file on the next volume must also be preceded by a HDR1 label.

On output, the HDR1 label block of the first file on the volume follows immediately the VOL1 label. The HDR1 label block is the first label block in a file-header label group for all subsequent files on a multifile volume.

FILE IDENTIFIER: On input, this field is compared to the file name if one was specified in the FIB macro. If a file name was not specified, the field is ignored. If a comparison is made and an unequal condition is found, an error condition exists. The same error procedure as described in the HDR1 identifier field is adopted. This field is left justified with spaces filled to the right.

On output, the file name is entered in this field left justified, with spaces filled to the right. If a file name is not present, this field is set to ASCII spaces.

FILE SET IDENTIFIER: On input, this field is processed in the same manner as the volume serial number. This field should be the same as that of the volume serial number for the first volume on which this file exists.

FILE SECTION NUMBER: This field must be ASCII 0001 on the first header label of each file. This field is incremented by one in each subsequent volume of the same file.

FILE SEQUENCE NUMBER: On input, this field contains a number from 0001 to 9999 indicating the relative position of the file within a multifile set in a volume. This number is always 0001 for a single file set.

On output, the first file on a volume has this field initialized to 0001. On each subsequent file in a multifile volume, this field is initialized to the proper number to indicate the position of the file within the file set (i.e., 0002-9999).

GENERATION NUMBER: This field is verified by UFAS on input tape files that have been selected for file concatenation via the $ FILGP control card. This field is initialized to the file generation number passed to UFAS by the Peripheral Allocator on output files that are members of a file group index.

This field is set to 0001 on output tape files that are not members of a file group index. No verification of this field takes place on input tape files that are either not members of a file group index or have not been selected for file generation via the $ FILGP control card.

GENERATION VERSION NUMBER: On input, UFAS ignores this field. On output, this field is initialized to ASCII 00.

CREATION DATE: This field consists of an ASCII space followed by two numeric characters for the year, followed by three numeric characters for the day (001 to 366) within the year. On input files, this field is ignored.

On output files, this field is initialized to the current date by UFAS.

EXPIRATION DATE: This field has the same format as the creation date field. On input UFAS ignores this field.

On output, the expiration date is compared to the current date to determine if the file can be overwritten. If the tape is unexpired, a message is issued to the operator; the operator then has the option to ignore the expiration date or abort the job. The new expiration date is calculated by adding the retention period specified in the FIB macro to today's date and placing it in the label.

ACCESSIBILITY: On input volumes, this field is ignored. On output, the field is initialized to an ASCII space.

BLOCK COUNT: On input files, this field is ignored. On output, this field is initialized to an ASCII 000000.

SYSTEM CODE: This field is ignored by UFAS on input files; on output files, this field is initialized to a system code; e.g., "SR2.0HIS".

RESERVED: This field is ignored by UFAS on input files; on output files, this field is initialized to ASCII spaces.

Character Position	Field Name	Contents
1-4	Label Identifier and Number	HDR2
5	Record Format	F - Fixed-length V - Variable-length S - Spanned U - Undefined
6-10	Block Length	5 numeric characters
11-15	Record Length	5 numeric characters
16-50	Reserved for System	Any alphanumeric characters
51-52	Buffer Offset Length	2 numeric characters
53-80	Reserved for Future Use	Spaces

Figure 1-16. Second File-Header Label (HDR2)

LABEL IDENTIFIER AND NUMBER: When an input file is opened, UFAS examines the first four bytes of the label block immediately following the HDR1 label block. If that label block does not contain the ASCII characters HDR2, the label routine assumes that a HDR2 label does not exist and skips to the first tape mark following the HDR1 label block. If a HDR2 label block is the next block, it is processed and the label routine then skips to the tape mark. In either case, additional file-header labels (HDR3 through HDR9) and user header labels (UHLa) are bypassed on input. If a tape mark is found, label processing on input is assumed to be completed. On output, UFAS writes a HDR2 label block immediately following a HDR1 label block (unless user specifies that labels are omitted). A tape mark is written immediately following the HDR2 label block.

BLOCK LENGTH: This field contains the data block length in bytes. For fixed-length records this is the actual block size. For variable-length records this is the maximum physical block length less the block serial number field length. The block serial number field, which is optional, may appear as the first six bytes of each data block. The block-length field is expressed in ASCII numeric characters, right justified with zero fill.

On input, this field is compared to the block size specified in the FIB macro. If the FIB macro value is not zero and is less than the HDR2 block size, an exception condition occurs. If the FIB macro value is zero, greater than, or equal to the HDR2 block size, the HDR2 value is used. On output, the block length expressed in ASCII and right justified, is entered in this field.

RECORD LENGTH: This field contains the record length in bytes for logical records. For fixed-length records this field contains the actual record length. For variable-length records this field contains the maximum record length excluding the four-byte, record-length field that appears as the first four bytes of each logical record.

On input, UFAS verifies this field against the value in the FIB macro. If the FIB macro value is less than the HDR2 value, an exception condition occurs. If the FIB macro value is greater than or equal to the HDR2 value, the HDR2 value is used. On output the record length is placed in this field in ASCII characters.

RESERVED: On input, this field is ignored by UFAS. On output, this field is initialized to ASCII spaces.

BUFFER OFFSET LENGTH: On input, this field is used by UFAS to determine if block serial numbers are used. On output this field is initialized to the size of the block serial number field, if block serial numbers are specified.

Character Position	Field Name	Contents
1-4	Label Identifier and Number	EOF1 or EOV1
5-54	Same as the corresponding fields in HDR1	Same as the corresponding fields in HDR1
55-60	Block Count	6 numeric characters denoting the number of data blocks
61-73	System Code	13 alphanumeric characters
74-79	Next Volume Serial Number	6 alphanumeric ASCII characters
80	Reserved	Spaces

Figure 1-17. First End-Of-File/Volume Label (EOF1/EOV1)

First End-of-File/Volume Label (EOF1/EOV1)

LABEL IDENTIFIER AND NUMBER: On input files, UFAS searches for an EOF1 or an EOV1 following a tape mark. When an EOV1 label block is encountered on input, any subsequent end-of-volume labels (EOV2 through EOV9) are bypassed and reel switching takes place. When an EOF1 label block is encountered, reel switching does not take place and that file is assumed to be complete within that volume. Subsequent end-of-file labels (EOF2 through EOF9) and user trailer labels (UTLa) are ignored.

If neither an EOV1 or EOF1 appear as the first trailer label block on an input file, an error condition exists. A message is then sent to the operator who may continue execution, force an end-of-file condition, or abort the job.

UFAS always seeks an EOF1 label block in the close processing mode, as the first trailer label encountered.

On output files, UFAS produces an EOV1 label block preceded by a tape mark whenever it encounters an end-of-reel foil while processing the file. The UFAS close processing always produces an EOF1 label as the first trailer label on an output file.

CORRESPONDING HDR1 FIELDS: On input files, these fields are ignored by UFAS; on output, these fields are provided under the rules stated in the corresponding field descriptions in the HDR1 label block definition.

BLOCK COUNT: On input, UFAS checks this field against the block count maintained in the file control table for this file.

On output files, UFAS converts the block count to ASCII and places it in this field right-justified and zero-filled to the left. This field indicates the number of data blocks of the file recorded on the volume.

SYSTEM CODE: On input files, this field is ignored by UFAS. On output, this field is initialized to a system code; e.g., "SR2.OHIS".

NEXT VOLUME SERIAL NUMBER: (EOV1 label only) This field contains the serial number of the next volume for UFF file types. This field contains spaces for all other file types.

RESERVED: On input files, this field is ignored. On output files, this field is initialized to ASCII spaces.

Character Position	Field Name	Contents
1-4	Label Identifier and Number	EOF2 or EOV2
5	Record Format	Same as HDR2
6-10	Block Length	Same as HDR2
11-15	Record Length	Same as HDR2
16-50	Reserved	Spaces
51-52	Buffer Offset Length	2 numeric characters
53-80	Reserved	Spaces

Figure 1-18. Second End-Of-File/Volume Label (EOF2/EOV2)

Second End-of-File/Volume Label (EOF2/EOV2)

LABEL IDENTIFIER AND NUMBER: On input files, these labels and any subsequent label blocks (EOF3-EOF9, UTLa-UTLz, or EOV3-EOV9) are ignored. UFAS skips to the following tape mark.

On output files, an EOF2 label block is written immediately following an EOF1 label block (if a HDR2 was written at the beginning of the file). Then two tape marks are written following the EOF2 label block. An EOV2 label block is written following an EOV1 label block and two tape marks are written immediately after the EOV2 label block.

RECORD FORMAT: Ignored by UFAS on input files. Same action taken on output as for HDR2 label conditions for the corresponding field.

BLOCK LENGTH: Ignored by UFAS on input files. Same action taken on output as for HDR2 label conditions for the corresponding field.

RECORD LENGTH: Ignored by UFAS on input files. Same action taken on output as for HDR2 label conditions for the corresponding field.

RESERVED: Ignored by UFAS on input files. Set to ASCII spaces on output files.

BUFFER OFFSET LENGTH: Ignored by UFAS on input files. On output, this field is initialized to the size of the block serial number field, if hlock serial numbers were specified.

RESERVED: Ignored by UFAS on input files. Set to ASCII spaces on output files.

Multi-Volume Output Processing

When an end-of-volume label (EOV1) is prepared and written, the operator is requested to enter the serial number of the next scratch tape volume. The entered number is placed in the EOV1 label and in the VOL1 volume header label of the next scratch tape that is mounted. The serial number is also used for tape ready messages.

Multi-Volume Input Processing

When an end-of-volume label (EOV1) is read during input processing, a check is made for a next volume serial number. If one is present, it is used in tape ready messages and in volume label processing to ensure that the correct reel is mounted. If a next volume serial number is not present, end-of-volume processing continues.

UNLABELED UFF TAPE FILES

Unlabeled UFF tape files conform to one of the following arrangements:

- Single-file single-volume, single-file multi-volume, or multi-file single-volume
- Optional block serial numbers (BSN) and record control words (RCW) (not supported for files opened for extend)
- The open EXTEND option only for single-file, single or multi-volume arrangements
- The concept of a NULL file for UFF unlabeled not supported
- Segmented (spanned) records if RCWs are present

UNLABELED UFF FILE PROCESSING

The various file arrangements available with Unlabeled UFF tape are illustrated in figure 1-19:

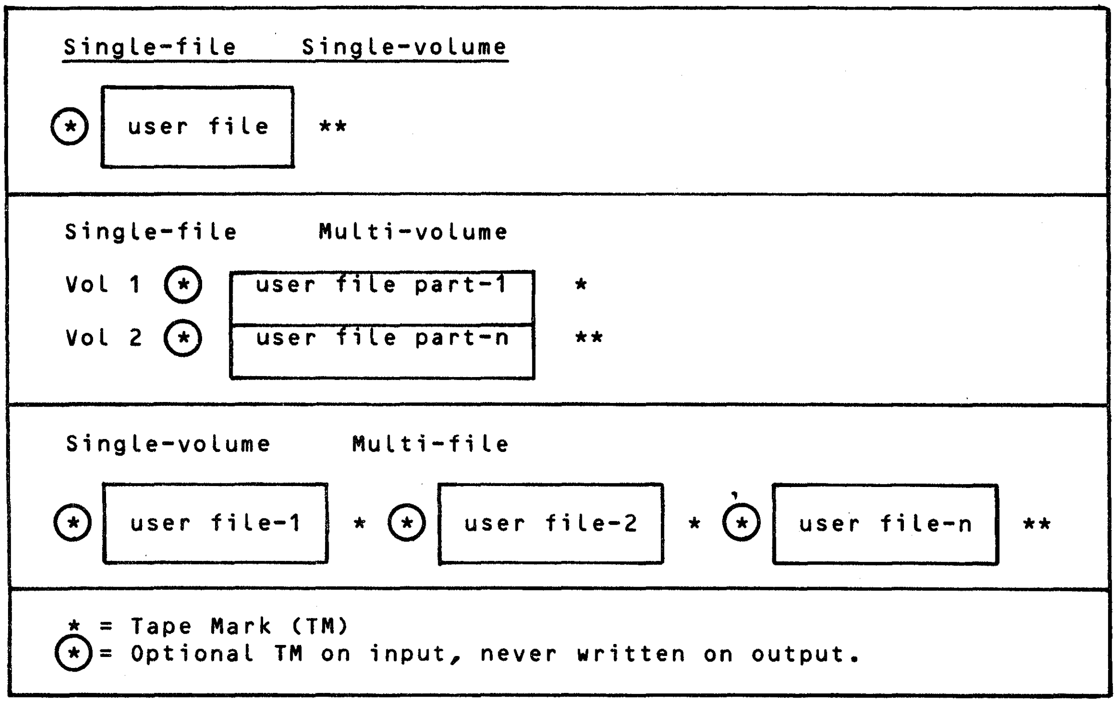

Single-file Single-volume
(*) user file **
Single-file Multi-volume
Vol 1 (*) user file part-1 *
Vol 2 (*) user file part-n **
Single-volume Multi-file
(*) user file-1 * (*) user file-2 * (*) user file-n **
* = Tape Mark (TM)
(*) = Optional TM on input, never written on output.

Figure 1-19. Unlabeled UFF File Arrangements

Single File Input Processing

When a tape mark is encountered on an unlabeled input single file volume, a subsequent read request is interpreted as a request for volume switching to take place. It is the user's responsibility to determine the true end of the data, for example, by placing a dummy EOF record at the end of the data.

Multi-File Input Processing

When a tape mark is encountered on an unlabeled input multi-file volume, a subsequent read request is interpreted as a request for the first data block following the mark. It is the user's responsibility to determine the true end of the data. If an attempt is made to read beyond the tape mark of the last file of the volume, an erroneous tape condition can result.

On an unlabeled multi-file volume, the UFAS programmer may specify a file sequence number. At file open time, UFAS positions the current volume to the beginning of that file.

Output Processing

For unlabeled output files, the detection of the end of tape (EOT) foil causes a tape mark to be written and unit switching to occur.

Extend Processing

Single file single-volume and single file multi-volume files may be extended. UFAS opens the file to a position that permits logical records to be appended to the current file. If the file is multi-volume, it is the user's responsibility to ensure that the current volume is the final volume, i.e., the volume to which data is to be appended.

System Input and Output Files

System Input and Output files cannot be opened for either the extend or input/output processing modes.

PHYSICAL ATTRIBUTES

Block serial number (BSN)

Variable-length records (Each logical record is preceded by a record control word - RCW.)

Binary recording mode

Character set - Source or destination device dependent

Maximum block size - 320 words

No labels

The report code identifies a record as belonging to a specific report or card deck. A report code that is unique for each file code is automatically generated. If the user does not specify a report code the default report code is assumed.

Logical record octal media code is used by media conversion to determine the specific action required for each record when the file is assigned to SYSOUT by means of the $ SYSOUT or the $ REMOTE card. The following media codes are assigned:

00 - Not a media conversion record, or no printer slew controls
01 - Binary card image
02 - BCD card image
03 - BCD print line image
04 - Reserved for user
05 - Time sharing system ASCII[1]
06 - GFRC format ASCII
07 - ASCII print line image
10 - Time sharing system information record
11 - BCD print line image
12 - ASCII card image
15 - ASCII print line image

NOTE: Media codes 11 and 15 indicate that a user defined report code was inserted in the first two character positions of the logical record. Appropriate ignore characters are entered in these character positions before output is released to the printer.

GFRC File Format

UFAS processes GFRC files on 7- and 9-track tapes or linked mass storage (mass storage allocated as linked). The linked mass storage files have the same physical attributes as the System Input and Output files.

[1] Not supported by UFAS

PHYSICAL ATTRIBUTES (Tape Files)

7- or 9-track tape

Block serial number - Optional

Variable- or fixed-length records (If variable length, each logical record is preceded by a RCW.)

Binary or BCD recording mode

Labels optional - Recorded in binary

GENERAL FORMAT (Labeled Tape Files)

Header Label Group	Beginning Tape Label	-BTL	Required
	Tape Mark	-TM	
File Data	Data Block 1		Optional
	Data Block n		
Trailer Label Group	Tape Mark	-TM	Required
	Trailer Label	-EOF	
	Tape Mark	-TM	
	Partial Tape Label	-PTL	

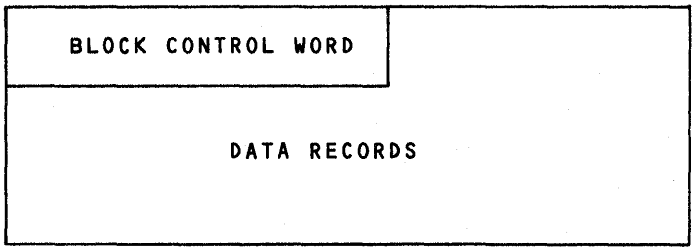
DATA FORMAT

The file space on tape or mass storage contains blocks of data and control information. Each block is composed of a block control word and data records. The block control word is optional on tape files. If present, it is the first word of each block and contains the following binary values:

bits 0-17 Block serial number - The sequential number of the block (within the current file or within the current reel) if it is a multireel file.

bits 18-35 Block size - Size of the block, in words, including the block control word. Maximum size of a block is 4095 words on tape and 320 words on mass storage.

The following diagram illustrates a block with a block control word.



Variable-Length Records

If the logical records contained within each block are of variable length, each record has a record control word as the first word. The contents of the record control word are:

bits 0-17 Binary record size in words, excluding the record control word. When a file is assigned to a mass storage device, this value is zero if the word is interpreted as an end-of-file (EOF) mark. In this instance bits 20-23 contain the end-of-file mark 1111.

bits 18-19 Next available 9-bit character position in last word. Field is interpreted as (only with media code 6):

00 - Full word used

01 - One character used

10 - Two characters used

11 - Three characters used

bits 20-23 Only used when bits 0-17 are zero. When this occurs, these bits contain end-of-file mark 1111.

bits 24-25 Zeros

bits 26-29 Logical record octal media code is used by media conversion to determine the specific action required for each record when the file is assigned to SYSOUT by means of the $ SYSOUT or the $ REMOTE card. The following octal media codes are assigned:

00 - Not a media conversion record, or no printer slew controls

01 - Binary card image

02 - BCD card image

03 - BCD print line image

04 - Reserved for user

05 - Time sharing system ASCII[1]

06 - GFRC format ASCII

07 - ASCII print line image

10 - Time sharing system information record

11 - BCD print line image

12 - ASCII card image

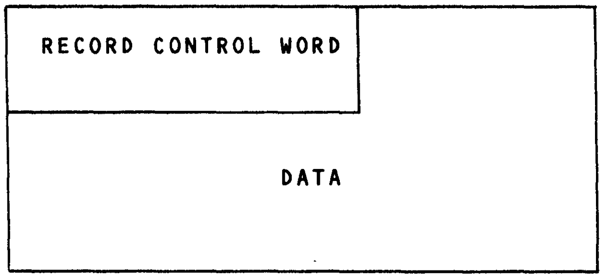
15 - ASCII print line image

bits 30-35 Report code identifying this record as one belonging to a specific report or deck.

NOTE: Media codes 11 and 15 indicate that a user defined report code was inserted in the first two character positions of the logical record. Appropriate ignore characters are entered in these character positions before output is released to the printer.

[1] Not supported by UFAS

The following diagram depicts a logical record with a record control word.



Partitioned Records

A partitioned record is a variable-length logical record that is larger than the block size being used. It is, therefore, broken up or partitioned over two or more physical records. A partitioned record always starts at the beginning of a block and the end of a partitioned record always ends a block. A variable-length record file may contain both partitioned and non-partitioned records. Partitioned records can reside only on files with media codes 0 or 6.

For partitioned record files, there are two or more record control words (one for each segment of the partitioned record). The contents of the first record control word for a partitioned record area are:

bits 0-17 - Record size in this block
bits 18-23 - Not used
bits 24-25 - First segment code (01)
bits 26-29 - Media code (0 or 6)
bits 30-35 - Report code

If the next segment of the partitioned record is not the last segment, then the record control word contents for the segment are:

bits 0-17 - Record size in this block
bits 18-23 - Not used
bits 24-25 - Intermediate segment code (10)
bits 26-35 - Segment number

The contents of the record control word for the last segment of a partitioned record are:

bits 0-17 - Record size in this block
bits 18-19 - Next available 9-bit character position in last word (media code 6 only)
bits 20-23 - Not used
bits 24-25 - Last segment code (11)
bits 26-35 - Segment number

PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING

User programs can create ASCII or BCD print-line report files to be used as input to the Page Printing System (PPS).

Print-line reports containing ASCII data require a $ DATA .U parameter card at execution time. The format of the parameter card is shown in the "Control Cards" subsection of Appendix A. This option is restricted to an output file with these attributes:

- GFRC Sequential organization
- Octal media code 7 or 15
- Directed to 9-track tape

Print-line reports containing BCD data do not require $ DATA .U processing by UFAS to be used as input to the PPS.

PPS tape formats can be in any of these arrangements:

- single-file, single-volume
- single-file, multi-volume
- multi-file, single-volume
- multi-file, multi-volume

Control Word Format

ASCII data written to a PPS print-line report contain a Block Control Word (BCW) with the format shown in Figure 1-19.1.

Bits	Contents
0-1	must be zero
2-8	upper 7 bits of Block Serial Number (binary)
9-10	must be zero
11-17	lower 7 bits of Block Serial Number (binary)
18-19	must be zero
20-26	upper 7 bits of block size (binary)
27-28	must be zero
29-35	lower 7 bits of block size (binary)

Figure 1-19.1 PPS BLock Control Word

Records written to a PPS tape must be defined as variable length. The Record Control Word (RCW) should contain the information shown in Figure 1-19.2.

Bits	Contents
0-1	must be zero
2-3	partitioned record segment code (unused)
4-8	upper 5 bits of record size (binary)
9-10	must be zero
11-17	lower 7 bits of record size (binary)
18-20	must be zero
21-22	next 9 bit character position (unused)
23-26	media code
27-29	must be zero
30-35	report code

Figure 1-19.2 PPS Record Control Word

UFAS checks BCW and RCWs to ensure that all fields contain valid formats. After the validity of the control words is ensured, UFAS writes the data with a Write Tape 9 command. This requires that the data consists of exclusively ASCII characters. Any data that contains binary items may cause the write command to fail and the job to abort.

The GFRC tape file format conforms to one of the following arrangements: single-file single-volume, single-file multi-volume, multifile single-volume, or multifile multi-volume.

A volume may contain a portion of a file, a complete file, or more than one file. The volume or volumes on which a file resides is considered a volume set.

The GFRC file format contains the following labels:

1. Beginning of tape label (BTL) header label
2. End-of-reel (EOR) or end-of-file (EOF) trailer label
3. Partial tape label (PTL)

The tape mark is octal 17 for a 7-track tape and octal 23 for a 9-track tape. All GFRC labels written by UFAS are 20 words long and are recorded in binary.

If a file is multi-volume, item 2 is an EOR for every volume but the last one, on which it is an EOF label. If a volume contains more than one file, items 1 and 2 are repeated for each file. Partial labels are written as end-of-information banners on a volume.

The various file arrangements available with this format are illustrated in Figure 1-20.

Single-File Single-Volume

BTL*-----File A----*EOF*PTL

Single-File Multi-Volume

BTL*----First Section of File A----*EOR*
BTL*----Last Section of File A----*EOF*PTL

Multifile Single-Volume

BTL*----File A----*EOF*BTL*----File B----*EOF*PTL

Multifile Multi-Volume (Only with UFAS)

BTL*----File A----*EOF*BTL*----First Section of File B----*EOR*
BTL*----Last Section of File B----*EOF*BTL*----File C----*EOF*PTL

NOTE: Asterisk(*) represents tape mark

Figure 1-20. GFRC Tape File Format

The name and characteristics of the label fields for the Beginning of Tape Label, End-of-File or End-of-Reel Label, and Partial Tape Label are listed in Figures 1-21 through 1-23 followed by a description of each item.

Word	Field Name	Content
1-2	Label Identifier	GEbb600bBTLb
3	Installation Identification	6 alphanumeric characters
4	Tape Reel Serial Number	5 alphanumeric characters right justified
5	File Serial Number	5 alphanumeric characters right justified
6	Reel Sequence Number	4 numeric characters right justified
7	Creation Date	2 characters (yy) for the year followed by 3 characters (ddd) for the day of the year, right justified
8	UFAS Identifier Retention Period	3 alphanumeric characters - Dbb 3 numeric characters
9-10	File Name	12 alphanumeric characters
11-13	User Words	(Arbitrary)
14	Reel Number in Preceding EOR is Erroneous	bbbbbb or PRVERR
15	File Sequence Number	Binary
16	Header Sequence Number	Binary
17	Retention Date	Binary year and day of year
18	Tape Density	Binary
19-20	Reserved for System	

Figure 1-21. GFRC Beginning of Tape Label (GFRC BTL)

Beginning of Tape Label (BTL)

LABEL IDENTIFIER: On input volumes, UFAS expects a label with the identifier GEbb600bBTLb. Any other label is considered an error and an operator message is generated. The operator can accept the reel as valid, mount the correct reel to be processed, or abort the job. On output, the label identifier, GEbb600BTLb, appears in the first two words of the label.

When a file is opened for input and at the beginning of each subsequent reel of a multireel file, the label is checked to ensure that the correct reel is being processed. The following conditions must be met for an input label to be valid:

1. The file serial number taken from the $ TAPE card must agree with the file serial number in the label. This test is bypassed if the $ TAPE file control card contains the file serial number 99999.

2. The reel sequence number taken from the $ TAPE card (default of 1) must agree with that contained in the label. When reel switching occurs, the reel sequence number is incremented.

3. The file name supplied by the user in the FIB macro must agree with the file name contained in the label. If no file name is supplied, this test is bypassed.

4. The reel serial number in the label must agree with the reel serial number in a previous EOR label. This test is bypassed if the literal PRVERR appears in the label.

If these tests are passed, a message is written to the user's execution report indicating tape reel serial number, file code, file sequence number, and reel sequence number.

If any of the preceding tests fail, an operator message is generated. This message includes the SNUMB, unit code, and all necessary information from the label. The operator may then either accept the reel as valid, accept this reel and all subsequent reels as valid, mount the correct reel to be processed, or abort the job.

When an output file is opened and after reel switching, the beginning of tape label is read. This may be a blank reel label or a header label on which the retention period has expired. (Retention period = date created + retention days.) If these conditions are not met, a message is typed to the operator indicating the invalid condition. The operator may accept the reel as valid by entering the correct reel serial number, removing the reel and mounting a new one, or aborting the job.

Acceptance of a reel after an invalid condition and if this is not the first reel, the reel number provided by previous EOR trailer label processing is compared with the reel serial number of the current reel. If the two numbers do not match, a message is written to the operator who can continue execution, dismount the tape and mount the correct one, or abort the job.

After acceptance of a reel for output, a message is written to the user's execution report indicating tape reel serial number, file code, file sequence number, and reel sequence number.

INSTALLATION IDENTIFICATION: Constant information for a given user installation. This field is ignored by UFAS label checking routines.

TAPE REEL SERIAL NUMBER: The serial number of the physical tape reel. This number is marked externally on the reel itself.

FILE SERIAL NUMBER: The tape reel serial number of the first reel of the file.

REEL SEQUENCE NUMBER: The number of the reel within a given file. For example, the first reel of a file is reel 0001, the second is 0002, etc.

CREATION DATE: Year and day of year that the tape was created; recorded in BCD.

UFAS IDENTIFIER: The first character is a D for files written by UFAS. This character is a blank in GFRC-created labels.

RETENTION PERIOD: Three BCD characters specifying the number of days that the tape must be kept before being erased (see Retention Date).

FILE NAME: Twelve BCD characters, left justified, giving the user-defined name of this file.

USER WORDS: These words are available for the user program and are bypassed by UFAS label checking.

REEL NUMBER ERRONEOUS: On output, the reel number provided by the previous EOR trailer label processing is compared with the reel serial number of the current reel. If the two numbers do not match, a message is generated and the operator can continue execution, dismount the tape and mount the correct one, or abort the job. If the operator continues by ignoring the error, a literal PRVERR is placed in the header label. On input, the reel serial number in the label must agree with the reel serial number in the previous EOR label. This test is bypassed if the literal PRVERR appears in this field.

FILE SEQUENCE NUMBER: For multifile volumes, this binary field is used in positioning to the correct file.

HEADER SEQUENCE NUMBER: For each file of a multifile set that extends across several reels, this binary field is 1 for the first reel of the file and is incremented on successive reels.

RETENTION DATE: Date in binary when this tape may be reused. This field is ignored on input. On output the existing date is compared to the current date. If the retention date is less, a message is issued and the operator can ignore the error or abort the job. The new retention date is calculated by adding the retention period to today's date and placing it in the label.

TAPE DENSITY: Recording density for this tape in bits per inch (bpi):

1	-	200 bpi
2	-	556 bpi
4	-	800 bpi
9	-	1600 bpi
12	-	6250 bpi

The bits are right justified in the word. This field is set on output tape processing. On input, the specified density is used to read the tape. If this density conflicts with that specified on the $ TAPE card or the default density, a message is written on the execution report.

RESERVED: Reserved for System use.

Word	Field Name	Content
1	Label Identifier	ƀEORƀƀ or ƀEOFƀƀ
2	Block Count	Binary or BCD (6 numeric characters)
3-12	User Words	(Arbitrary)
13	UFAS Identifier	3 alphanumeric characters - Dƀƀ
14	Reel Serial Number of Next reel	6 characters
	or	
	All blanks	ƀƀƀƀƀƀ
15-20	Reserved for System	

Figure 1-22. GFRC End-of-Reel/File Label (EOR/EOF)

End-of-Reel/File Label (EOR/EOF)

LABEL IDENTIFIER: This field identifies the label as an end-of-file label or end-of-reel label, "ƀEOFƀƀ" or "ƀEORƀƀ", respectively.

On input, the occurrence of an EOR label causes tape switching to the next reel. The block count which appears in each trailer label is checked and if the count does not agree with the block count maintained by UFAS an error message is given to the operator, who can then either continue execution, force end-of-file, or abort the job. After the block count is checked, an EOR label is checked to determine if it contains the number of the next reel of this file. If the next reel serial number is present, it is used in the next input file header label check.

When the end-of-tape foil is encountered during the process of writing on an output reel, a tape mark is written on the tape. A message is issued to the operator requesting the reel serial number of the next sequential reel. The operator then types in the reel number as it appears on the tape reel. A message is then issued to the operator requesting verification of the reel serial number that was previously entered. After verification, this number is placed in a message that is written on the user's execution report. The number is also placed into the EOR label to be written on the current reel. The output trailer label when written also includes the block count which indicates the number of data blocks recorded on the file. When the new reel is provided, processing is continued.

BLOCK COUNT: Block count contained in the trailer label is the number of data blocks contained on this reel of this file. In the case of a multifile reel, the block count is the number of data blocks for the current file only. This number is recorded as a 36-bit binary number or a 6-character BCD number depending upon the recording mode of the file for the 14-word (GFRC-created) labels. A 20-word (UFAS-created) trailer label always contains a binary number.

USER WORDS: These words are available for the user program and are bypassed by UFAS label checking.

UFAS IDENTIFIER: The first character is a D if the file was written by UFAS. This character is a blank in GFRC-created labels.

REEL SERIAL NUMBER: For a multi-volume set, this field gives the reel serial number of the next tape in sequence and is used on input header checking to verify the reel number.

Word	Field Name	Content
1-4	Same as Beginning of Tape Label	(see Figure 1-14)
5-10	All Zeros	
11-14	Not used	
15-16	Same as Beginning of Tape Label	(see Figure 1-14)
17	Reel Sequence Number	(see Figure 1-14)
18-20	All Zeros	

Figure 1-23. GFRC Partial Tape Label (PTL)

PARTIAL TAPE LABEL (PTL)

The corresponding fields are processed as described for the beginning of tape label.

Label Exits

Four label exit points are provided to the user during label processing. These four points are:

- Before Beginning File Label
- After Beginning File Label
- Before Ending File Label
- After Ending File Label

The user can request control for any combination of the exit points. During label processing, control is transferred to the user after UFAS prepares a label image. Then, the user may process the label; following completion of the processing, control must be returned to UFAS.

User label processing can include performing any UFAS function on any file other than the one for which the exit is specified. If the function causes a label exit to occur, control must be returned to UFAS in the reverse sequence of events. For example, label exit processing for file A includes opening file B for which there is a beginning label exit. In such a case, file B label processing must conclude with a return to UFAS at the file B return point and file A label processing with a return at the file A return point.

UNLABELED GFRC TAPE FILES

Unlabeled GFRC tape files conform to one of the following:

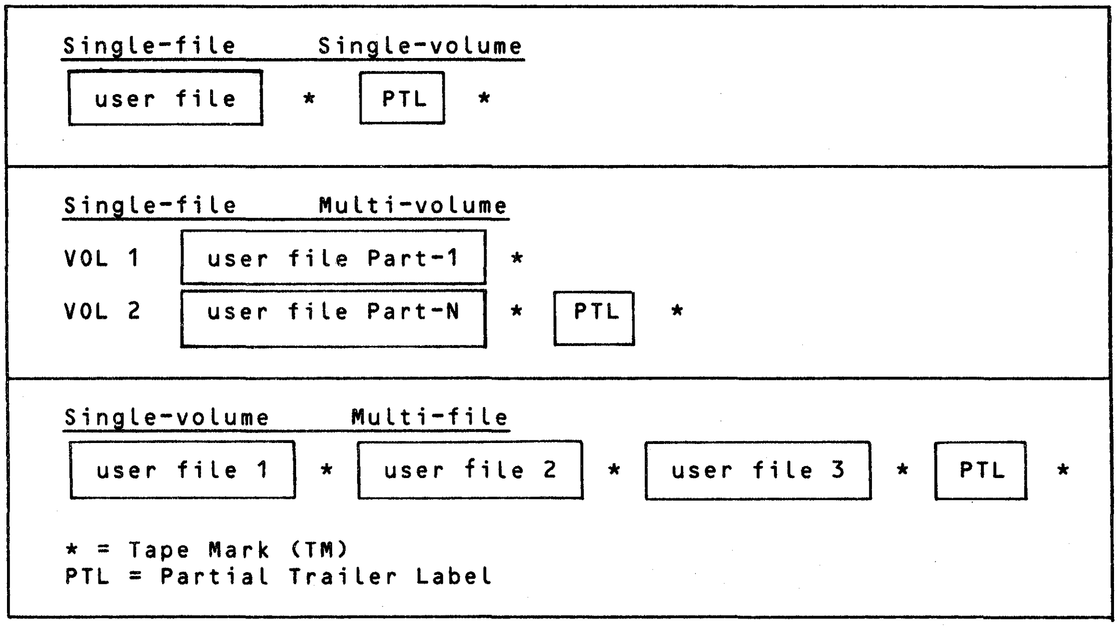
- single-file single-volume
- single-file multi-volume
- multi-file single volume

Block serial numbers (BSN) and record control words (RCW) are optional.

The EXTEND Processing mode is not available for unlabeled GFRC files.

Records may be segmented (spanned); if so, RCWs must be present.

The various file arrangements available with Unlabeled GFRC tape are illustrated in figure 1-24:



* = Tape Mark (TM)
PTL = Partial Trailer Label

Figure 1-24. Unlabeled GFRC Tape Arrangements

UNLABELED GFRC FILE PROCESSING

Single File Input Processing

When a tape mark is encountered on an unlabeled input single file volume, a subsequent read request is interpreted as a request for volume switching to take place. It is the user's responsibility to determine the true end of the data, for example, by placing a dummy EOF record at the end of the data.

Multi-File Input Processing

When a Tape mark is encountered on an unlabeled input multi-file volume, a subsequent read request is interpreted as a request for the first data block following the mark. It is the users responsibility to determine the true end of the data. If an attempt is made to read beyond the tape mark of the last file of the volume, an erroneous tape condition can result.

On an unlabeled multi-file volume, the UFAS programmer may specify a file sequence number. At file open time, UFAS positions the current volume to the beginning of that file.

Output Processing

For unlabeled output files, the detection of the EOT foil causes a tape mark to be written and unit switching to occur.

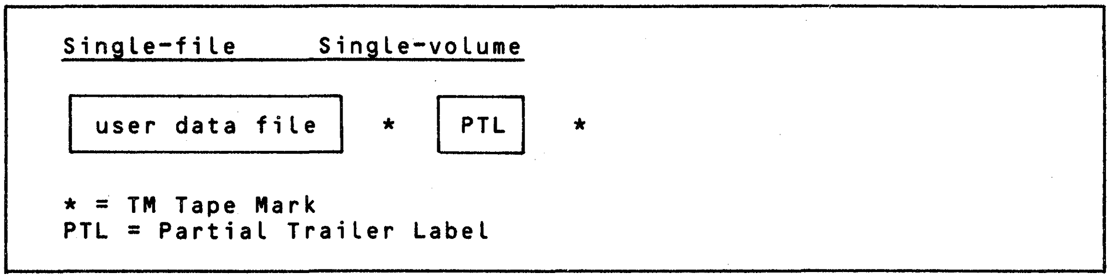
UNLT TAPE FILE FORMAT

- UFAS processes UNLT unlabeled files on 7- and 9-track tapes.
- The UNLT tape file format is reserved for a single-file, single-volume arrangement.
- The files data blocks are automatically blocked one. Block Serial Number (BSN) and Record Control Words (RCW) are not applicable.
- The tape mark is Octal 17 for 7-track and Octal 23 for 9-track tapes.

Physical Attributes

- 7- or 9-track tape
- blocking factor of one
- FLR only
- no BSN
- no labels
- no RCW
- Binary recording mode

The file arrangement available with the format is illustrated in figure 1-25:



* = TM Tape Mark
PTL = Partial Trailer Label

Figure 1-25. UNLT Tape File Arrangement

UNLABELED UNLT FILE PROCESSING

Single File Input Processing

It is the user's responsibility to determine the true end of the data, for example, by placing a dummy End Of File Record at the end of the data. If an attempt is made to read beyond the End Of Tape mark (foil) an erroneous tape condition can result.

IBM Tape File Format

The file and label formats as well as the processing for IBM tape files are identical with those for the American National Standard files, except as noted.

PHYSICAL ATTRIBUTES

- 9-track tape
- Variable- or fixed-length records (If variable length, each logical record is preceded by a record control word - record length is in binary.)
- Byte recording mode (Write and read tape 9 - four 8-bit bytes)
- Labels - (optional, contain EBCDIC characters)

GENERAL FORMAT - LABELED TAPES

```
                       Volume    VOL1                Required
Header                           HDR1                Required
Label      Header Labels         HDR2
Group                            HDR3                Optional
                                 HDRn
              User Labels        UHL1
                                 UHL8                Optional
                Tape Mark        TM                  Required
File                             Data Block 1
Data                             Data Block n        Optional
                Tape Mark        TM                  Required
                                 EOF1                Required
Trailer   Trailer Labels         EOF2
Label                            EOF3                Optional
Group                            EOFn
                                 UTL1
              User Labels        UTL8                Optional
                Tape Mark        TM                  Required
                Tape Mark        TM                  Required
```

GENERAL FORMAT - UNLABELED TAPES

```
 (TM)                Optional
 Data Block 1
      .
      .              File 1
      .
 Data Block n
 TM                  Required

 (TM)                Optional
 Data Block 1
      .
      .              File 2
      .
 Data Block n
 TM                  Required

 (TM)                Optional
 .
 .
 .
 .
 TM                  Required
```

TM = required tape mark
(TM) = optional tape mark

DATA FORMAT

A file may contain either fixed-length or variable-length records. In a file with variable-length records, the first four bytes at the beginning of the block contain the block-length field and the first four bytes of each record contain the record size. Both the block length and the record size are expressed in binary in the first two bytes and zeros in the remaining two.

Records can be segmented (spanned). The binary segment control code is the third byte of the record size field. The segment control values are as follows:

0 - segment contained entirely within this block
1 - first segment
2 - last segment
3 - intermediate segment

LABEL FORMATS AND PROCESSING

Unlike the American National Standard file volumes, all labeled volumes in IBM files have only one tape mark at the end, except the last volume which has two tape marks. On output, the second file-header label (HDR2) and the second end-of-volume label (EOV2) or the second end-of-file (EOF2) label are required labels on IBM files, but optional on American National Standard files.

No optional user-volume labels (UVLn) are permitted. Only file user-header and user-trailer labels (UHL1/UTL1 - UHL8/UTL8) are allowed.

IBM Volume-Header Label (VOL1)

Character Position	Field Name	Contents
11	Accessibility	Must be zero
42-51	Owner Identifier	10 alphanumeric characters
80	No Version Number	Spaces

IBM First File-Header Label (HDR1)

Character Position	Field Name	Contents
54	Accessibility	Zero means no restriction

IBM Second File-Header Label (HDR2)

Character Position	Field Name	Contents
5	Record Format	F - Fixed length V - Variable length U - Undefined
11-15	Record Length	Character count

NOTE: The content of the second file-header label (HDR2) record length field is record length for fixed-length records and maximum record length plus 4 (RCW) for variable-length records.

IBM Second File-Header/End-of-File (HDR2/EOF2)

Character Position	Field Name	Contents
16	Reserved	
17	File Position	0 - first volume of file 1 - not first volume of multi-volume
18-34	Job per Job Identifier	First 8 bytes - job name 9th byte - 1 Last 8 bytes - job step
35-36	Tape Recording Technique	9-track tape - blanks
37	Printer Control Characters	A - American National Standard M - Machine b̸ - None
38	Reserved	Spaces
39	Block Attribute (Data set format)	B - Blocked S - Spanned R - Blocked and spanned b̸ - Neither blocked nor spanned
40-80	Reserved	Spaces

UFAS UNLABELED IBM TAPE PROCESSING

The UFAS user can now read and write certain forms of unlabeled IBM-type magnetic tape files. The following three forms of IBM unlabeled tape files are supported.

- Single-volume/Single-file
- Single-volume/Multifile
- Multivolume/Single-file

File Formats

In the following formats, the beginning tape mark, (TM), is optional on input and output; while the End-of-File tape mark, TM, is required.

- SINGLE-VOLUME/SINGLE-FILE FORMAT

  (TM) User File TM

- SINGLE-VOLUME/MULTIFILE FORMAT

  (TM) User File - 1 TM (TM) User file - 2 TM (TM) User file - n TM

- MULTI-VOLUME/SINGLE-FILE FORMAT

  Volume 1 - (TM) User File - 1, Part 1 TM

  Volume n - (TM) User File - 1, Part n TM

Unlabeled tapes contain no volume labels, header labels, or trailer labels. They contain only data blocks and tape marks.

Unlabeled File Processing Procedure

SINGLE-FILE INPUT PROCESSING

When a tape mark is encountered on an input single-file volume, the return from the Read Command is always to the EOF position specified. The user must determine the true end of the data file. A subsequent Read indicates that unit switching is required.

MULTI-FILE INPUT PROCESSING

When a tape mark is encountered on an unlabeled input multi-file volume, a subsequent Read request is interpreted as a request for the first data block following the tape mark. The user must determine the true end of the data file. If an attempt is made to read beyond the tape mark of the last file of a volume, an erroneous tape condition can result.

On an unlabeled multi-file volume, a user can specify a file sequence number. When a file is opened, UFAS positions the current volume at the beginning of the file. The requested file must be on the current volume.

OUTPUT PROCESSING

For unlabeled output files, detection of the end-of-tape foil causes a tape mark to be written and unit switching to occur.

Unlabeled File Data Format

The tapes are nine-track, with four 8-bit bytes per word. They are processed with Tape-9 Read and Write commands.

A file can contain either fixed-length or variable-length records. In a file with variable-length records, the first four bytes at the beginning of the block contain the block-length field; while the first four bytes of each record contain the record length. Both the block length and the record length are expressed in binary in the first two bytes, with zeros in the remaining two. Each length tally includes the four-byte length field. The variable-length record layout is as follows:

```
Bytes- 0 1 2 3 4 5 6 7
       B B 0 0 R R 0 0  User Record - 1    R R 0 0  User

       Record - 2           0 0  User Record - n

       Where:  BB - Block length tally (binary)
               RR - Record length tally (binary)
```

Variable-length records can be segmented (spanned). The third byte in the record length field contains the segment control code (binary). The segment control values are as follows:

0 - Segment contained entirely within this block
1 - First segment
2 - Last segment
3 - Intermediate segment

A file that contains fixed-length records does not have either block or record control bytes. The user must specify the record size, and records cannot be segmented.

UFAS Open Procedures

SINGLE-FILE INPUT OPEN PROCEDURE

In opening a single-volume single-file or multivolume single-file file for input, UFAS positions to the beginning of the file on the current volume. Although beginning tape marks are optional, if two tape marks are encountered before a data block, UFAS sets an EOF indicator. A subsequent Read request will then cause UFAS to transfer to the user's EOF address. If the file is a multivolume file, the user must make sure that the current volume is the first volume.

MULTIFILE INPUT OPEN PROCEDURE

In opening a multifile file for input, UFAS positions itself to the specified file on the current volume. The user must make sure that the requested file is on the current volume. Files can be delimited by either one or two tape marks. UFAS proceeds down the volume, counting files until the specified file is reached. The first file on a volume is considered file number 1.

OUTPUT OPEN PROCEDURE

UFAS writes a beginning tape mark for any unlabeled IBM tape file that is opened for output, unless the "TPMARK/NO/" option is specified for the file in the .U data file.

EXTEND OPEN PROCEDURE

In opening a single-volume or multi-volume single-file file for extend, UFAS opens the file to a position that permits logical records to be appended to the current file. If the file is multi-volume, the user must make sure that the current volume is the final volume, i.e., the volume upon which data is to be appended.

A file resident on a multi-file volume cannot be opened for extend.

H2000 Tape File Format

UFAS provides file and content management for H2000 MOD1 and MOD4 file formats on 7- and 9-track tapes.

H2000/60 Magnetic Tape Interchange Facility

The 2000/60 Magnetic Tape Interchange Facility is a combination of hardware and software components that enable the Series 60 Level 66 user to read and write Series 2000 magnetic tape files.

Tapes to be read or written must be mounted on tape drives controlled by an MPC tape controller equipped with Magnetic Tape Interchange Facility. This facility accommodates both GCOS and Series 2000 files.

User programs that process H2000 with UFAS must allocate tape units with either $ TAPE27 or $ TAPE29 control cards.

PHYSICAL ATTRIBUTES

- 7- or 9-track tape
- Variable- or fixed-length records
- Labels if present - Recorded in binary for MOD1
  - Recorded in BCD for MOD4
  - 80 characters (input and output)
  - 120 characters (input only)
- Data blocks - Recorded in binary
- Character set - HBCD
- Density - 200, 556, 800, 1600 bpi (no 6250 bpi)

GENERAL FORMAT (Labeled Tape Files)

	Single-Volume File	Multi-Volume File	
Header Label Group	1HDRƀ (Follows 120-character label if MOD4)	1HDRƀ (If MOD4)	1HDRƀ
File Data	Data Block	Data Block	Data Block
Trailer Label Group	1EOFƀ 1ERIƀ 1ERIƀ	1EOFƀ 1ERIƀ 1ERIƀ	1EOFƀ 1ERIƀ 1ERIƀ

DATA FORMAT

A file may contain fixed-length or variable-length records, which may be unblocked or blocked, according to format type. Only fixed-length records can be blocked. Padding is used to provide an integral number of records for a physical block that must have an exact number of records. A padding character can have any 6-bit configuration, except zeros. On both input and output a padded record is composed only of pad characters. On output, padded records are generated to fill a physical block when the Close function is invoked. Padded records are not delivered to the user program. For the MOD1 file format, a banner character may be used with each physical block to distinguish data blocks, label blocks, and non-data blocks to be ignored (such as checkpoint information).

The file format types supported by UFAS are the following:

MOD1

AF - Unblocked fixed-length records
AV - Unblocked variable-length records (without Record or Item character counts)
B - Blocked fixed-length records (padded records allowed)

MOD4

1F - Unblocked fixed-length records
1V - Unblocked variable-length records (without Record or Item character counts)
2 - Blocked fixed-length records (padded records allowed)

LABEL FORMATS AND PROCESSING

The H2000 tape file formats can reside only on a single-file single-volume or on a single-file multi-volume file.

The name and characteristics of the label fields for the Beginning of Tape Label, End-of-File or End-of-Reel Label, and End-of-File or End-of-Reel Label Structure are listed in Figures 1-26, 1-27, 1-28, 1-29, and 1-30.

Character Position	Field Name	Contents
1-5	Label Identifier	1HDRƀ
6-10	Tape Reel Serial Number or Volume Identifier	5 HBCD characters
11-16	File Serial Number	5 HBCD characters followed by minus sign "-"
17-20	Reel Sequence Number	3 numeric characters followed by blank space
21-30	File Name	10 alphanumeric characters, blank filled, left justified
31-35	Creation Date	2 characters (yy) for the year followed by 3 characters (ddd) for the day of the year
36-40	Retention Period	3 HBCD characters preceded by minus sign and followed by blank space where ccc characters are: 000 ccc 999
41-80	Reserved for System	

Figure 1-26. H2000 Beginning of Tape Label (BTL) - 80 characters

Character Position	Field Name	Contents
1-6	Label Identifier	1HDRƀƀ
7-10	Retention Period	Zero followed by 3 numeric characters
11-15	Creation Date	2 characters (yy) for the year followed by 3 characters (ddd) for the day of the year
16-25	File Name	10 alphanumeric characters
26-30	File Serial Number	5 HBCD characters
31-36	Tape Reel Serial Number or Volume Identifier	5 HBCD characters followed by blank space
37-40	Reel Sequence Number	Zero followed by 3 numeric characters
41-120	Ignored During Header Processing	

Figure 1-27. H2000 Beginning of Tape Label (BTL) - 120 Characters

LABEL IDENTIFIER: On input volumes, if the H2000 tape is declared to be labeled, UFAS expects a label with an identifier. Absence of a label is considered an error and an operator message is generated. The operator can accept the reel as valid, mount the correct reel to be processed, or abort the job.

When a file is opened for input and at the beginning of each subsequent reel of a multireel file, the label is checked to ensure that the correct reel is being processed. The following conditions must be met for an input label to be valid:

1. The file serial number taken from the $ TAPE card must agree with the file serial number in the label. This test is bypassed if the $ TAPE file control card contains the file serial number 99999.

2. The reel sequence number taken from the $ TAPE card (default 1) must agree with that contained in the label. When reel switching occurs, the reel sequence number is incremented.

3. The file name supplied by the user in the FIB macro must agree with the file name contained in the label. If no file name is supplied this test is bypassed.

If these tests are passed, a message is written to the user's execution report indicating tape reel serial number, file code, file sequence number, and reel sequence number.

If any of the preceding tests fail, an operator message is generated. The operator may then either accept the reel as valid, accept this reel and all subsequent reels as valid, mount the correct reel to be processed, or abort the job.

When an output file is opened and after reel switching, which is automatic, the beginning of tape label is read. This may be a blank reel label or a header label on which the retention period has expired. (Retention period = date created + retention days.) If these conditions are not met, a message is typed to the operator indicating the invalid condition. The operator may accept the reel as valid by entering the correct reel serial number, removing the reel and mounting a new one, aborting the job, or ignoring the retention period.

TAPE REEL SERIAL NUMBER: The serial number or volume identification of the physical tape reel. This number is marked externally on the reel itself.

FILE SERIAL NUMBER: The tape reel serial number of the first reel of the file.

REEL SEQUENCE NUMBER: The number of the reel within a given file. For example, the first reel of a file is reel 001, the second is 002, etc.

FILE NAME: Ten BCD characters, left justified, giving the user-defined name of this file.

CREATION DATE: Year and day of year that the tape was created.

RETENTION PERIOD: Number of days before this tape may be reused. This field is ignored on input. On output, the retention date is compared to the current date. If the retention date is greater than current date, a message is issued and the operator can ignore the error or abort the job. The retention date is calculated by adding the retention period to today's date.

RESERVED: Reserved for System use.

Character Position	Field Name	Contents
1-5	Label Identifier	1EOFƀ or 1EORƀ
6-10	Block Count	5 numeric characters
11-20	Record Count	10 numeric characters
21-30	Hash Total or File Name	(Arbitrary)
31-80	Reserved for System	

Figure 1-28. End-of-File/Reel Label (EOF/EOR) - 80 Characters

Character Position	Field Name	Contents
1-6	Label Identifier	1EOFƀ or 1EORƀ
7-66	Ignored During Trailer Label Processing	
67-72	Block Count	6 numeric characters
73-120	Ignored During Trailer Label Processing	

Figure 1-29. End-of-File/Reel Label (EOF/EOR) - 120 Characters

End-of-File/Reel Label (EOF/EOR)

LABEL IDENTIFIER: This field identifies the label as an end-of-file label or an end-of-reel label, "1EOFƀ" or "1EORƀ", respectively.

On input the occurrence of an EOR label causes tape switching to the next reel. The block count which appears in each trailer label is checked and if the count does not agree with the block count maintained by UFAS an error message is given to the operator, who can then either continue execution, force end-of-file, or abort the job.

When the end-of-tape foil is encountered during the process of writing on an output reel, a 1EORƀ output trailer label is written. When the new reel is provided, processing is continued.

BLOCK COUNT: Block count contained in the trailer label is the number of data blocks contained on this reel of this file. In the case of a multireel file, the block count is the number of data blocks for the current reel only.

RECORD COUNT: Number of data records on the reel. For unblocked records, this value is the same as the block count. For blocked records, this value does not include pad records.

HASH TOTAL OR FILE NAME: Ignored on input. On output, contains file name from the FIB.

End-of-File/Reel Label Structure

Input				
1EOF or 1EOR	1ERI	1ERI		
1EOF or 1EOR	TM	1ERI		
1EOF or 1EOR	TM	1ERI	TM	1ERI[1]
1EOF or 1EOR	TM	1ERI	1ERI	
Output				
MOD1 (odd parity labels)				
1EOF or 1EOR	1ERI	1ERI		
MOD1 or MOD4 (even parity labels)				
1EOF or 1EOR	TM	1ERI	1ERI	
[1] Optional				

Figure 1-30. End-of-File/Reel Label Structure

UFAS recognizes and processes any of the label structures listed for input and always builds an end-of-file/reel group label as indicated for output.

Unlabeled Tape Files

Magnetic tape file labeling is optional for UFF, IBM, H2000, and GFRC, but required for American National Standard. The first and last blocks (control intervals) of an unlabeled tape reel are processed as data. Nothing follows the last block on H2000 UNLABELED files; however, GFRC files are followed by a dummy Trailer Label. It is the user's responsibility to determine EOF on UNLABELED H2000 tapes. Processing of both unlabeled and labeled files is identical in every respect but one: volume and file label processing is bypassed on unlabeled files.

Processing Functions

The UFAS functions for record processing are grouped by file organization and file format in Tables 1-1, 1-2, and 1-3. In addition to the functions to be performed, these tables also list the respective processing mode, and access mode with the applicable parameters.

Table 1-1. Sequential File Functions

UFF, ANS[a], H2000, GFRC, and IBM Sequential Files		
		Access Mode Parameters
Function	Processing Mode	Sequential
OPEN	O, I, I/O[b], Ext[c]	File-ID
WRITE	O, Ext[c]	Next Record
REWRITE	I/O[b]	Current Record
READ	I, I/O[b]	Next Record
CLOSE		File I-D
INFO	O, I, I/O, Ext	File I-D

[a] American National Standard

[b] Applies only to UFF on mass storage.

[c] Extend processing mode is neither applicable to GFRC files on mass storage nor to H2000 files.

Table 1-2. Relative File Functions

UFF Relative Files				
Function	Processing Mode	Access Mode Parameters		
		Sequential	Random	Dynamic
OPEN	O, I, I/O	File-ID	File-ID	File-ID
WRITE	O	Next Record	Relative Key	Relative Key
WRITE	I/O	(Invalid)	Relative Key	Relative Key
REWRITE	I/O	Current Record	Relative Key	Relative Key
DELETE	I/O	Current Record	Relative Key	Relative Key
START	I, I/O	Relative Key	(Invalid)	Relative Key
READ	I, I/O	Next Record	(Invalid)	Next Record
READ	I, I/O	(Invalid)	Relative Key	Relative Key
CLOSE		File-ID	File-ID	File-ID
INFO	O, I, I/O	File-ID	File-ID	File-ID

Table 1-3. Indexed File Functions

UFF Indexed and ISP Files

Function	Processing Mode	Access Mode Parameters		
		Sequential	Random	Dynamic
OPEN	O[a], I, I/O	File-ID	File-ID	File-ID
WRITE	O[a]	Next Record	(Invalid)	Next Record
WRITE	I/O	(Invalid)	Prime Key	Prime Key
REWRITE	I/O	Current Record	Prime Key	Prime Key
DELETE	I/O	Current Record	Prime Key	Prime Key
START	I, I/O	Key of Reference[b]	(Invalid)	Key of Reference[b]
READ	I, I/O	Next Record	(Invalid)	Next Record
READ	I, I/O	(Invalid)	Key of Reference[b]	Key of Reference[b]
CLOSE		File-ID	File-ID	File-ID
INFO	O[ab], I, I/O	File-ID	File-ID	File-ID

[a] Applies only to UFF indexed files
[b] Prime key for ISP files

FILE PROTECTION AND CONCURRENT ACCESS CONTROL

File protection and concurrent access control is allowed only for files that are allocated as random; that is, all UFF file organizations and the ISP file format. When a file is protected by the File Management Supervisor (FMS) ABORT/ROLLBACK/ facility, images of pages (control intervals) are written to a disk collection file under supervision of FMS before they are altered (Befores). If the FMS ACCESS/MONITOR/ is specified during concurrent execution of user programs, simultaneous access to a page by two or more programs can be denied to prevent a deadlock.

During such an occurrence, changes to the file are cancelled and the program is rolled back. If a checkpoint was not specified, the program is aborted; if a checkpoint was specified, the activity is restarted.

If any file in the activity has File Management Supervisor (FMS) ABORT/ROLLBACK/ protection, all permanent files allocated for random update must have ABORT/ROLLBACK/. Otherwise, the activity is aborted by UFAS when the first unprotected file is being opened. Exceptions to this rule are the files that have the character "S" as the first character of the file code. Such files do not require ABORT/ROLLBACK/ when using ACCESS/MONITOR/ protection; however, only file checkpoints can be taken and automatic job restart of these "S" files is not possible if they conflict with other jobs.

If any file in the activity has FMS ACCESS/MONITOR/ protection, UFAS performs ACCESS/MONITOR/ functions on all UFAS files. That is, on a checkpoint, all buffers and record pointers are set empty after they are wiped out.

After any of the following conditions, current record values are eliminated:

- Checkpoint files and files with ACCESS/MONITOR/ present
- Checkpoint files and programs and files with ACCESS/MONITOR/ present
- Rollback files
- Rollback files and programs, and files with ACCESS/MONITOR/ present

All control intervals (pages) accessed before a checkpoint are released (not monitored by FMS); therefore, the record pointer values are not valid.

To continue a series of Get Next functions after one of the four preceding conditions it is necessary to:

1. Save the key value of current record prior to execution of a checkpoint or rollback.
2. Execute a Position macro greater than the saved key value after the checkpoint or rollback.
3. Continue the Get Next function sequence.

When using concurrent access, control interval size must be specified to the File Management Supervisor in the PAGESIZE option. With the TSS Access subsystem, control interval size can be specified by using the PAGESIZE option for the File Create and File Modify directives in the short form. The PAGESIZE (number of words in a page) specified must correspond to the control interval size.

BUFFER POOLING

Buffer pooling can only be requested for UFF relative, indexed, and integrated files. A $ DATA control card must be used to specify the buffers to be pooled (see Appendix A). Pooling can be designated for specific files or for all files of a particular type, such as I-D-S/II.

TRANSLITERATION TABLES

The input and output transliteration tables may be either system or user-supplied tables and are used to transliterate the characters of a record from the external code set to the internal code set or vice versa.

The transliteration tables available have the following designations:

Name	Character Set		
U.GTOA	GBCD	to	ASCII
U.GTOH	GBCD	to	HBCD
U.GTOE	GBCD	to	EBCDIC
U.ATOH	ASCII	to	HBCD
U.ATOG	ASCII	to	GBCD
U.ATOE	ASCII	to	EBCDIC
U.HTOA	HBCD	to	ASCII
U.HTOG	HBCD	to	GBCD
U.HTOE	HBCD	to	EBCDIC
U.ETOH	EBCDIC	to	HBCD
U.ETOG	EBCDIC	to	GBCD
U.ETOA	EBCDIC	to	ASCII

With GMAP, the user must supply a symbol reference for the desired tables.

A user-supplied transliteration table must have the following format:

- 9-bit bytes (4 characters per word)
- Characters right justified within each byte
- Octal code in ascending numerical order
- Maximum table size is 128 words (512 bytes)

SECTION II

UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION

A sequential file can contain either variable-length or fixed-length records on mass storage and 7- or 9-track tape devices. However, these files can be accessed only sequentially. All functions available for the file formats of the sequential organization are listed with their respective macros in Table 2-1. Some of these functions are applicable only to UFF and GFRC files on mass storage as noted. Files are opened, processed, and closed by UFAS in response to a FIB macro and specific functional macro requests.

The processing mode to open a sequential file can be output, input, input/output, and extend. Opening a file for input/output assumes that the file is on a mass storage device and that existing records can be replaced and updated on the file. For extend, positioning takes place so that the first record written is the next logical record following the last current logical record.

Table 2-1 lists the sequential file functions, the functional macros, the processing modes, and the access modes with applicable parameters.

Table 2-1. Sequential File Functions And Processing Modes

UFF and GFRC			
Function	Functional Macro	Processing Mode	Access Mode Parameters Sequential
Open	DOPEN	0, I, I/0[a] , Ext[b]	File-Id
Put	DLPUT	0, Ext[b]	Next Record[a]
Rewrite	DLRWR	I/0[a]	Current Record
Get Next	DLGXT	I, I/0[a]	Next Record
Position	DLPOS	I, I/0,0[d]	Specified Record
Return	UFSRET	0, I, I/0 , Ext[b]	
Checkpoint Files	DCKPF	I/0	
Checkpoint Files and Program[c]	DCKPT	I/0	
Rollback Files	DROLF	I/0	
Rollback Files and Program	DROLP	I/0	
Wrap-up	DWRAP	0, I, I/0[a] , Ext[b]	
Close	DCLOS		File-Id
INFO	DINFO	0, I, I/0[a], Ext	File-ID

[a] Applies only to files on mass storage

[b] Extend processing mode is not applicable to GFRC files on mass storage

[c] Applies only to UFF on mass storage

[d] Output applies only to UFF Sequential and GFRC

FILE INFORMATION BLOCK MACRO

The file information block macro provides the user with an efficient means of communicating to the Unified File Access System the specific characteristics of a file. However, before any communication can be established with UFAS, the user must invoke the macro package with the command LODM .DMAC.

All file processing information between the user and UFAS is channeled through the FIB macro. For each file to be accessed, only one FIB macro may be coded in the user program to define the characteristics of the file and the required processing parameters.

Format

The FIB macro for a sequential file has the following format:

```
FIBMAC fc,ffi, [parameter-name-1,parameter-value-1,...,

parameter-name-n,parameter-value-n]
```

where:

fc - file code assigned to file

ffi - file format indicator (optional)

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area, file status code, and file name, but three are required for the retention period. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area) multiple symbols defining the same address must be used. Default values should not be specified for parameters that are not required, instead allow the default value to be applied automatically.

File Format	File Format Indicator	
UFF	UFF	(Default format)
American National Standard	ANSI	
GFRC	GFRC	
IBM	IBM	
Unlabeled GFRC Tape	UNLT	(UFAS assumes GFRC format - 1 record/block)

The following list contains all the parameters and their respective defaults which are applied if the parameter name and value are not specified:

Parameter	Name	Value	Default	Comments
Number of buffers allocated by USAF	NBUF	Integer ≤ 255 (mass storage) Integer ≤ 2 (tape, P* or I* files)	1	For files residing on tape
				For files residing on mass storage
				If CISZ parameter is specified:
			$\frac{N}{CISZ}$	N = 16384 bytes (4096 words) for UFF files or 24576 characters (4096 words) for GFRC files
				If CISZ parameter is not specified:
			8	For UFF files
			12	For GFRC files
			2	For P* or I* files
Record size in bytes or characters	RSZ	Integer ≤ n		Maximum record size if records are variable length (IFLR parameter = 0); actual record size if records are fixed length (IFLR parameter = 1).
		Maximum n=9999 bytes	80	UFF, American National Standard, and IBM tape files, when records are not spanned.
		Maximum n=2 -1 characters	80	UFF, American National Standard, and IBM tape files when records are spanned.
			84	GFRC files when records are partitioned.
		Maximum n=24570 characters	84	GFRC files when records are not partitioned.
		Maximum n ≤ 16128 bytes	80	UFF (non-partitioned mass storage files)
		Maximum n=2 -1 bytes		UFF (partitioned mass storage files)

Parameter	Name	Value	Default	Comments
Record size word	RSZWRD	Address		Location of a 36-bit binary record size field used as a subordinate to the VIB record size field. UFAS returns the record size to this field on input. On output, the record size provided in this field is used when no record size is provided with DLPUT macro.
			0	If no record size word is specified, either VIB or FIB record size is used.
Record size error indication	IRSZER			Indicates the action UFAS is to take when conflict occurs between the actual record size and that specified by the user in RSZ or RSZWRD, or the VIB record size.
			0	Record size conflicts are ignored. The smallest actual record size or FIB record size is used as the correct record size.
		1		Execute the error procedures specified by the USERER or ERRTBL parameters.
Control interval size in 9-bit bytes or 6-bit characters	CISZ	Integer n		
		Maximum n=16128 bytes	2048	UFF (mass storage)
		Maximum		GFRC (mass storage):
		n=1920 characters	1920	for BCD data
		n=1280 bytes	1280	for ASCII data
		Maximum n=16380 bytes	16380	UFF, American National Standard, and IBM tape files
		Maximum		GFRC (tape):
		n=24570 characters	24570	for BCD data
		n=16380 bytes	16380	for ASCII data

Parameter	Name	Value	Default	Comments
External code	EXTCOD			Code set of the data on the external device. This code is used with the CISZ parameter to calculate the number of words to allocate per buffer. Also, used with the internal code set parameter to establish the size of the characters within the record and the process area, so that the records may be converted to this code set as they are moved to the buffer. NOTE: UFF files may specify either ASCII or 36-bit binary as the external code set. If ASCII is specified, then the internal code set may be any of those listed, except 36-bit binary. If the external code set is 36-bit binary, then the internal code set must be 36-bit binary.
		ASC	ASC	ASCII character set.
		EBCD		EBCDIC character set.
		GBCD		GBCD (GFRC BCD).
		HBCD		HBCD (Honeywell BCD).
		JIS		JIS (Japanese Industrial Set).
		BIN		36-bit binary (UFF only)
Internal code	INTCOD			Code set of the data in the process area. Records are converted to this code set as they are moved to the process area.
		ASC	ASC	ASCII character set.
		EBCD		EBCDIC character set.
		GBCD		GBCD character set.
		HBCD		HBCD character set.
		JIS		JIS character set.
		BIN		36-bit binary (UFF only)

Parameter	Name	Value	Default	Comments
Checkpoint	CHKPT	Integer ≤ 2^{17} -1		Checkpoints are taken every time this many records are accessed.
			0	No checkpoints are taken based on record count.
Process area	PROAR	Address, character		Location and starting character or byte position of an area within the user program to which records are moved and from which they are retrieved with Put and Get macros. For GFRC files character position is assumed to be zero. The size of the process area is assumed to be equal to the FIB record size. If this parameter is not specified, the in-place processing parameter must be specified.
In-place processing	INPLAC	Address		Location of a 24-bit word and byte address used to hold a record address or the address of the next record area within the buffer. When a Get macro is issued, UFAS stores the address of the record in this location. On output, at the completion of each Put macro, and, on open, the address of the next record area in the buffer is stored in this location. If this parameter is not specified, the process area parameter must be specified. NOTE: This parameter applies only to GFRC, UFF, and IBM file formats.
File status code	FSCODE	Address, character		Location and starting byte position of a two-byte ASCII file status code field in which UFAS stores a unique status code whenever an exception condition occurs (see Section VIII).
User label process	LABAR	Address, character		Location and starting character or byte position of an area within the user program to which User label records are moved to or retrieved from during User label processing. This parameter must be present for ANS, UFF, and IBM tape files with user label processing.
			0,0	No file status code field.

Parameter	Name	Value	Default	Comments
User error procedure	USERER	Address		Location of user-supplied error procedure to which UFAS transfers control whenever an error is detected.
			0	No user error procedure.
Error table	ERRTBL	Address		Location of user-supplied table of error procedures based on processing mode. The format of the error table is: word 0, bits 0-17: address of error procedure for files in output mode. word 1, bits 0-17: address of error procedure for files in input mode. word 2, bits 0-17: address of error procedure for files in input/output mode. word 3, bits 0-17: address of error procedure for files in extend mode. (bits 18-35 of each word are reserved.) When an error is detected, control is transferred to the appropriate error procedure only if the user error procedure parameter is not specified. If neither a user error procedure parameter nor an error table parameter is specified, UFAS aborts when an error is detected.
			0	No user error procedures based on processing mode.
Input transliteration table	INPTRN	Address		Location of a system or user-supplied table. This table is used to transliterate the characters of a record from the external code set to the internal code set, as the record is moved from the buffer to the user's process area. Unless a process area parameter is specified, this parameter is ignored.

Parameter	Name	Value	Default	Comments
			0	Transliteration not required. Records are moved from the buffer to the user's process area (if specified) without transliteration. In this instance, the internal and external code sets are usually the same.
Output transliteration table	OUTTRN	Address		Location of a system or user-supplied transliteration table. This table is used to transliterate the characters of a record from the internal code set to the external code set, as the record is moved from the user's process area to the buffer. Unless a process area parameter is specified, this parameter is ignored.
			0	Transliteration not required. Records are moved from the user's process area (if specified) to the buffer without transliteration. In this instance, the internal and external code sets are usually the same.
File code list	FCLST	Address		If the file resides on more than one volume, this parameter defines the location of a list of file codes assigned to the second and succeeding volumes of the file. This parameter applies only to mass storage files. The format of the file code list is: word 0, bits 0-35: number of file codes in the list, in binary. word 1-n, bits 0-17: reserved. bits 18-35: file code in ASCII.
			0	Only one file code is assigned to the file.

Parameter	Name	Value	Default	Comments
Media code	MEDCOD			Applies only to GFRC files.
		0	0	Not a media conversion record or no printer slew controls.
		1		Binary card image.
		2		BCD card image.
		3		BCD print line image.
		6		GFRC format in ASCII.
		7		ASCII print line image.
		8		TSS information record.
		9		BCD print line image with report code as first two characters of a logical record.
		10		ASCII card image.
		13		ASCII print line image with report code as first two characters of a logical record.
Block serial number indicator	IBSN			Applies only to UFF (tape), American National Standard, and GFRC files.
		0	0	Block serial numbers are used.
		1		Block serial numbers are not used.
			1	American National Standard files.
Fixed-length record indicator	IFLR			Applies only to UFF, American National Standard, IBM, and GFRC tape files.
		0	0	Records are variable length.
		1		Records are fixed length.
Optional file indicator	IOPTFL			Applies only to sequential files open for input.
		0	0	File must be present.
		1		Actual file may or may not be present when file is opened. If not present, the first Get macro results in an end-of-file condition.

Parameter	Name	Value	Default	Comment
Multifile indicator	IMULTF			Applies only to labeled tape files and IBM and UFF unlabeled
		0	0	File does not reside on multifile volume set.
		1		File resides on multifile volume set (cannot be used with open extend).
Partitioned/ spanned record indicator	IPARRC			Applies to UFF, American National Standard, and IBM tape files as well as GFRC and UFF mass storage files
		0	0	File does not contain partitioned/spanned records.
		1		File contains partitioned/ spanned records.
Recording mode indicator	IRMODE			Applies only to GFRC tape files
		0	0	Recording mode is binary.
		1		Recording mode is BCD.
File sequence number	FILSEQ			Applies only to labeled input tape files and IBM and UFF unlabeled
		0	0	Tape is already positioned to beginning of next file.
		Integer $\leq$ 512		Sequence number of the file on a multifile volume set
File name in ASCII	FLNAME	Address, character		Applies only to labeled files
				Location and starting byte position of file name. File name (in ASCII) is 17 bytes for UFF, American National Standard and IBM files, 12 bytes for GFRC files.
			0,0	No file name is provided. If parameter is not required, allow default to be assumed.

Parameter	Name	Value	Default	Comments
Retention period	RETPER	Address, character, length		Applies only to labeled files.
				Location of starting byte position and length of the retention period field (maximum of three bytes).
			0,0,0	No retention period is provided. If parameter is not required allow default to be assumed.
Label exit table	LBLEXT	Address		Applies only to labeled ANS, UFF, IBM, or GFRC tape files. If this parameter is specified control is transferred to the appropriate label procedure.
				Location of a table containing user-supplied label procedures.
				The format of the GFRC table is:
				word 0, bits 0-17: address of before beginning file label procedure; user receives control after label is read, but before it is checked.
				word 1, bits 0-17: address of after beginning file label procedure; user receives control after label is checked.
				word 2, bits 0-17: address of before ending file label procedure; user receives control after label is read, but before it is checked.
				word 3, bits 0-17: address of after ending file label procedure; user receives control after label is checked.
				(bits 18-35 of each word are reserved.)

Parameter	Name	Value	Default	Comments
				For GFRC files when control is transferred to the user, index register 1 contains the FIB address of the file and index register 2 contains the address of the buffer containing the label. Index register 0 contains the return address to UFAS which the user may use upon completion of the label procedure to allow UFAS to continue, if the return macro (UFSRET) is not used instead of register 0.
				The format of the ANS, UFF, and IBM Table is 12 words in length. Bits 18-35 of each word are reserved. All 12 words must be present and bits 0-17 must be zero if the exit is not desired. Bits 0-17 contain the address of the User label procedure to be performed. The exit functions are as follows:
				Word 1 - Input User Header labels Word 2 - Input User Volume labels Word 3 - Output User Header labels Word 4 - Output User Volume labels Word 5 - Extend User Header labels Word 6 - Extend User Volume labels Word 7 - Input User EOF trailer labels Word 8 - Input User EOV trailer labels Word 9 - Output User EOF trailer labels Word 10 - Output User EOV trailer labels Word 11 - Extend User EOF trailer labels Word 12 - Extend User EOV trailer labels
			0	No user label exit procedures.

Parameter	Name	Value	Default	Comments
Label indicator	ILABEL			Indicates the presence or absence of file labels. (Applies only to UFF files on mass storage, plus GFRC tape and IBM tape files.
		0	0	File contains labels.
		1		File is unlabeled.

Rules

1. The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner cause an error to be generated.

2. If media code is not zero, a process area is required.

3. If system transliteration tables are required, the parameter values for the input (INPTRN) and output (OUTTRN) transliteration tables must be the SYMREF of the tables. If a user-supplied transliteration table is used, the INPTRN and OUTTRN values are the addresses of the respective tables.

FUNCTIONAL MACROS

Even though each one of the functional macros available performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value is assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. The only exception is when a user error or label exit procedure is entered. In this instance, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error or label procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of one or more files, verifies their existence, and establishes tables for file processing. DOPEN consists of the following four functions:

1. Single-Open/User-Defined FIB List
2. Single-Open/Macro-Defined FIB List
3. Multiple-Open/User-Defined FIB List
4. Multiple-Open/Macro-Defined FIB Lists

SINGLE-OPEN/USER-DEFINED FIB LIST

Format

```
DOPEN ⎧ FIBLS,address of FIB list              ⎫    , [parameter-name-1,
      ⎩ FIBLSI,address of pointer to FIB list  ⎭

       parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to a set of routines used to process the file.
	SYM	Routine package SYMREF name		Package of routines needed to process the file.
	SYMI	Address		Location of the pointer to the routine package.

Parameter	Name	Value	Default	Comments
Processing mode indicator				Optional
	MODE	OT	OT	Output
		IN		Input
		IO		Input/Output (applies only to files on mass storage)
		EX		Extend (does not apply to GFRC on mass storage)
	MODEI	Address		Location of processing mode indicator (2 ASCII uppercase characters in bits 0-17)
Rewind indicator				Optional Applies only to tape files
	REW	Y	Y	Rewind on open
		N		Do not rewind on open
	REWI	Address		Location of rewind indicator, in uppercase ASCII, with either Y and blank or N and blank in bits 0-17
User-Supplied EOF character				Optional Applies only to TSS input (default is carriage return)
	EOFCHR	Integer	13	User-supplied EOF character
	EOFCHRI	Address		Location of EOF character in bits 0-35

Routine Package SYMREF Name

The Unified File Access System is a flexible system that allows the user to utilize the whole system or just portions of the system that are applicable to the format of the file to be processed. In order to select the UFAS routines to be linked at load time, the user must supply a SYMREF (directory) name for the specific set of routines desired. Table 2-2 lists the file format characteristics and the corresponding directory names.

Table 2-2. Sequential File Format Directories

File Format	Access Mode	Processing Mode	Directory Name (SYMREF)
GFRC			
FLR Process Area	Sequential	0	.DFLPO[a]
FLR no Process Area	Sequential	0	.DFLNO[a]
VLR Process Area	Sequential	0	.DVLPO[b]
VLR no Process Area	Sequential	0	.DVLNO[b]
Printer Formatted Output	Sequential	0	.DPRNT[c]
FLR	Sequential	I	.DFLRI[a]/.DAFPI[e]
VLR	Sequential	I	.DVLRI[b]/.DAVPI[e]
VLR	Sequential	I/0	.DVIO[d]
American National Standard			
FLR/VLR	Sequential	I	.DAFIN/.DAAPF[e]
FLR/VLR	Sequential	0	.DAFOT
IBM			
FLR/VLR	Sequential	I	.DIFIN/.DAIPF[e]
FLR/VLR	Sequential	0	.DIFOT
UFF			
Sequential (mass storage)	Sequential	I,I/0,0,Ext	.DSEQ/.DASPI[e]
Sequential	Sequential	I	.DUTPI[b]
Sequential	Sequential	0	.DUTPO[b]

[a] Uses tape and media code zero

[b] Uses tape and mass storage

[c] Uses tape or mass storage and media codes 3, 7, 11, and 15 (octal)

[d] Uses mass storage only

[e] Includes routines for positioning function

Rules

1. The Open macro must be executed before the execution of any input/output statements for a file.

2. After the initial execution of an Open macro for a file, each subsequent DOPEN for this file must be preceded by the execution of a Close macro.

   Open output allows file creation. Open input allows an existing file to be read. Updating of an existing file requires opening for input or output processing. Adding records to an existing file requires opening for extend processing.

3. The Open macro does not obtain or release the first data record.

4. For files being opened for input or input/output, the DOPEN establishes the current record pointer so that the first data record in the file can be accessed.

5. For files opened in the extend mode, the file is positioned immediately following the last current logical record. Subsequent write statements add records to the file as though the file was opened for output.

6. For files opened for output, the current record pointer is initialized to point to the beginning of the user's data area in the logical data space.

7. The I/O processing mode applies only to mass storage files.

8. The extend processing mode applies only to sequential single files (single- or multi-unit).

9. Open label processing and file positioning takes place as follows:

   a. When a sequential file is opened for output, standard header labels are written.

   b. When a sequential file is opened for input, header labels are checked.

   c. When a sequential file is opened for extend:

      Beginning file label or unit label is processed for input.

      Existing ending file label is processed for input.

      Ending file label is deleted and processing proceeds as though the file was opened for output.

d. When a file on a multifile volume is to be opened, the file sequence number in the FIB and the rewind indicator in the FIB list are used to position as follows:

If the file sequence number is zero and no rewind is specified, then the reel is assumed to be already positioned at the beginning of the file desired (input) or at the end of the previous file (output).

If the file sequence number is zero and rewind is specified, then positioning is made to the beginning of the reel (input and output).

If the file sequence number is "n", positioning is made to the beginning of file number "n" (input) or at the end of file "n" (output). The end of file "n" is specified by file sequence number "n+1".

10. If in-place processing is specified, the Open macro returns the address of the first record area in the buffer to the location specified by the in-place parameter.

Example

Open a GFRC file with fixed-length records for output. This Open macro contains the address of the FIB list (LIST) in which the FIB address (FIB), directory (SYMREF) name (.DFLPO), and processing mode is stored. Note that the MODEI parameter references a location that contains the processing mode indicator "OT". Because the rewind indicator is omitted, the file is rewound before opening (default).

```
1      8      16
---------------------------------------------------------
       LODM   .DMAC
       SYMREF .DFLPO
       DOPEN  FIBLS,LIST, [FIBA,FIB,SYM,.DFLPO,MODEI,ACCMOD]
       .
       .
       .
LIST   BSS    3
FIB    FIBMAC GO,GFRC,  [IFLR,1,PROAR,CARD,0,RSZ,80]
       .
       .
       .
ACCMOD UASCI  1,OT
CARD   BSS    20
```

SINGLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of only one file. It builds a FIB list in the user program immediately following the DOPEN command, and storage is allocated automatically.

Format

```
     DOPEN  ,, [parameter-name-1,parameter-value-1,...,parameter-name-n,
     ETC       parameter-value-n]
```

The following parameters and respective defaults apply.

Parameter	Name	Value	Default	Comments

Same as Single-Open/User-Defined Function.

Rules:

1. Do not supply FIB list address.
2. Left bracket must be preceded by two commas.
3. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/USER-SUPPLIED FIB LIST

This function initiates the processing of from one to n files. It builds one FIB list, which is used by all files.

Format

```
     DOPEN   FIBLS, FIB list address,  [*, FIB address, mode,
     ETC     rewind indicator, symref name, *, FIB address 2,
     ETC     mode 2, rewind indicator 2, symref 2,....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	symref name		Required; package of routines needed to process the files

Rules:

1. Space for FIB list must be defined as 1+2n words where n is the number of files to be opened.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of from one to n files. It builds one FIB list within the user program immediately following the DOPEN command. This list is used by all files. Storage for the FIB list is automatically allocated.

Format

```
DOPEN   ,, [*, FIB address, mode, rewind indicator
ETC        symref name,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	symref name		Required; package of routines needed to process the files

Rules:

1. Two commas must precede the left bracket enclosing file parameters.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. FIB list address should not be supplied.
6. All other rules that apply to the Single-Open/User-Defined Function.

Put Macro - DLPUT

Function

The Put macro places a logical record in the file.

Format

```
DLPUT  FIB address,VIB address,,

         [VIB-parameter-name-1,VIB-parameter-value-1,...,

         VIB-parameter-name-n, VIB-parameter-value-n]
```

NOTE: VIB must be defined as BSS 2.

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Record size				Optional Used with variable-length records, but ignored with fixed-length records.
	RSZ	Integer		Must be less than or equal to RSZ in FIB.
	RSZI	Address		Location of field containing record size in bits 0-17.
Report code				Optional Report code applies only to GFRC files. If a negative value is supplied, a default report code, assigned when the file is opened, is used.
	RPCOD	1≤Integer < 63	0	
	RPCODI	Address		Location of report code field in bits 18-35.
Number of lines to slew				Required Only if slew code SLCOD(I) is 4 or 5. Applies only to System Output and GFRC files.
	SLEWN	Address		Location of word containing number of lines to slew in bits 18-35.

Parameter	Name	Value	Default	Comments
Slew code				Optional Applies only to GFRC files Specifies spacing before or after print
	SLCOD	0	0	Print after slewing one line
		1		Print before slewing any lines
		2		Print before top of page
		3		Print after top of page
		4		Print before slewing the number of lines specified in SLEWN
		5		Print after slewing the number of lines specified in SLEWN
	SLCODI	Address		Location of word containing slew code in bits 18-20 (bits 21-35 must be zero)
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.
Segment code	SGCOD	Integer	0	Applies when user supplies record segments to UFAS where: 0 = only segment 1 = first segment 2 = middle segment (GFRC) last segment (UFF) 3 = last segment (GFRC) middle segment (UFF)
	SECODI	Address		Location of word containing the segment code in bits 18-20
	NOCRLF		None	Indicates that the normal carriage return line feed sequence after TSS output will be suppressed. The user will take care of slew controls.
Number of user-supplied control characters	CCHRS	Integer	0	Optional applies only to TSS terminal output
	CCHRSI	Address		Location of word containing CCHRS in bits 18-35

Rules

1. The designated file must be open for output or extend at the time of the execution of this statement.

2. The current record pointer is unaffected by the execution of a Put macro.

3. The maximum record size for a file is established at the time the file is created and cannot subsequently be changed.

4. When an attempt is made to write beyond the externally defined boundaries of a sequential file, an exception condition results and the file is unaffected.

5. If a process area is specified in the FIB, the record is moved from the process area to the buffer. If an output transliteration table is specified, the entire record is transliterated from the internal code set to the external code set as it is moved.

6. Before the Put macro is issued, if in-place processing is specified, the user must have placed the record in the record area denoted by the address returned after the previous Put or Open macro.

7. If the record size is not specified and records have variable length, the record size in the record size word parameter is used. If no parameter is specified, the FIB record size is used.

Example

Put a 70-character line in the file specified by FIB2. The line includes a slew code to print the line before slewing (15 lines) and a report code of 25.

```
1      8       16
______________________________________________________________________________
       DLPUT   FIB2,VIB2,,[RSZI,ADDSIZ,RPCOD,25,SLEWN,SL15,SLCOD,4,FIBI]
       .
       .
       .
FIB2   ARG     FIBX
       .
       .
       .
VIB2   BSS     2
       .
       .
       .
ADDSIZ ZERO    70,0
SL15   DEC     15
FIBX   FIBMAC  PR,GFRC,  [RSZ,160,PROAR,LINE,0,MEDCOD,7]
       .
       .
       .
LINE   BSS     40
```

Rewrite Macro - DLRWR

Function

The Rewrite macro causes the record made available by the last Get Next function to be replaced in the file.

Format

```
DLRWR  FIB address,VIB address,alternate return address,

       [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 1.

The parameters and respective defaults for the Rewrite macro are as follows:

Parameter	Name	Value	Default	Comments
Record size				Ignored for in-place processing. Optional Specifies the number of characters of the record to be rewritten (when process area is specified).
	RSZ	Integer		Value must be equal to the size of the record being replaced.
	RSZI	Address		Location of record size field in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for I/O at the time of execution of this statement.

2. For files opened for sequential access, the last input/output function executed for the associated file, prior to the execution of the DLRWR macro, must have been a successfully executed DLGXT.

3. The size of the record must be equal to the size of the record being replaced.

4. The current record pointer is not affected by the execution of a DLRWR macro.

5. When the rewrite is unsuccessful, the file is unaffected.

6. If record size is not specified, the record size in the FIB record size word parameter is used. If no parameter is specified, the FIB maximum record size is used.

Example

Rewrite the 20-byte record just read from the file specified in FIBAD. If the record size is not equal to that of the record being replaced, return is made to ALTRT.

```
1      8         16
------------------------------------------------------------
       DLRWR     FIBAD,VIBAD,ALTRT, [RSZ,20]
       .
       .
       .
FIBAD  FIBMAC    DF,,  [PROAR,RECORD,0]
       .
       .
       .
VIBAD  BSS       1
ALTRT  NULL
       .
       .
       .
RECORD BSS       20
```

Get Next Macro - DLGXT

Function

The Get Next macro makes the next logical record available in the predecessor-to-successor relationship established when the file was created.

Format

```
DLGXT  FIB address,VIB address,end-of-file address,

       [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 1.

The parameters and respective defaults for the Get Next macro are as follows:

Parameter	Name	Value	Default	Comments
Record size				Applies only to GFRC and IBM files. Ignored for in-place processing. Optional Specifies the number of characters to be moved to the process area from the record when the entire record is not required (when process area is specified). UFAS stores the actual record size in the FIB record size word parameter.
	RSZ	Integer		Value must be less than or equal to RSZ value given in the FIB macro.
	RSZI	Address		Location of record size in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input or I/O at the time this macro is executed.

One segment selection	ONESEG	None	Optional If this keyword is specified, the implication is that only one segment will be returned for segmented records
TSS User's PROMPT	PROMPT	Address	Address of word containing: bits 0-17 Address of start of user's input prompt bits 18-35 User's prompt character length

Rules

1. The designated file must be open for input or I/O at the time this macro is executed.

2. An end-of-file condition is returned when no next logical record exists in the file. This condition is considered an unsuccessful read request. Control is returned to the end-of-file address.

3. If an optional file is not present, then an end-of-file condition is returned on the first read request. The end-of-file address receives control.

4. If the position of the current record pointer is undefined, an error condition results. The current record pointer is not defined following an unsuccessful read request.

5. If the record size parameter is not specified, the whole record is moved to the process area and the record size is made available to the user in bits 0-17 of word 0 of the VIB and in the word specified by the record size word parameter in the FIB.

6. If a VIB is not specified, the whole record is moved to the process area and the record size is made available to the user only if a record size word parameter is specified in the FIB.

7. For sequential files on removable mass storage or labeled tape, the following steps are executed if the end of unit is reached before an end of file:

   a. Trailer label is checked.

   b. Unit switch is attempted (if this is the last unit, an end-of-file condition results).

   c. Header label of the next unit is checked.

   d. First record of the next unit is made available to the program.

Example

Get the next record in sequential order and move 50 bytes of a 100-byte record to the user process area. The actual record size (100 bytes) is placed in the location defined by the record size word parameter (NMBR). At end of file execute the procedure beginning at ALT.

```
1      8      16
_____________________________________________
       DLGXT  FIB,VIB,ALT, [RSZ,50,FIBI]
       .
       .
       .
VIB    BSS    1
ALT    NULL
       .
       .
       .
FIB    ARG    FIBV
FIBV   FIBMAC AA,UFF,  [RSZWRD,NMBR,PROAR,RECORD,0,RSZ,100]
       .
       .
       .
NMBR   BSS    1
RECORD BSS    25
```

Position Macro - DLPOS

Function

The Position macro provides for logically positioning the file for subsequent access.

Format

```
DLPOS  FIB address,VIB address,alternate return address,
       [VIB-parameter-name-1,VIB-parameter-value-1,
       VIB-parameter-name-2,VIB-parameter-value-2]
```

NOTE: VIB must be defined as BSS 2.

The parameters and respective defaults for the Position macro are as follows:

Parameter	Name	Value	Default	Comments
File beginning	BEGIN	FILE		Optional Device is positioned at beginning of file .
Backward positioning				Optional Specifies the number of logical records to be skipped over in the backward direction.
	BACK	Integer	0	Number of logical records to be skipped over in the backward direction[1].
	BACKI	Address		Location containing a 36-bit binary value representing the number of logical records to be skipped over in the backward direction[1].
Forward positioning				Optional Specifies the number of logical records to be skipped over in the forward direction.
	FORW	Integer	0	Number of logical records to be skipped over in the forward direction.
	FORWI	Address		Location containing a 36-bit binary value representing the number of logical records to be skipped over in the forward direction.

[1] Applies only to single-volume files.

Parameter	Name	Value	Default	Comments
Number of logical records skipped over	OUTNBI	Address		Optional Location in which a 36-bit binary value representing the number of logical records actually skipped over is returned. Applicable only when backward or forward positioning is used.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open in the input or input/output processing mode.

2. If an end-of-file is encountered before n logical records are bypassed, the alternate return is taken. The number of logical records actually skipped over is returned to the user through the OUTNBI parameter. If the contents of the OUTNBI parameter do not equal the original input parameter, the status field must be checked to determine if an end-of-file condition occurred.

3. If an end-of-volume is encountered before n logical records are bypassed, UFAS performs volume switching, label-checking, and positioning at the proper place on the new volume.

4. If a beginning-of-file is encountered before n logical records are bypassed, the alternate return is taken. The number of logical records actually skipped over is returned to the user through the OUTNBI parameter. If the contents of the OUTNBI parameter do not equal the original input, the status field must be checked to determine if a beginning-of-file condition occurred.

5. If a beginning-of-volume is encountered before n logical records are bypassed, an error condition occurs and control is passed on to the user-specified error routine if it exists; otherwise, an abort occurs.

Examples

Rewind file.

```
1      8       16
_____________________________________________
       DLPOS   FIB1,VIB1,ALT1,[BEGIN,FILE]
       .
       .
       .
VIB1   BSS     2
FIB1   FIBMAC  RF,UFF, [PROAR,RECORD,0]
       .
       .
       .
ALT1   NULL
       .
       .
       .
RECORD BSS     20
```

Backspace one logical record.

```
1      8       16
____________________________________________________
       DLPOS   FIB2,VIB2,ALT2,[BACK,1,OUTNBI,TRUENB]
       .
       .
       .
VIB2   BSS     2
TRUENB DEC     0
FIB2   FIBMAC  B1,UFF,  [PROAR,RECORD,0]
       .
       .
       .
ALT2   NULL
       .
       .
       .
RECORD BSS     20
```

Skip over four logical records in the forward direction.

```
1      8       16
____________________________________________________
       DLPOS   FIB3,VIB3,ALT3, [FORWI,RECNB,OUTNBI,TRUENB]
       .
       .
       .
VIB3   BSS     2
RECNB  DEC     4
TRUENB DEC     0
FIB3   FIBMAC  F4,UFF, [PROAR,RECORD,0]
       .
       .
       .
ALT3   NULL
       .
       .
       .
RECORD BSS     20
```

Read User Labels - DLRLB

Function

The read user label macro retrieves a user label record and places it in the defined label process area.

Format

DLRLB FIB address, VIB address, Alternate return, [VIB parameter-name-1]

NOTE: VIB must be defined as BSS 1.

Parameter	Name	Value	Default	Comments
Indirect FIB Addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. This macro may be used only in an ANS, UFF, or IBM user label processing procedure.

2. The alternate return is taken when no next user label is present.(Major Status Code 5) A UFSRET (USERLB) command must be issued to return to UFAS.

Example

Read the next user head label record and place it in the label process area. If no next user label record is present, go to a procedure beginning at ALT.

```
1      8       16
_____________________________________

EXADR  DLRLB   FIB,VIB,ALT,[FIBI]
       .
       .
       .
VIB    BSS     1
ALT    NULL
       .
       .
       UFSRET  USERLB
FIB    ZERO    FIBV
FIBV   FIBMAC  AA,UFF,[LABAR,AREA,0,STATUS,0,LABEXT,LABTB]
AREA   BSS     20
STATUS BSS     1
LABTB  ZERO    EXADR,0        INPUT USER HEADER LABELS EXIT
       BSS     11
       .
       .
       .
       .
       .
       .
```

Write User Labels - DLWLB

Function

The write user label macro writes the user label record which has been placed in the defined label process area.

Format

DLWLB FIB address, VIB address,,[VIB parameter-name-1]

NOTE: VIB must be defined as BSS 2.

Parameter	Name	Value	Default	Comments
Indirect FIB Addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. This macro may be used only in a ANS, UFF or IBM user label processing procedure.

2. The user label identification will be checked for a UVL, UHL, or UTL in the first three positions of the label. The identification must be defined as UASCI. The program will be aborted if the test fails.

3. A UFSRET (USERIB) must be issued to terminate the user label procedure.

Example

Write a user head label record which is in the user label process area.

```
1      8       16
EXADR  DLWLB   FIB,VIB,,[FIBI]
       .
       .
       .
       UFSRET  USERLB
VIB    BSS     2
FIB    ZERO    FIBV
FIBV   FIBMAC  AA,UFF,[LABAR,ULAB,0,STATUS,0,LBLEXT,LABTB]
STATUS BSS     1
LABTB  ZERO    0,0
       ZERO    0,0
       ZERO    EXADR,0         OUTPUT USER HEADER LABEL EXIT
       BSS     9
        .
        .
        .
        .
        .
        .
ULAB   UASCI   1,UHLO
ULDATA BSS     19
```

Return Macro - UFSRET

Function

The Return macro provides a means of returning control to UFAS after a user error or label processing procedure is completed.

Format

$$\text{UFSRET}\left\{\begin{array}{l}\text{USERLB}\\\text{LABEL}\\\text{ERROR}\end{array}\right\}$$

LABEL - Must be specified when returning to UFAS from user GFRC label exit processing.

ERROR - Must be specified to return from a user error procedure.

NOTE: Both the user error and the label procedures are specified in the FIB macro.

USERLB - Must be specified when returning to UFAS from an ANS, UFF or IBM user label exit processing.

Rule

When a user error or label procedure contains a UFAS macro, which can result also in a user error or label procedure for another file being entered by UFAS, the second procedure must conclude also with a UFSRET return for that particular function.

Example

Return to UFAS, which has issued a user label exit call to the user.

```
1      8      16
       UFSRET LABEL
```

Information Macro - DINFO

FUNCTION

The DINFO macro stores File Access Control Table (FACT) information or the size of each section of the FACT table in a user specified table area.

FORMAT

```
DINFO   FIB address, VIB address, alternate return  address,  [parameter-name-1,
        parameter-value-1...., parameter-name-n, parameter-value-n]

NOTE:   VIB must be defined as BSS 1
```

The parameters and respective defaults for the Information Macro are as follows:

Parameter	Name	Value	Default	Comments
Function Code	FCT	1		Store FACT section 1 into user table area. Includes words FACT-8 through FACT +98 for all file types.
		2		Store FACT section 2 into user table area. Includes words 0 through +59 for UFF Indexed files with no alternate keys: 0 through +81 for UFF Indexed with alternate keys; 0 through +17 for ANSI/IGM/GFRC and UFF tape 0 through +41 for ISP; and 0 through +9 for H2000.
		3		Store FACT section 3 into user table area. Include words 0 through 26 (dec.)
		4	4	Store the entire FACT into the user table area.
		5		Store FACT-8 through FACT +5 of section 1 into user table area. This option provides FACT information that is equivalent to the GFRC FILCB.
		6		Store the size (in binary) of FACT sections 1, 2, and 3 into a 3-word user defined table area. Zero will be given as the size of section 3 if it is non-existent.

Table Pointer	TBL	Address	Required. The location of a table large enough to contain the requested information.
Indirect FIB Addressing	FIBI	None	Optional. If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address in bits 0-17.

RULES:

1. The designated file must be opened in either the input, I/O, output or extended processing mode.

2. The TBL parameter is required and must point to an area large enough to contain the requested information.

3. The user must provide an alternate return address.

4. If an error is detected during execution of the DINFO macro, UFAS will return control to the user via the alternate return except when executing the DINFO macro on any unopened file in which case UFAS will branch to a users error procedure if it is present or abort.

5. A function code of 6 requires a table size of 3 words. After a successful execution of a DINFO macro the table will contain the following values:

Word	Contents
1	Size of FACT section 1, right-justified. (In Binary)
2	Size of FACT section 2, right-justified. (In Binary)
3	Size of FACT section 3, right-justified. (In Binary) A size of zero will be given if section 3 is non-existent.

6. If no function code (FCT) parameter is specified the entire FACT will be given to the user.

7. The FACT sizes given in this specification are applicable to SR ADF2.

8. This macro may not be executed for files processed under TP or I-D-S/II.

9. The following error conditions will be identified:

a.	8-10	File not opened. (ALT return not taken)
b.	F0	A potential F0 memory fault during DINFO execution may take place because of insufficient table area size.
c.	F1	Invalid function code
d.	F2	The user asked for FACT section 3, and it does not exist.

10. The sizes, in decimal, of the entire FACT for the various file formats are:

UFF SEQUENTIAL TAPE AND MASS STORE	192
UFF RELATIVE	192
UFF INDEXED	192
IBM	152
ANSI	152
H2000	136
UNLT	152
GFRC Tape and Mass Store	152
ISP	149

Example

Store the entire FACT contents of the file specified by FIBI into a table specified by TABLE.

```
1      8      16
______________________________________________
       DINFD  FIB1,VIBI,ALTRET,[FCT,4,TBL,TABLE]
       .
       .
       .
FIB1   FIBMAC DA,UFF[PROAR,RECORD,0,ORG,IND,
       ETC    KEYPTR,PRIME],KY
       .
       .
       .
RECORD BSS    21
PRIME  VFD    1/0,17/0,18/4
       ZERO
VIBI   BSS    1
TABLE  NULL
       .
       .
       .
ALTRET NULL
```

Checkpoint Files Macro - DCKPF

Function

The Checkpoint Files macro establishes a checkpoint for all open files.

Format

DCKPF

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.

2. All opened files are included in the checkpoint.

3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur, since the program is not checkpointed.

4. The macro performs a buffer flush before establishing the checkpoint.

5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.

6. Checkpoint is taken only if protected files are present and have been opened.

Checkpoint Files and Program Macro - DCKPT

Function

The Checkpoint Files and Program macro establishes a checkpoint for all open files and for the program.

Format

DCKPT

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.

2. All opened files are included in the checkpoint.

3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur.

4. The macro performs a buffer flush before establishing the checkpoint.

5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.

6. A QX file large enough to accommodate the program must be allocated, but not declared in the user program (see Program Recovery/Restart manual).

7. Checkpoint is taken only if protected files are present and have been opened.

8. The Q register (lower 18 bits) contains the status code returned by MME GECHEK as described in the General Comprehensive Operating Supervisor manual. Zero status indicates a successful checkpoint.

Rollback Files Macro - DROLF

Function

The Rollback Files macro restores the files open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the previous checkpoint.

Format

```
DROLF
```

Rules

1. A DCKPF or DCKPT macro must be executed before performing the rollback.

2. The current record pointer is undefined after a rollback.

Rollback Files and Program Macro - DROLP

Function

The Rollback Files and Program macro restores the program and all files that are open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the checkpoint and to restart the program from that point.

Format

DROLP

Rules

1. A DCKPT macro must be executed before a rollback is executed.

2. The current record pointer is undefined after a rollback.

Wrap-up Macro - DWRAP

Function

The Wrap-up macro closes files currently open.

Format

DWRAP

Rule

All opened files are closed with rewind. Device disposition codes are not honored.

Close Macro - DCLOS

Functions

The Close macro terminates the processing of one or more files. DCLOS consists of the following four functions:

1. Single-Close/User-Defined FIB List
2. Single-Close/Macro-Defined FIB List
3. Multiple-Close/User-Defined FIB List
4. Multiple-Close/Macro-Defined FIB List

SINGLE-CLOSE/USER-DEFINED FIB LIST

Format

```
DCLOS { FIBLS,address of FIB list              },[parameter-name-1,
      { FIBLSI,address of pointer to FIB list  }

        parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Rewind indicator				Optional Applies only to tape and removable mass storage files.
	REW	RWFL	RWFL	Rewind file.
		NRFL		Do not rewind file.
		RLFL		Rewind and lock file (see note).
		LKFL		Lock file without rewind (see note).
		RUFL		Rewind and unload file.
		R3FL		Lock file with rewind and unload (see note).

Parameter	Name	Value	Default	Comments
		RWUN		Rewind unit.
		NRUN		Do not rewind unit.
		RUUN		Rewind and unload unit.
	REWI	Address		Location of rewind indicator (4 ASCII uppercase characters).
	REWV			Rewind value.
		0	0	Rewind file.
		1		Do not rewind file.
		2		Rewind and lock file (see note).
		3		Lock file without rewind (see note).
		4		Rewind and unload file.
		6		Lock file with rewind and unload (see note).
		8		Rewind unit.
		9		Do not rewind unit.
		10		Rewind and lock unit (see note).
		12		Rewind and unload unit.
	REWVI	Address		Location of rewind binary representation value in bits 18-35.

NOTE: Whenever the lock parameter is used, the file is disposed of as indicated by the disposition code specified on the device control card (see Control Cards Reference Manual). The lock parameter and a save or continue disposition code are mutually exclusive. If a file that has an S or a C disposition code is closed with lock, the lock request is ignored, the file is not rewound or released, and no indication is given to the user that the lock request was ignored.

Rules:

1. The Close macro can only be executed for a file that was opened with DOPEN.
2. The unit designation applies only to multiunit files on tape.
3. Close unit parameters
   a. No rewind means that the current unit is left in its current position following trailer label processing.
   b. Rewind means that the current unit is positioned to its physical beginning of tape marker.

c. Rewind and unload means that the current unit can be immediately removed from the device. However, this unit can be accessed again in its proper order, if the file is successfully closed and reopened.

4. Close file parameters

a. No rewind means that the current unit is left in its current position following trailer label processing. No processing is done on other units of a multiunit file.

b. Rewind means that the current unit is positioned to its physical beginning of tape marker, whereas rewind and unload means that the unit can be immediately removed from the device. No processing is done on other units of a multiunit file.

5. Close unit label processing file positioning

a. For files opened for input or I/O:

If user issues close unit, current trailer label is not checked. If UFAS detects end of unit, swapping takes place automatically and all labels are checked.

Unit switch takes place.

Next header label is checked.

Position is made to the beginning of the file space on the next unit.

b. For files opened for output or extend:

Buffers are flushed and current trailer label is built.

Unit switch takes place.

Next header label is built.

Position is made to the beginning of the file space on the next unit.

6. Close file label processing

a. If the file was opened for input or I/O, the trailer label is checked.

b. If the file was opened for output or extend, the trailer label is built.

Example

Close the file specified by the file code in the FIB with the rewind and lock parameters.

```
1      8       16
-----------------------------------------------
       DCLOS   FIBLS,LIST2,[FIBAI,ADDFIB,REW,RLFL]
       .
ADDFIB ARG     FIB
LIST2  BSS     2
       .
       .
FIB    FIBMAC  CF,ANSI, [PROAR,RECORD,0]
RECORD BSS     20
```

SINGLE-CLOSE/MACRO-DEFINED FIB LIST

Format

```
DCLOS ,, [parameter-name-1,parameter-value-1,...parameter-name-n,parameter-value-n]
```

The following parameters and respective defaults apply:

Parameters	Name	Value	Default	Comments

Same as Single Close/User-Defined Function

Rules:

1. FIB list address is not to be supplied.

2. Left bracket must be preceded by two commas.

3. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/USER-SUPPLIED FIB LIST

This function terminates the processing of one to n files. Only one FIB list is built, which is utilized by all files to be closed.

Format

```
     DCLOS  FIBLS, FIB list address , [*, FIB address,
     ETC    RewV, *, FIB address, RewV,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB address	address		Required; location of FIB, for file to be closed
Rewind Indicators			Rewind Value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and lock file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Space for the FIB list must be defined as 1+n words, where n is the number of files to be closed.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/MACRO-DEFINED FIB LIST

This function terminates the processing of from one to n files. It builds one FIB list in the user program, immediately following the DCLOS command. Storage is automatically allocated, and the list is used by all files that are closed via this function.

Format

```
DCLOS ,, [*,FIB Address, REW-V,[*,FIB Address2,REW-V,...]
```

The following parameters and respective defaults apply:

Parameters	Value	Default	Comments
FIB address	address		Required; location of FIB for file to be closed
REWIND INDICATOR			Rewind value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and unload file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Two commas must precede the left bracket enclosing file parameters.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. The FIB list address should not be supplied.

6. All Single-Close/User-Defined Function Rules apply.

SECTION III

SERIES 2000 SEQUENTIAL FILE ORGANIZATION

A Series 2000 (H2000) file contains either blocked or unblocked fixed-length records or unblocked variable-length records on 7- or 9-track tape devices, which can be accessed only sequentially. The functions available for the H2000 file formats of the sequential file organization are listed with the respective macros in Table 3-1. Files are opened, processed, and closed by UFAS in response to a FIB macro and specific functional macro requests. UFAS provides sequential H2000 file management only for single-file single-volume or single-file multi-volume files.

Table 3-1 lists the sequential H2000 file functions, the functional macros, the processing modes, and the access modes with applicable parameters for tape files.

Table 3-1. H2000 Sequential Tape File Functions and Processing Modes

MOD1 Types A and B - MOD4 Types 1 and 2			
Function	Functional Macro	Processing Mode	Access Mode Parameters Sequential
Open	DOPEN	O,I	File-Id
Put	DLPUT	O	Next Record
Get Next	DLGXT	I	Next Record
Return	UFSRET		
Wrap-up	DWRAP		
Close	DCLOS	O,I	File-Id
Info	DINFO	O,I	File-Id

2000/60 Magnetic Tape Interchange Facility

The 2000/60 Magnetic Tape Interchange Facility is a combination of hardware and software components that enable the Series 60 Level 66 user to read and write Series 2000 magnetic tape files.

Tapes to be read or written must be mounted on tape drives controlled by an MPC tape controller equipped with the Magnetic Tape Interchange Facility. This facility accommodates both GCOS and Series 2000 files.

User programs that process H2000 with UFAS must allocate tape units with either $ TAPE27 or $ TAPE29 control cards.

FILE INFORMATION BLOCK MACRO

The file information block macro provides the user with a means of communicating to the Unified File Access System the specific characteristics of a file. However, before any communication can be established with UFAS, the user must invoke the macro package with the command LODM .DMAC.

All file processing information exchanged between the user and UFAS is channeled through the FIB macro. For each file to be accessed, a FIB macro must be coded in the user program to define the characteristics of the file and the required processing parameters.

Format

The FIB macro for a H2000 sequential file has the following format:

```
FIBMAC fc,H2000,[parameter-name-1, parameter-value-1,...]
```

where:

fc - file code assigned to file

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area, file status code, and file name, but three are required for the retention period. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area) multiple symbols defining the same address must be used. Default values should not be specified for parameters that are not required, instead allow the default value to be applied automatically. For easy reference, these parameters are set in three groups as follows:

- Required parameters for all H2000 tapes

  To specify H2000 tape as medium, character set used by user program, and tape label information.

- User parameters

  To specify parameters only related to user program and independent of H2000 tapes.

- Tape format parameters

  To describe physical characteristics of H2000 tape to be processed.

The following list contains all the parameters and respective default values available for this FIB macro arranged in the order previously described:

Parameter	Name	Value	Default	Comments
Required				
External code	EXTCOD			Code set of the data on the external device. This code is used with the CISZ parameter to calculate the number of words to allocate per buffer. Also, used with the internal code set parameter to establish the size of the characters within the record and the process area, so that the records may be converted to this code set as they are moved to the buffer.
		HBCD	HBCD	Honeywell BCD (odd parity character set).
Internal code	INTCOD			Code set of the data in the process area. Records are converted to this code set as they are moved to the process area.
		ASC	ASC	ASCII character set.
		HBCD		HBCD character set.
		GBCD		GBCD character set.
Input transliteration table	INPTRN	Address		Location of a system or user-supplied table. This table is used to transliterate the characters of a record from the external code set to the internal code set, as the record is moved from the buffer to the user's process area.
			0	Transliteration not required. Records are moved from the buffer to the user's process area without transliteration. In this instance, the internal and external code sets are usually the same.
Output transliteration table	OUTTRN	Address		Location of a system or user-supplied table. This table is used to transliterate the characters of a record from the internal code set to the external code set, as the record is moved from the user's process area to the buffer.
			0	Transliteration not required. Records are moved from the user's process area (if specified) to the buffer without transliteration. In this instance, the internal and external code sets are usually the same.

Parameter	Name	Value	Default	Comments
File name	FLNAME	Address, byte 0≤byte≤3		Location of 10-byte ASCII file name to be used in header and trailer label (not to be confused with visual file name from $ TAPE card appearing in console messages).
			0,0	No file name is provided. If parameter is not required, allow default to be assumed.
Retention period	RETPER	Address, byte, length		Location of starting byte position and length of the retention period field (maximum length is 3 bytes).
			0,0,0	Number of days tape may not be overwritten without operator intervention. A retention period of 999 means tape cannot be overwritten without operator intervention.
User				
Process area	PROAR	Address, byte		Location and starting byte position of an area within the user program to which records are moved and from which they are retrieved with Put and Get macros. Byte position must be specified as zero. The size of the process area is assumed to be equal to the FIB record size.
File status code	FSCODE	Address, byte		Location and starting byte position of a two-byte ASCII status code field in which UFAS stores a unique status code whenever an exception condition occurs (see Section VIII).
			0,0	No file status code field.
Record size word	RSZWRD	Address		Location of a 36-bit binary record size field used as a subordinate to the VIB record size field. UFAS returns the record size to this field on input. On output, a record size may be provided in this field when no record size is provided with DLPUT macro.
			0	If no record size word is specified, either VIB or FIB record size is used.

Parameter	Name	Value	Default	Comments
Optional file indicator	IOPTFL			Applies only to files open for input.
		0	0	File must be present.
		1		Actual file may not be present when Open macro is executed. If not present, the first Get macro issued results in an end-of-file condition.
Checkpoint	CHKPT	Integer $\leq 2^{17}-1$		Checkpoints are taken every time this many records are accessed.
			0	No checkpoints are taken based on record count.
User error procedure	USERER	Address		Location of user-supplied error procedure to which UFAS transfers control whenever an error is detected.
			0	No user error procedure.
Error table	ERRTBL	Address		Location of user-supplied table of error procedures based on processing mode. The format of the error table is: word 0, bits 0-17: address of error procedure for files in output mode. word 1, bits 0-17: address of error procedure for files in input mode. (bits 18-35 of each word are reserved.) When an error is detected, control is transferred to the appropriate error procedure only if the user error procedure parameter is not specified. If neither a user error procedure parameter nor an error table parameter is specified, UFAS aborts when an error is detected.
			0	No error table.
Number of buffers allocated by UFAS	NBUF	Integer 1 or 2	1	

Parameter	Name	Value	Default	Comments
Tape Format				
Tape file formats	TYPE			MOD1 type A or B and MOD4 type 1 or 2 file formats.
		1.AF	1.AF	Unblocked fixed-length records.
		1.AV		Unblocked variable-length records.
		1.B		Blocked fixed-length records.
		4.1F		Unblocked fixed-length records.
		4.1V		Unblocked variable-length records.
		4.2		Blocked fixed-length records.
Record size	RSZ	Integer 1≤n≤24570	80	Number of 6-bit characters per record. Characters per record for 1.AF, 1.B, 4.1F, and 4.2. Maximum size if records are variable length (1.AV or 4.1V).
Number of records block	BLOCK	Integer≥1		Number of records per physical block on tape.
			10	Must be specified for all blocked formats and is assumed to be 1 for unblocked formats.
Banner indicator	BANNER	NONE RANGE DEFAULT dd 40≤dd≤57	NONE	User-supplied character code. Banner character separates physical blocks with data records from blocks for header or trailer labels and for checkpoint records. If the first character of a physical block does not match the banner character it is assumed that no data records are present.
Padded record	PAD	dd NONE		Padded records have all padding characters and are used to fill the last records of a physical block when file is closed. The two octal-digits dd (e.g., 77) represent a character in HBCD character set.
			77	Used for odd parity data.
			11	Used for even parity data.
Label indicator	LABELF	80 120 UNLAB		Indicates presence or absence of labels. If absent on input, no end of file is detected and no end-of-reel unit switch can occur.

Rules

1. The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner cause the open function to return an exception code to the user.

2. A banner character, when present, is the first character of a physical block and indicates if the block is a label, a data block, or the block is to be discarded (e.g., checkpoint information). The HBCD character to be checked (on input) or written (on output) is specified by the two octal digits "dd" representing the character code (e.g., 41 for J). The value 41 (J) can also be specified by entering the keyword DEFAULT. For input files, data blocks with any banner character in the range 40 (octal) ≤dd≤57 (octal) are used if the keyword RANGE is specified. If the file is not bannered, this can be specified with the keyword NONE.

3. If system transliteration tables are required, the parameter values for the input (INPTRN) and output (OUTTRN) transliteration tables must be the SYMREF of the tables. If a user-supplied transliteration table is used, the INPTRN and OUTTRN values are the addresses of the respective tables.

FUNCTIONAL MACROS

Although each one of the functional macros performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the associated parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value is assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. The only exception is when a user error or label exit procedure is entered. In this instance, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error or label procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of one or more files, verifies their existence, and establishes tables for file processing.

DOPEN consists of the following four functions:

1. Single-Open/User-Defined FIB List
2. Single-Open/Macro-Defined FIB List
3. Multiple-Open/User-Defined FIB List
4. Multiple-Open/Macro-Defined FIB List

SINGLE-OPEN/USER-DEFINED FIB LIST

Format

```
DOPEN{FIBLS,address of FIB list              } ,[parameter-name-1,
     {FIBLSI,address of pointer to FIB list  }

     parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to to a set of routines used to process the file.
	SYM	Routine package SYMREF name		Package of routines needed to process the file.

Parameter	Name	Value	Default	Comments
	SYMI	Address		Location of pointer to the routine package.
Processing mode indicator				Optional
	MODE	OT	OT	Output
		IN		Input
	MODEI	Address		Location of processing mode indicator (2 ASCII uppercase characters in bits 0-17).
Rewind indicator				
	REW	Y	Y	Rewind on open.
		N		Do not rewind on open.
	REWI	Address		Location of rewind indicator, in upper case ASCII, with Y and blanks or N and blanks in bits 0-17.

Routine Package SYMREF Name

The Unified File Access System is a flexible system that allows the user to utilize the whole system or just portions of the system that are applicable to the format of the file to be processed. In order to select the UFAS routines to be linked at load time, the user must supply a SYMREF (directory) name for the specific set of routines desired. Table 3-2 lists the file format characteristics and the corresponding directory names.

Table 3-2. Series 2000 Sequential File Format Directories

File Format	Access Mode	Processing Mode	Directory Name (SYMREF)	
			Unbannered	Bannered
MOD1				
Type A,F Unblocked FLR	Sequential	Input Output	.DI11N, .DOF1N,	.DI11B .DOF1B
Type A,V Unblocked VLR	Sequential	Input Output	.DI11N, .DOV1N,	.DI11B .DOV1B
Type B Blocked FLR	Sequential	Input Output	.DI1NN, .DOFNN,	.DI1NB .DOFNB
MOD4				
Type 1,F Unblocked FLR	Sequential	Input Output	.DI41N .DOF1N	
Type 1,V Unblocked VLR	Sequential	Input Output	.DI41N .DOV1N	
Type 2 Blocked FLR	Sequential	Input Output	.DI4NN .DOFNN	

Rules

1. To open a file for input, the Open macro uses the FIB address, the routine SYMREF address, the MODE indicator if supplied, and the FIB macro parameters. To open a file for output, the file status code (FSCODE) is used if supplied.

2. Files are rewound, unless file sequence number zero and REW=N is specified.

3. Label processing is effected, unless the permissible value UNLAB is specified in the FIB macro, as follows:

   a. When a file is opened for output, header labels are written.

   b. When a file is opened for input, header labels are checked.

4. The value 00 in the user's file status code indicates the file was opened successfully and is ready for processing. Any other value indicates that an error occurred during opening (possible contradictory FIB parameters or label processing error).

5. Open output implies file creation and open input means that an existing file is to be read.

Example

Open a H2000 file for output with rewind on open.

```
1      8       16
_____________________________________________________________________

       LODM    .DMAC
       SYMREF  .DOF1N,U.HTOA,U.ATOH
       DOPEN   FIBLS,FIBLST,[FIBA,FIB,MODE,OT,SYMI,DIRPTR,REWI,RWDIND]
         .
         .
         .
FIBLST BSS     3
RWDIND UASCI   1,Y
DIRPTR ARG     .DOF1N
FIB    FIBMAC  H2,H2000,[EXTCOD,HBCD,INTCOD,ASC,
       ETC     INPTRN,U.HTOA,OUTTRN,U.ATOH,FLNAME,FILNAM,0,
         .     RETPER,RETENP,0,1,PROAR,RECORD,0,FSCODE,STATCD,0,
         .     RSZWRD,RECSIZ,ERRTBL,ERRTBL,NBUF,2,TYPE,1.AF,RSZ,100,
       ETC     BLOCK,1,BANNER,NONE,PAD,45,LABELF,80]
FILNAM ASCII   3,H2000-FILE
RETENP ASCII   1,366
RECORD BSS     25
STATCD ASCII   00
RECSIZ DEC     100
ERRTBL ARG     ERROT
       ARG     ERRIN
ERRIN  NULL
         .
         .
         .
ERROT  NULL
         .
         .
         .
```

SINGLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of only one file. It builds FIB list in the user program immediately following the DOPEN command, and storage is allocated automatically.

Format

```
DOPEN   ,,[parameter-name-1,parameter-value-1,...parameter-name-n,
          parameter-value-n]

ETC       symref package
```

The following parameters and respective defaults apply.

Parameter	Name	Value	Default	Comments

Same as Single-Open/User-Defined Function.

Rules:

1. Do not supply FIB list address.
2. Left bracket must be preceded by two commas.
3. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/USER-SUPPLIED FIB LIST

This function initiates the processing of from one to n files. It builds one FIB list, which is used by all files.

Format

```
DOPEN   FIBLS, FIB list address, [*, FIB address, mode,

ETC     rewind indicator, symref name, *, FIB address 2,

ETC     mode 2, rewind indicator 2, symref 2,....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required: package of routines needed to process the file

Rules:

1. Space for FIB list must be defined as 1+2n words where n is the number of files to be opened.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of from one to n files. It builds one FIB list within the user program immediately following the DOPEN command. This list is used by all files. Storage for the FIB list is automatically allocated.

Format

```
DOPEN    ,,[*, FIB address, mode, rewind indicator

ETC           symref name,,......]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required: package of routines needed to process the file

Rules:

1. The macro provides space for FIB list of 1+2n words where n is the number of files to be opened.

2. Two commas must precede the left bracket.

3. Each list of parameters for every file must be preceded by an *.

4. Key words must not appear within the brackets.

5. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

6. FIB list address should not be supplied.

7. All other rules that apply to the Single-Open/User-Defined Function.

Put Macro - DLPUT

Function

The Put macro places a logical record at the end of the file; thus, the new record becomes the last record on the tape file.

Format

```
DLPUT  FIB address,VIB address,,

       [VIB-parameter-name-1,VIB-parameter-value-1,...,

       VIB-parameter-name-n,VIB-parameter-value-n]
```

NOTE: VIB must be defined as BSS 2.

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Record size				Optional Used with variable-length records, but ignored with fixed-length records.
	RSZ	Integer		Must be less than or equal to RSZ in FIB.
	RSZI	Address		Location of field containing record size in bits 0-17.
Indirect addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for output at the time of execution of this statement.

2. The Put macro uses for input the FIB address, the VIB record size if specified, the record size word if defined, the process area, and the FIB macro values specified at opening.

3. When a reel becomes full, a new reel is fetched automatically without intervention from the user program.

4. If a process area is specified in the FIB, the record is moved from the process area to the buffer. If an output transliteration table is specified, the entire record is transliterated from the internal code set to the external code set as it is moved.

5. If the record size is not specified and records have variable length, the record size in the record size word parameter is used. If no parameter is specified, the FIB record size is used.

Example

Enter a 65-character record as specified in the VIB.

```
1      8      16
_____________________________
       DLPUT  FIB,VIB,,[RSZI,VIBSIZ]
       .
       .
       .
VIB    BSS    2
VIBSIZ ZERO   65
FIB    FIBMAC H2,H2000,[INTCOD,HBCD,PROAR,RECORD,0,RSZ,65,LABELF,80]
       .
       .
       .
RECORD BSS    11
```

Get Next Macro - DLGXT

Function

The Get Next macro makes the next logical record available.

Format

```
DLGXT FIB address,VIB address,end-of-file address,

       [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 1.

The parameters and respective defaults for the Get Next macro are as follows:

Parameter	Name	Value	Default	Comments
Record size				Optional Specifies the number of characters to be moved from the record to the process area when the entire record is not required. UFAS stores the actual record size in the FIB record size word parameter.
	RSZ	Integer		Must be less than or equal to RSZ in FIB.
	RSZI	Address		Location of field containing record size in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input before execution of this statement.

2. The Get Next macro uses for input the FIB address, the VIB record size and end-of-file return address if specified, and the FIB macro values specified at the latest opening.

3. An end-of-file condition is returned when no next logical record exists in the file. This condition is considered an unsuccessful read request. Control is returned to the end-of-file address.

4. If an optional file is not present, then an end-of-file condition is returned on the first read request. The end-of-file address receives control.

5. If the first character of a physical block of a bannered file does not match the banner character, no data records are present and block is checked to verify if it has an end-of-reel or end-of-file record. If the first character of the block matches the banner character, data records are present, although one or more records may be padded.

6. Pad records may occur in physical blocks containing more than one data record (blocked files). On input, pad records are not delivered to the user program.

7. In multi-reel files, consecutive reels are fetched automatically without intervention from the user program.

8. If the record size parameter is not specified, the whole record is moved to the process area and the record size is made available to the user in bits 0-17 of word 0 of the VIB and in the word specified by the record size word parameter in the FIB.

9. If a VIB is not specified, the whole record is moved to the process area and the record size is made available to the user only if a record size word parameter is specified in the FIB.

Example

Get next record in sequential order and move 65 bytes of the record to the user process area.

```
1       8       16
_____________________________________
        DLGXT   FIB,VIB,EOFRET,[RSZI,RECSIZ]
        .
        .
        .
RECSIZ  ZERO    65
VIB     BSS     1
EOFRET  NULL
        .
        .
        .
FIB     FIBMAC  H2,H2000,[INTCOD,HBCD,PROAR,RECORD,0,TYPE,1.AV,LABELF,80]
        .
        .
        .
RECORD  BSS     14
```

Return Macro - UFSRET

Function

The Return macro provides a means of returning control to UFAS after a user error procedure is completed.

Format

```
UFSRET  ERROR
```

ERROR - Must be specified to return from a user error procedure.

NOTE: Error procedures are specified in the FIB macro.

Rule

When a user error procedure contains a UFAS macro, which can result also in a user error procedure for another file being entered by UFAS, the second procedure must conclude also with a UFSRET return for that particular function.

Example

Return to UFAS, which has issued a user error exit call to the user.

```
1      8       16
       UFSRET  ERROR
```

Wrap-up Macro - DWRAP

Function

The Wrap-up macro closes files currently open.

Format

```
DWRAP
```

Rule

All opened files are closed with rewind. Device disposition codes are not honored.

Information Macro - DINFO

FUNCTION

The DINFO macro either stores File Access Control Table (FACT) information or the size of each section of the FACT table in a user specified table area.

FORMAT

```
DINFO     FIB address, VIB address, alternate return address,
          [parameter-name-1, parameter-value-1....,
          Parameter-name-n, parameter-value-n]

          NOTE:     VIB must be defined as BSS 1
```

The parameters and respective defaults for the Information Macro are as follows:

Parameter	Name	Value	Default	Comments
Function Code	FCT	1		Store FACT section 1 into user table area. Includes words FACT-8 through FACT +98 for all file types.
		2		Store FACT section 2 into user table area. Includes words 0 through +59 for UFF Indexed files with no alternate keys: 0 through +81 for UFF Indexed with alternate keys: 0 through +17 for ANSI/IBM/GFRC and UFF tape 0 through +41 for ISP; and 0 through +9 for H2000.
		3		Store FACT section 3 into user table area. Include words 0 through 26 (dec.).
		4	4	Store the entire FACT into the user table area.
		5		Store FACT-8 through FACT +5 of section 1 into user table area. This option provides FACT information that is equivalent to the GFRC FILCB.
		6		Store the size (in binary) of FACT sections 1, 2, and 3 into a 3-word user defined table area. Zero will be given as the size of section 3 if it is non-existent.
Table Pointer	TBL	Address		Required. The location of a table large enough to contain the requested information.
Indirect FIB Addressing	FIBI	None		Optional. If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address in bits 0-17.

RULES:

1. The designated file must be opened in either the input or output.

2. The TBL parameter is required and must point to an area large enough to contain the requested information.

3. The user must provide an alternate return address.

4. If an error is detected during execution of the DINFO macro, UFAS will return control to the user via the alternate return except when executing the DINFO macro on any unopened file in which case UFAS will branch to a users error procedure if it is present or abort.

5. A function code of 6 requires a table size of 3 words. After a successful execution of a DINFO macro the table will contain the following values:

Word	Contents
1	Size of FACT section 1, right-justified. (In Binary)
2	Size of FACT section 2, right-justified. (In Binary)
3	Size of FACT section 3, right-justified. (In Binary) A size of zero will be given if section 3 is non-existent.

6. If not function code (FCT) parameter is specified the entire FACT will be given to the user.

7. The FACT sizes given in this specificaiton are applicable to SR ADF2.

8. This macro may not be executed for files processed under TP or I-D-S/II.

9. The following error conditions will be identified:

   a. 8-01 File not opened. (ALT return not taken)
   b. F0 A potential F0 memory fault during DINFO execution may take place because of insufficient table area size.
   c. F1 Invalid function code
   d. F2 The user asked for FACT section 3 and it does not exist.

10. The sizes, in decimal, of the entire FACT for the various file formats are as follows:

UFF SEQUENTIAL Tape or Mass Store	192
UFF RELATIVE	192
UFF INDEXED	192
IBM	152
ANSI	152
H2000	136
UNLT	152
GFRC Tape and Mass store	152
ISP	149

EXAMPLE

Store the entire FACT contents of the file specified by FIBI into a table specified by TABLE.

```
1      8        16
       DINFO    FIB1,VIBI,ALTRET,[FCT,4,TBL,TABLE]
       .
       .
       .
FIB1   FIBMAC   DA,UFF,[PROAR,RECORD,0,ORG,IND,
       ETC      KEYPTR,PRIME],KY
       .
       .
       .
RECORD BSS      21
PRIME  VFD      1/0,17/0,18/4
       ZERO
VIBI   BSS      1
TABLE  NULL
       .
       .
       .
ALTRET NULL
       .
       .
       .
```

Close Macro - DCLOS

Function

The Close macro terminates the processing of one or more files.

DCLOS consists of the following four functions:

1. Single-Close/User-Defined FIB List
2. Single-Close/Macro-Defined FIB List
3. Multiple-Close/User-Defined FIB List
4. Multiple-Close/Macro-Defined FIB List

SINGLE-CLOSE/USER-DEFINED FIB LIST

Format

```
DCLOS  { FIBLS,address of FIB list                 }  ,[parameter-name-1,
       { FIBLSI,address of pointer to FIB list     }

         parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Rewind indicator				Optional
	REW	RWFL	RWFL	Rewind file on close.
		NRFL		Do not rewind file. (see note).
		RLFL		Rewind and lock file (see note).
		LKFL		Lock file without rewind (see note).
		RUFL		Rewind and unload file (see note).
		R3FL		Lock file with rewind and unload (see note).
		RWUN		Rewind unit.
		NRUN		Do not rewind unit.
		RLUN		Rewind and lock unit.
		RUUN		Rewind and unload unit.
	REWI	Address		Location of rewind indicator (4 ASCII uppercase characters).
	REWV			Rewind value.
		0	0	Rewind file.
		1		Do not rewind file.
		2		Rewind and lock file.
		3		Lock file without rewind.
		4		Rewind and unload file.
		6		Lock file with rewind and unload.
		8		Rewind unit.
		9		Do not rewind unit.
		10		Rewind and lock unit.
		12		Rewind and unload unit.
	REWVI	Address		Location of rewind binary representation value in bits 18-35.

NOTE: Whenever the lock parameter is used, the file is disposed of as indicated by the disposition code specified on the device control card (see Control Cards Reference Manual). The lock parameter and a save or continue disposition code are mutually exclusive. If a file that has an S or a C disposition code is closed with lock, the lock request is ignored, the file is not rewound or released, and no indication is given to the user that the lock request was ignored.

Rules

1. The Close macro can only be executed for a file that has been opened.

2. No more than one rewind indicator parameter can be specified. If none is specified, zero is assumed.

3. Close file parameters

   No rewind means that the current unit is left in its current position following trailer label processing.

4. Close file label processing

   a. For files opened for input:

      Trailer label is checked.

      DOPEN for file is cancelled.

   b. For files opened for output:

      Buffers are flushed and current trailer label is built.

      DOPEN for file is cancelled.

5. Close unit label processing

   a. For files opened for input:

      Trailer label is not checked.

      If next reel is obtained, unit switch takes place, next header label is checked, and new reel is ready for execution of a Get Next or Close macro.

      If next reel is not obtained (current reel is last reel of file), old reel is unloaded and file is ready for execution of Get Next or Close macro.

      DOPEN is not cancelled.

   b. For files opened for output:

      Buffers are flushed and current trailer label is built.

      If next reel is obtained, unit switch takes place, new header label is built, and reel is ready for execution of a Put or Close macro.

      DOPEN is not cancelled.

Example

Close the file specified by the file code in the FIB with the rewind parameter.

```
1      8      16
       DCLOS  FIBLS,FIBLST,[FIBA,FIB,REWI,RWDTYP]
       .
       .
       .
FIBLST BSS    2
RWDTYP UASCI  1,RWFL
FIB    FIBMAC H2,H2000,[INTCOD,HBCD,PROAR,RECORD,0,LABELF,80]
       .
       .
       .
RECORD BSS    14
```

SINGLE-CLOSE/MACRO-DEFINED FIB LIST

Format

```
DCLOS  ,,[parameter-name-1,parameter-value-1,...parameter-name-n,
          parameter-value-n]
```

The following parameters and respective defaults apply:

Parameters	Name	Value	Default	Comments

Same as Single-Close/User-Defined Function

Rules:

1. FIB list address is not to be supplied.
2. Left bracket must be preceded by two commas.
3. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/USER-SUPPLIED FIB LIST

This function terminates the processing of one to n files. Only one FIB list is built, which is utilized by all files to be closed.

Format

```
DCLOS FIBLS, FIB list address ,[*, FIB address,
ETC    RewV, *, FIB address, RewV,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB address	address		Required; location of FIB, for file to be closed
Rewind Indicators			Rewind Value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and lock file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Space for the FIB list must be defined as 1+n words, where n is the number of files to be closed.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/MACRO-DEFINED FIB LIST

This function terminates the processing of from one to n files. It builds one FIB list in the user program, immediately following the DCLOS command. Storage is automatically allocated, and the list is used by all files that are closed via this function.

Format

```
DCLOS ,, [*,FIB Address, REW-V,*,FIB Address2,REW-V,...]
```

The following parameters and respective defaults apply:

Parameters	Value	Default	Comments
FIB address	address		Required; location of FIB for file to be closed
REWIND INDICATOR			Rewind value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and unload file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Two commas must precede the left bracket.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. The FIB list address should not be supplied.

6. All Single-Close/User-Defined Function Rules apply.

SECTION IV

RELATIVE FILE ORGANIZATION

A relative file must be allocated as random to a mass storage device and can be accessed sequentially, randomly, and dynamically (a combination of sequential and random modes). A record is accessed by a relative key or record number that is directly proportional to the position of the record in the file. Files are opened, processed, and closed by UFAS in response to a FIB macro and specific functional macro requests.

Positioning for sequential access is possible if the file is opened for input or input/output processing modes. Using the Get Next record function provides access to the next logical record in the file. Both the Rewrite and Delete functions can be used in the sequential mode after the Get Next record function. In the random access mode the functions Put, Rewrite, Get, and Delete require the relative key value. For the dynamic access mode, positioning is effected in the same manner as for the sequential access mode. The Get next record function retrieves the next record and the Get random record function retrieves a record based on a relative key.

Table 4-1 lists the relative file functions, the functional macros, the processing modes, and the access modes with applicable parameters for mass storage files.

Table 4-1. Relative File Functions and Processing Modes

UFF Relative Files					
			Access Mode Parameters		
Function	Functional Macro	Processing Mode	Sequential	Random	Dynamic
Open	DOPEN	O, I, I/O	File-Id	File-Id	File-Id
Put	DLPUT	O	Next Record	Relative Key	Relative Key
Put	DLPUT	I/O	(Invalid)	Relative Key	Relative Key
Rewrite	DLRWR	I/O	Current Record	Relative Key	Relative Key
Delete	DLDEL	I/O	Current Record	Relative Key	Relative Key
Position	DLPOS	I, I/O	Relative Key	(Invalid)	Relative Key
Get Next	DLGXT	I, I/O	Next Record	(Invalid)	Next Record
Get	DLGET	I, I/O	(Invalid)	Relative Key	Relative Key
Return	UFSRET				
Checkpoint Files	DCKPF	I/O			
Checkpoint Files and Program	DCKPT	I/O			
Rollback Files	DROLF	I/O			
Rollback Files and Program	DROLP	I/O			
Wrap-up	DWRAP				
Close	DCLOS		File-Id	File-Id	File-Id
Info	DINFO	O,I,I/O	File-Id	File-Id	File-Id

FILE INFORMATION BLOCK MACRO

The file information block macro enables the user to communicate the specific characteristics of a file to the Unified File Access System. However, before any communication can be established with UFAS to process a file, the user must invoke the macro package with the command LODM .DMAC. All file processing information between the user and UFAS is channeled through the FIB macro. For each file of a user program, a FIB must be coded to define the characteristics of the file and the required processing parameters.

Format

The format of the FIB macro for a relative file is as follows:

```
FIBMAC fc,UFF,[parameter-name-1,parameter-value-1,...,

parameter-name-n,parameter-value-n]
```

where:

fc - file code assigned to file

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area, file status code, and file name. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area) multiple symbols defining the same address must be used. Default values should not be specified for parameters that are not required, instead allow the default value to be applied automatically.

The following list contains all the parameters and respective defaults available for this FIB macro:

Parameter	Name	Value	Default	Comments
Number of buffers allocated by UFAS	NBUF	Integer$\leq$255	$\frac{N}{CISZ}$	If CISZ is specified, N=16384 bytes (4096 words).
			8	If CISZ is not specified.
Record size in bytes	RSZ	Integer$\leq$16128	80	Maximum record size (records are assumed to be of variable length).

Parameter	Name	Value	Default	Comments
Record size word	RSZWRD	Address		Location of a 36-bit binary record size field. Record size is returned to user on input.
			0	If no record size word is specified, either VIB or FIB record size is used.
Record size error indication	IRSZER			Indicates the action UFAS is to take when conflict occurs between the actual record size and that specified by the user in the FIB.
		0	0	Record size conflicts are ignored. The smallest record size of the actual record size and FIB record size is used as the correct record size.
		1		Execute the error procedures specified by the USERER or ERRTBL parameters.
File name	FLNAME	Address, character		Applies only to labeled files. Location and starting byte position of a 17-byte ASCII file name.
			0,0	No file name is provided.
Control interval size in bytes	CISZ	Integer≤16128	2048	
External code	EXTCOD			Code set of the data on the external device. This code is used with the CISZ parameter to calculate the number of words to allocate per buffer. Also, used with the internal code set parameter to establish the size of the characters within the record and the process area, so that the records may be converted to this code set as they are moved to the buffer.
		ASC	ASC	ASCII character set.

Parameter	Name	Value	Default	Comments
Internal code	INTCOD			Code set of the data in the process area. Records are converted to this code set as they are moved to the process area.
		ASC	ASC	ASCII character set.
		EBCD		EBCDIC character set.
		GBCD		GBCD character set.
		HBCD		HBCD character set.
		JIS		JIS character set.
Checkpoint	CHKPT	Integer $\leq 2^{17}$ -1		Checkpoints are taken every time this many records are accessed.
			0	No checkpoints are taken based on record count.
Process area	PROAR	Address, character		Location and starting character position of an area within the user program to which records are moved and from which they are retrieved with Put and Get macros. The size of the process area is assumed to be equal to the FIB record size.
File status code	FSCODE	Address, character		Location and starting byte position of a two-byte ASCII file status code field in which UFAS stores a unique status code whenever an exception condition occurs (see Section VIII).
			0,0	No file status code field.
User Error procedure	USERER	Address		Location of user-supplied error procedure to which UFAS transfers control whenever an error is detected.
			0	No user error procedure.

Parameter	Name	Value	Default	Comments
Error table	ERRTBL	Address		Location of user-supplied table of error procedures based on processing mode. The format of the error table is: word 0, bits 0-17: address of error procedure for files in output mode. word 1, bits 0-17: address of error procedure for files in input mode. word 2, bits 0-17: address of error procedure for files in input/output mode. (bits 18-35 of each word are reserved.) When an error is detected, control is transferred to the appropriate error procedure only if the user error procedure parameter is not specified. If neither a user error procedure parameter nor an error table parameter is specified, UFAS aborts when an error is detected.
			0	No user error procedures based on processing mode.
Input transliteration table	INPTRN	Address		Location of a system or user-supplied transliteration table. This table is used to transliterate the characters of a record from the external code set to the internal code set, as the record is moved from the buffer to the user's process area.
			0	Transliteration not required. Records are moved from the buffer to the user's process area without transliteration. In this instance the internal and external code sets are usually the same.

Parameter	Name	Value	Default	Comments
Output transliteration table	OUTTRN	Address		Location of a system or user-supplied transliteration table. This table is used to transliterate the characters of a record from the internal code set, as the record is moved from the user's process area to the buffer.
			0	Transliteration not required. Records are moved from the user's process area to the buffer without transliteration. In this instance, the internal and external code sets are usually the same.
File code list	FCLST	Address		If the file resides on more than one volume, this parameter defines the location of a list of file codes assigned to the second and succeeding volumes of the file. The format of the file code list is: word 0, bits 0-35: number of file codes in the list, in binary. words 1-n, bits 0-17: reserved. bits 18-35: file code in ASCII.
			0	Only one file code is assigned to the file.
File organization	ORG	REL		Must be supplied.
Access mode	ACCMOD	SEQ	SEQ	Sequential mode (sequential only).
		RAN		Random mode (random only).
		DYN		Dynamic mode (sequential and random).

Parameter	Name	Value	Default	Comments
Key type	KEYTPE			Specifies the type of key to be associated with the record. For GMAP programs the key type should be zero or default. Key types 1 through 4 are provided for COBOL and other high-level language processors.
		0	0	Key is a 36-bit binary value located in key pointer parameter.
		1		Key is numeric ASCII (fixed point).
		2		Key is BCD numeric packed decimal (fixed point).
		3		Key is 16-bit binary.
		4		Key is 32-bit binary.
Key pointer	KEYPTR	Address	0	If KEYTPE equals 0, this parameter is the location of the 36-bit binary key. If KEYTPE equals 1, 2, 3, or 4, this parameter is the location of a descriptor of the key field.
Label indicator	ILABEL			Indicates the presence or absence of labels.
		0	0	File contains labels.
		1		File does not contain labels.

Rules

1. The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner cause the open function to return an exception condition code to the activity.

2. All pointers can be reset whenever required. Hence, the user can change pointers at will with the last pointer value always prevailing.

3. If system transliteration tables are required, the parameter values for the input (INPTRN) and output (OUTTRN) transliteration tables must be the SYMREF of the tables. If a user-supplied transliteration table is used, the INPTRN and OUTTRN values are the addresses of the respective tables.

FUNCTIONAL MACROS

Although each one of the functional macros available for a relative file performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value is assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. The only exception is when a user error procedure is entered. In this instance, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of one or more files, verifies their existence, and establishes tables for file processing.

DOPEN consists of the following four functions:

1. Single-Open/User-Defined FIB List
2. Single-Open/Macro-Defined FIB List
3. Multiple-Open/User-Defined FIB List
4. Multiple-Open/Macro-Defined FIB List

SINGLE-OPEN/USER-DEFINED FIB LIST

Format

```
DOPEN ⎧ FIBLS,address of FIB list               ⎫   ,[parameter-name-1,
      ⎩ FIBLSI,address of pointer to FIB list   ⎭

       parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to a set of routines used to process the file.
	SYM	Routine package SYMREF name		Package of routines needed to process the file.
	SYMI	Address		Location of pointer to the routine package.

Parameter	Name	Value	Default	Comments
Processing mode indicator				Optional
	MODE	OT	OT	Output
		IN		Input
		IO		Input/Output
	MODEI	Address		Location of processing mode indicator (2 ASCII uppercase characters in bits 0-17).

Routine Package SYMREF Name

The Unified File Access System permits the user to utilize the system in its entirety or just portions of the system as required. Selection of the UFAS routines to be linked at load time is effected by the user with a SYMREF (directory) name that designates the specific set of routines desired. Table 4-2 lists the relative fiie format characteristics and the corresponding directory names.

Table 4-2. Relative File Format Directories

File Format	Access Mode	Processing Mode	Directory Name (SYMREF)
Relative	Sequential	I,I/O,O	.DRELS
Relative	Random	I,I/O,O	.DRELR
Relative	Dynamic	I,I/O,O	.DRELC

Rules

1. The Open macro must be successfully completed before the execution of any input/output statements for a file.

2. After the initial execution of an Open macro for a file, each subsequent DOPEN for this file must be preceded by the execution of a Close macro without the lock or unit parameters.

3. Open output must be used to create a file. During the open output processing the entire file space is initialized to zero. Open input is used to read an existing file and open input/output is used to update an existing file.

4. The Open macro does not obtain or release the first data record.

5. For files being opened for input or input/output, the DOPEN establishes the current record pointer so that the first data record in the file can be accessed.

6. For files opened for output, the current record pointer is initialized to point to the beginning of the user's data area in the logical data space.

Example

Open the relative file specified in FIBAD2 for sequential output.

```
1       8       16
_____________________________________________

        LODM    .DMAC
        SYMREF  .DRELS
        DOPEN   FIBLSI,LISTAD, FIBA,FIBAD2,SYMI,ADSYM
        .
        .
        .
LISTAD  ARG     FIBLST
FIBLST  BSS     3
FIBAD2  FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
        .
        .
        .
ADSYM   ARG     .DRELS
RECORD  BSS     20
```

SINGLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of only one file. It builds a FIB list in the user program immediately following the DOPEN command, and storage is allocated automatically.

Format

```
DOPEN     ,,[parameter-name-1,parameter-value-1,...parameter-name-n,
ETC       parameter-value-n]
```

The following parameters and respective defaults apply.

Parameter	Name	Value	Default	Comments

Same as Single-Open/User-Defined Function.

Rules:

1. Do not supply FIB list address.

2. Left bracket must be preceded by two commas.

3. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/USER-SUPPLIED FIB LIST

This function initiates the processing of from one to n files. It builds one FIB list, which is used by all files.

Format

```
DOPEN   FIBLS, FIB list address, [*, FIB address, mode,

ETC     rewind indicator, symref name, *, FIB address 2,

ETC     mode 2, rewind indicator 2, symref 2,....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required: package of routines needed to process the file

Rules:

1. Space for the FIB list must be defined as 1+2n words where n is the number of files to be opened.

2. Each list of parameters for every file <u>must</u> be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters <u>must</u> be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of from 1 to n files. It builds one FIB list within the user program immediately following the DOPEN command. This list is used by all files. Storage for the FIB list is automatically allocated.

Format

```
     DOPEN       ,,[*, FIB address, mode, rewind indicator

     ETC         symref name,*,......]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required: location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required: package of routines needed to process the file

Rules:

1. Two commas must precede the left bracket.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. FIB list address should not be supplied.
6. All other rules that apply to the Single-Open/User-Defined Function.

Put Macro - DLPUT

Function

The Put macro places a logical record in the file.

Format

```
DLPUT  FIB address,VIB address,alternate return address,
       [VIB-parameter-name-1,VIB-parameter-value-1,...,
       VIB-parameter-name-n,VIB-parameter-value-n]
```

NOTE: VIB must be defined as BSS 2.

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Record size				Optional
	RSZ	Integer		Must be less than or equal to RSZ in FIB
	RSZI	Address		Location of field containing record size in bits 0-17
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for output or I/O at the time of the execution of this statement.

2. The current record pointer is unaffected by the execution of a Put macro.

3. The maximum record size for a file is established at the time the file is created and cannot subsequently be changed.

4. When a relative file is opened in the output mode, the Put macro places records in the file as follows:

   a. If the access mode is sequential, the first record has a relative key value of 1, and subsequent records have relative key values of 2, 3, 4, etc. The relative key value is returned after the function is successfully completed (KEYPTR).

b. If the access mode is random or dynamic, the key type and key pointer parameters in the FIB must be initialized with the relative key value to be associated with the record; that is, the first PUT may specify relative key value 60, the second PUT key value 90, the third PUT key value 20, etc. The invalid key condition results when the relative key value specifies a record that already exists. Control is returned to the alternate return address.

5. When a relative file is opened for I/O and the access mode is random or dynamic, the value of the key data item to be written to the file must be initialized by the program with the relative record number. An invalid key condition results if the relative key value specifies a record that already exists. Control is returned to the alternate return address.

6. When an attempt is made to write beyond the externally defined boundaries of a relative file, an exception condition results and the file is unaffected. Control is returned to the alternate return address.

7. When the invalid key condition is recognized, the execution of the write function is unsuccessful and the contents of the record area are unaffected. Control is returned to the alternate return address.

8. If an output transliteration table is specified, the entire record is transliterated from the internal to the external code set as the record is moved to the buffer.

9. If the record size is not specified, the record size in the FIB record size word parameter is used. If no parameter is specified, the FIB maximum record size is used.

Example

Put a 27-byte record in the file specified by FIB1.

```
1      8       16
_________________________________
       DLPUT   FIB1,VIB1,ALT1,[RSZ,27]
       .
       .
       .
FIB1   FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
VIB1   BSS     2
ALT1   NULL
       .
       .
       .
RECORD BSS     20
```

Rewrite Macro - DLRWR

Function

The Rewrite macro causes the specified record to be rewritten to the file.

Format

```
DLRWR  FIB address,VIB address,alternate return address,

          [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 1.

The parameters and respective defaults for the Rewrite macro are as follows:

Parameter	Name	Value	Default	Comments
Record size				Optional Specifies the number of characters of the record to be rewritten.
	RSZ	Integer		Value must be equal to the size of the record being replaced.
	RSZI	Address		Location of record size field in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for I/O at the time of execution of this statement.

2. For files opened for sequential access, the last input/output function executed for the associated file, prior to the execution of the DLRWR macro, must have been a successfully executed DLGXT.

3. The size of the record must be equal to the size of the record being replaced.

4. The current record pointer is not affected by the execution of a DLRWR function.

5. For a relative file opened for random or dynamic access, UFAS logically replaces the record specified by the contents of the relative key. If the file does not contain the record or the record is logically deleted, the invalid key condition is returned. Control is returned to the alternate return address.

6. When the rewrite is unsuccessful, the file is unaffected.

7. If the record size is not specified, the record size in the FIB record size word parameter is used. If no parameter is specified, the FIB maximum record size is used.

Example

Rewrite a 20-byte record in the file specified in the FIB. On a logical error, return to ALT.

```
1      8      16
       DLRWR  FIB,VIB,ALT,[RSZI,ADRSZ]
       .
       .
       .
ADRSZ  ZERO   20
FIB    FIBMAC RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
VIB    BSS    1
ALT    NULL
       .
       .
       .
RECORD BSS    20
```

Delete Macro - DLDEL

Function

The Delete macro logically deletes a record from a file.

Format

DLDEL FIB address,,alternate return address,[parameter-name-1]

The parameters and respective defaults for the Delete macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open in the I/O mode at the time of the execution of this statement.

2. For a file opened in the sequential access mode, the statement executed for this file prior to the execution of the DLDEL function must have been a successful DLGXT.

3. For a relative file opened for random or dynamic access, the record identified by the relative key field (KEYPTR) in the FIB is logically deleted from the file. If the record does not exist, an invalid key condition is returned and the alternate return is taken.

4. The current record pointer is not affected by the execution of the DLDEL function.

Example

Delete the record specified in FIB2 and on a logical error execute the procedure at ALT2.

```
1       8        16
        DLDEL    FIB2,,ALT2
        .
        .
        .
FIB2    FIBMAC   RF,UFF,[PROAR,RECORD,0,ORG,REL]
        .
        .
        .
ALT2    NULL
        .
        .
        .
RECORD  BSS      20
```

Position Macro - DLPOS

Function

The Position macro provides for logically positioning the file for subsequent access.

Format

```
DLPOS   FIB address,VIB address,alternate return address,

        [VIB-parameter-name-1,VIB-parameter-value-1,
        VIB-parameter-name-2,VIB-parameter-value-2]
```

NOTE: VIB must be defined as BSS 2.

The Position macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
File beginning	BEGIN	FILE		Optional Device is positioned at beginning of file.
Backward positioning				Optional Specifies the number of logical records to be skipped over in the backward direction.
	BACK	Integer		Number of logical records to be skipped over in the backward direction.
	BACKI	Address		Location containing a 36-bit binary value representing the number of logical records to be skipped over in the backward direction.

Parameter	Name	Value	Default	Comments
Forward positioning				Optional Specifies the number of logical records to be skipped over in the forward direction.
	FORW	Integer		Number of logical records to be skipped over in the forward direction.
	FORWI	Address		Location containing a 36-bit binary value representing the number of logical records to be skipped over in the forward direction.
Random positioning				Optional Positions the file to a relative key.
	FCT	EQ	EQ	Means equal to the specified key. The key in question is described by the key type and key pointer parameters in the file information block.
		GT		Means greater than the specified key.
		GE		Means greater than or equal to the specified key.
	FCTI	Address		Location of a word having FCT value in bits 0-17 in ASCII.
Number of logical records skipped over	OUTNBI	Address		Optional Location in which a 36-bit binary value representing the number of logical records actually skipped over is returned. Only when backward or forward positioning is used.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open in the input or input/output processing mode.

2. If an end-of-file is encountered before n logical records are bypassed, the number of logical records actually skipped over is returned to the user through the OUTNBI parameter. If the contents of the OUTNBI parameter do not equal the original input parameter, the value of this parameter must be checked to determine if an end-of-file condition occurred.

3. If a beginning-of-file is encountered before n logical records are bypassed, the number of logical records actually skipped over is returned to the user through the OUTNBI parameter. If the contents of the OUTNBI parameter do not equal the original input parameter, the value of this parameter must be checked to determine if a beginning-of-file condition occurred.

4. The type of comparison specified (equal to, greater than, or equal to or greater than) occurs between the key associated with a record in the file and a data item defined by the relative key field (KEYPTR in FIB).

5. If no VIB parameter is specified, the relational operator equal to is implied by default.

6. If the comparison is not satisfied, execution is unsuccessful if the position of the current record pointer is undefined. An invalid key condition is returned. The alternate return is taken.

7. The relative key is pointed to by the KEYPTR field of word 6 of the file information block.

Examples

Rewind file.

```
1      8      16
______________________________________
       DLPOS  FIB1,VIB1,ALT1,[BEGIN,FILE]
       .
       .
       .
VIB    BSS    2
FIB1   FIBMAC RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
ALT1   NULL
       .
       .
       .
RECORD BSS    20
```

Backspace one logical record.

```
1      8       16
       DLPOS   FIB2,VIB2,ALT2,[BACK,1,OUTNBI,TRUENB]
       .
       .
       .
VIB2   BSS     2
TRUENB DEC     0
FIB2   FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
ALT2   NULL
       .
       .
       .
RECORD BSS     20
```

Skip over four logical records in the forward direction.

```
1      8       16
       DLPOS   FIB3,VIB3,ALT3,[FORWI,RECNB,OUTNBI,TRUENB]
       .
       .
       .
VIB3   BSS     2
RECNB  DEC     4
TRUENB DEC     0
FIB3   FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
ALT3   NULL
       .
       .
       .
RECORD BSS     20
```

Position a file to a specified key.

```
1      8       16
       DLPOS   FIB4,VIB4,ALT4,[FCT,EQ]
       .
       .
       .
VIB4   BSS     2
FIB4   FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL,ACCMOD,DYN]
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS     20
```

Set the current record pointer to the record with relative record number greater than or equal to the relative record number specified in the FIB. If such a record does not exist, execute the code at ALT.

```
1      8       16
______________________________________________________
       DLPOS   FIB,VIB,ALT,[FCT,GE]
       .
       .
       .
VIB    BSS     2
FIB    FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL,ACCMOD,DYN]
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS     20
```

Get Next Macro - DLGXT

Function

The Get Next macro makes the next logical record available in ascending relative key order.

Format

```
DLGXT  FIB address,,end-of-file address,[parameter-name-1]
```

The parameters and respective defaults for the Get Next macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input or I/O at the time this function is executed.

2. If the next logically sequential record was deleted, it is bypassed.

3. An end-of-file condition is returned when no next logical record exists in the file. This condition is considered an unsuccessful read request. Control is returned to the end-of-file address.

4. If the position of the current record pointer is undefined, an error condition results. The current record pointer is not defined following an unsuccessful read request.

5. The content of relative key field (KEYPTR) is updated so that it contains the relative record number of the record made available.

6. The record is moved from the buffer to the process area. If an input transliteration table is specified, the entire record is transliterated from the external to the internal code set as the record is moved.

7. The size of the record is returned to the user in the FIB record size word parameter.

Example

Get the next record. If end of file, go to ALT.

```
1      8       16
       DLGXT   FIB,,ALT
       .
       .
       .
FIB    FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS     20
```

Get Macro - DLGET

Function

The Get macro makes available a specified record based on a relative key.

Format

```
DLGET  FIB address,,alternate return address,[parameter-name-1]
```

The parameters and respective defaults for the Get macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input or I/O at the time this function is executed.

2. Following the unsuccessful execution of any DLGET function, the position of the current record pointer is undefined. The contents of the associated record area remain undisturbed.

3. The execution of a DLGET sets the current record pointer to, and makes available, the record whose relative record number is contained in the relative key field (KEYPTR) specified in the FIB. If the file does not contain such a record, an error condition exists and execution of the DLGET is unsuccessful. Control is returned to the alternate return address.

4. The record is moved from the buffer to the process area. If an input transliteration table is specified, the entire record is transliterated from the external to the internal code set as the record is moved.

5. The size of the record is returned to the user in the FIB record size word parameter.

Example

Get the record with relative record number equal to that specified on the FIB. If it is not an active record return to ALT.

```
1      8      16
       DLGET  FIB,,ALT
       .
       .
       .
FIB    FIBMAC RF,UFF,[PROAR,RECORD,0,ORG,REL,ACCMOD,RAN]
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS    20
```

Return Macro - UFSRET

Function

The Return macro provides a means of returning control to UFAS after a user error processing procedure is completed.

Format

```
UFSRET  ERROR
```

NOTE: User error procedures are specified in the FIB macro via the user error procedure and error table parameters.

Rule

When a user error procedure contains a UFAS macro, which can result also in a user error for another file being entered by UFAS, the second procedure must conclude also with a UFSRET return for that particular function.

Example

Return to UFAS after a user error exit.

```
1      8      16
______________________
       UFSRET ERROR
```

FUNCTION

The DINFO macro either stores File Access Control Table (FACT) information or the size of each section of the FACT table in a user specified table area.

FORMAT

DINFO FIB address, VIB address, alternate return address, [parameter-name-1, parameter-value-1...., parameter-name-n, parameter-value-n]

NOTE: VIB must be defined as BSS 1

The parameters and respective defaults for the Information Macro are as follows:

Parameter	Name	Value	Default	Comments
Function Code	FCT	1		Store FACT section 1 into user table area. Includes words FACT-8 through FACT +98 for all file types.
		2		Store FACT section 2 into user table area. Includes words 0 through +59 for UFF Indexed files with no alternate keys: 0 through +81 for UFF Indexed with alternate keys; 0 through + 17 for ANSI/IBM/GFRC and UFF Tape, 0 through +41 for ISP and 0 through +9 for H2000.
		3		Store FACT section 3 into user table area. Include words 0 through 26 (dec.)
		4	4	Store the entire FACT into the user table area.
		5		Store FACT-8 through FACT +5 of section 1 into user table area. This option provides FACT information that is equivalent to the GFRC FILCB.
		6		Store the size (in binary) of FACT sections 1, 2, and 3 into a 3-word user defined table area. Zero will be given as the size of section 3 if it is non-existent.
Table Pointer	TBL	Address		Required. The location of a table large enough to contain the requested information.
Indirect FIB Addressing	FIBI	None		Optional. If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address in bits 0-17.

RULES:

1. The designated file must be opened in either the input, I/O or output.

2. The TBL parameter is required and must point to an area large enough to contain the requested information.

3. The user must provide an alternate return address.

4. If an error is detected during execution of the DINFO macro, UFAS will return control to the user via the alternate return except when executing the DINFO macro on any unopened file in which case UFAS will branch to a users error procedure if it is present or abort.

5. A function code of 6 requires a table size of 3 words. After a successful execution of a DINFO macro the table will contain the following values:

Word	Contents
1	Size of FACT section 1, right-justified. (In Binary)
2	Size of FACT section 2, right-justified. (In Binary)
3	Size of FACT section 3, right-justified. (In Binary) A size of zero will be given if section 3 is non-existent.

6. If no function code (FCT) parameter is specified the entire FACT will be given to the user.

7. The FACT sizes given in this specification are applicable to SR ADF2.

8. This macro may not be executed for files processed under TP or I-D-S/II.

9. The following error conditions will be identified:

   a. 8-01 File not opened. (ALT return not taken)
   b. F0 A potential F0 memory fauilt during DINFO execution may take place because of insufficient table area size.
   c. F1 Invalid function code
   d. F2 The user asked for FACT section 3 and it does not exist.

10. The sizes, in decimal, of the entire FACT for the various file formats are as follows:

UFF SEQUENTIAL Tape of Mass Storage....	192
UFF RELATIVE..........................	192
UFF INDEXED...........................	192
IBM...................................	152
ANSI..................................	152
H2000.................................	136
UNLT..................................	152
GFRC Tape and Mass Storage............	152
ISP...................................	149

EXAMPLE

Store the entire FACT contents of the file specified by FIBI into a table specified by TABLE.

```
1      8       16
       DINFO   FIB1,VIBI,ALTRET,[FCT,4,TBL,TABLE]
       .
       .
       .
FIB1   FIBMAC  DA,UFF,[PROAR,RECORD,0,ORG,IND,
       ETC     KEYPTR,PRIME],KY
       .
       .
       .
RECORD BSS     21
PRIME  VFD     1/0,17/0,18/4
       ZERO
VIBI   BSS     1
TABLE  NULL
       .
       .
       .
ALTRET NULL
```

Checkpoint Files Macro - DCKPF

Function

The Checkpoint Files macro establishes a checkpoint for all open files.

Format

DCKPF

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.
2. All opened files are included in the checkpoint.
3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur, since the program is not checkpointed.
4. The macro performs the buffer flush before establishing the checkpoint.
5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.
6. Checkpoint is taken only if protected files are present and have been opened.

Checkpoint Files and Program Macro - DCKPT

Function

The Checkpoint Files and Program macro establishes a checkpoint for all open files and for the program.

Format

DCKPT

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.

2. All open files are included in the checkpoint.

3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur.

4. The macro performs the buffer flush before establishing the checkpoint.

5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.

6. A QX file large enough to accommodate the program must be allocated, but not declared in the user program (see Program Recovery/Restart manual).

7. Checkpoint is taken only if protected files are present and have been opened.

8. The Q register (lower 18 bits) contains the status code returned by MME GECHEK as described in the General Comprehensive Operating Supervisor manual. Zero status indicates a successful checkpoint.

Rollback Files Macro - DROLF

Function

The Rollback Files macro restores the files open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the previous checkpoint.

Format

```
DROLF
```

Rules

1. A DCKPF or DCKPT macro must be executed before performing the rollback.
2. After a rollback, the current record pointer is undefined.

Rollback Files and Program Macro - DROLP

Function

The Rollback Files and Program macro restores the program and all files that are open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the checkpoint and to restart the program from that point.

Format

```
DROLP
```

Rules

1. A DCKPT macro must be executed before a rollback is executed.

2. After a rollback, the current record pointer is undefined.

Wrap-up Macro - DWRAP

Function

The Wrap-up macro closes files currently open.

Format

DWRAP

Rule

All opened files are closed. Device disposition codes are not honored.

Close Macro - DCLOS

Function

The Close macro terminates the processing of one or more files. DCLOS consists of the following four functions:

1. Single-Close/User-Defined FIB List
2. Single-Close/Macro-Defined FIB List
3. Multiple-Close/User-Defined FIB List
4. Multiple-Close/Macro-Defined FIB List

SINGLE-CLOSE/USER-DEFINED FIB LIST

Format

```
DCLOS ⎧ FIBLS,address of FIB list             ⎫ ,[parameter-name-1,
      ⎩ FIBLSI,address of pointer to FIB list ⎭

        parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Rewind indicator				Optional
	REW	RWFL	RWFL	Rewind file on close.
		LKFL		Lock file (see note).
	REWI	Address		Location of rewind indicator (4 ASCII uppercase characters).
	REWV			Rewind value.
		0	0	Rewind file.
		3		Lock file (see note).
	REWVI	Address		Location of rewind binary representation value in bits 18-35.

NOTE: Whenever the lock parameter is used, the file is disposed of as indicated by the disposition code specified on the device control card (see Control Cards Reference Manual). The lock parameter and a save or continue disposition code are mutually exclusive. If a file that has an S or a C disposition code is closed with lock, the lock request is ignored, the file is not rewound or released, and no indication is given to the user that the lock request was ignored.

Rule

The Close macro can only be executed for a file that was opened with DOPEN.

Example

Close and lock the file specified in the FIB at ADF.

```
1       8       16
__________________________________________
        DCLOS   FIBLS,LIST3,[FIBA,ADF,REWV,3]
        .
        .
        .
LIST3   BSS     2
ADF     FIBMAC  RF,UFF,[PROAR,RECORD,0,ORG,REL]
RECORD  BSS     20
```

SINGLE-CLOSE/MACRO-DEFINED FIB LIST

Format

DCLOS ,,[parameter-name-1,parameter-value-1,...parameter-name-n, parameter-value-n]

The following parameters and respective defaults apply:

Parameters	Name	Value	Default	Comments

Same as Single-Close/User-Defined Function

Rules:

1. FIB list address is not to be supplied.
2. Left bracket must be preceded by two commas.
3. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/USER-SUPPLIED FIB LIST

This function terminates the processing of one to n files. Only one FIB list is built, which is utilized by all files to be closed.

Format

```
DCLOS  FIBLS, FIB list address ,[*, FIB address,
ETC    RewV, *, FIB address, RewV,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB address	address		Required; location of FIB, for file to be closed
Rewind Indicators			
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and lock file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Space for the FIB list must be defined as 1+n words, where n is the number of files to be closed.
2. Each list of parameters for every file <u>must</u> be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters <u>must</u> be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/MACRO-DEFINED FUNCTION

This function terminates the processing of from one to n files. It builds one FIB list in the user program, immediately following the DCLOS command. Storage is automatically allocated, and the list is used by all files that are closed via this function.

Format

```
DCLOS ,, [*,FIB Address, REW-V,*,FIB Address2,REW-V,...]
```

The following parameters and respective defaults apply:

Parameters	Value	Default	Comments
FIB address	address		Required; location of FIB for file to be closed
REWIND INDICATOR			Rewind value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and unload file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Two commas must precede the left bracket.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. The FIB list address should not be supplied.

6. All Single-Close/User-Defined Function Rules apply.

SECTION V

INDEXED FILE ORGANIZATION

A UFF indexed file must be allocated as random to a mass storage device. When the indexed file is created, the records are placed on the file in physical and logical sequential order. Each record is identified by a unique key that is entered in the prime key field. In addition, an index file is built at the same time as the data file. An indexed file can be accessed sequentially, randomly, and dynamically. Files are opened, processed, and closed by UFAS in response to a FIB macro and specific functional macro requests.

Positioning can be effected in the sequential and dynamic access modes if the file is opened for input or input/output. When the Put function is used to add records to a file, which is opened for input/output, the records are linked in logical sequential order.

An indexed file can be accessed with an alternate key by executing a DLPOS or a DLGET macro. The alternate key to be used is described by the parameters applicable to these macros. The value of the alternate key must be placed in the appropriate offset in the process area before UFAS is called. No restriction exists on the number of alternate keys that can be used.

A Get Next macro can be used to access an indexed file sequentially in accordance with an alternate key sequence. The alternate key used must have been established as the key of reference by previously executing a DLGET or DLPOS macro.

Additional buffers should be provided when using alternate keys. A minimum of five buffers for the first alternate key and more buffers for additional alternate keys increases throughput.

In an activity where an indexed file with alternate keys is being created (output processing only), UFAS performs a sort. No other sort files can be included in this activity.

Table 5-1 lists the indexed file functions, the functional macros, the processing modes, and the access modes with applicable parameters for indexed files.

Table 5-1. Indexed File Functions and Processing Modes

UFF Indexed Files					
			Access Mode Parameters		
Function	Functional Macro	Processing Mode	Sequential	Random	Dynamic
Open	DOPEN	O, I, I/O	File-Id	File-Id	File-Id
Put	DLPUT	O	Next Record	(Invalid)	Next Record
Put	DLPUT	I/O	(Invalid)	Prime Key	Prime Key
Rewrite	DLRWR	I/O	Current Record	Prime Key	Prime Key
Delete	DLDEL	I/O	Current Record	Prime Key	Prime Key
Position	DLPOS	I, I/O	Key of Reference	(Invalid)	Key of Reference
Get Next	DLGXT	I, I/O	Next Record	(Invalid)	Next Record
Get	DLGET	I, I/O	(Invalid)	Key of Reference	Key of Reference
Return	UFSRET	O,I,I/O			
Checkpoint Files	DCKPF	I/O			
Checkpoint Files and Program	DCKPT	I/O			
Rollback Files	DROLF	I/O			
Rollback Files and Program	DROLP	I/O			
Wrap-up	DWRAP	O,I,I/O			
Close	DCLOS	O,I,I/O	File-Id	File-Id	File-Id
Info	DINFO	O,I,I/O	File-Id	File-Id	File-Id

FILE INFORMATION BLOCK MACRO

The file information block macro provides the user with an efficient means of communicating to the Unified File Access System the specific characteristics of a file. However, before any communication can be established with UFAS, the user must invoke the macro package with the command LODM .DMAC. All file processing information between the user and UFAS is channeled through the FIB macro. For each file to be accessed, a FIB must be coded in the user program to define the characteristics of the file and the required processing parameters.

Format

The format of the FIB macro for a UFF indexed file is as follows:

```
     FIBMAC fc,UFF,[parameter-name-1,parameter-value-1,...,

             parameter-name-n,parameter-value-n],fci
```

where:

```
     fc  - file code assigned to data file
     fci - file code assigned to index file
```

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area, file status code, and file name. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area) multiple symbols defining the same address must be used. Default values should not be specified for parameters that are not required, instead allow the default value to be applied automatically.

The following list contains all the parameters and respective defaults available for this FIB macro:

Parameter	Name	Value	Default	Comments
Number of buffers allocated by UFAS	NBUF	3≤(minimum) Integer ≤ 255	$\frac{N}{CISZ}$	If CISZ is specified, N=16384 bytes (4096 words).
			8	If CISZ is not specified.
				Note: Additional buffers should be provided when using alternate keys. A minimum of five buffers for the first alternate key and more buffers for additional alternate keys increases throughput.
Record size in bytes	RSZ	Integer≤16096	80	Maximum record size (records are assumed to have variable length).
Control interval size in bytes or characters	CISZ	Integer≤16128	2048	
Record size word	RSZWRD	Address		Location of a 36-bit binary record size field. Record size is returned to user on input.
			0	If no record size is specified, either VIB or FIB record size is used.
Record size error indication	IRSZER			Indicates the action UFAS is to take when conflict occurs between the actual record size and that specified by the user.
		0	0	Record size conflicts are ignored. The smallest record size of the actual record size and FIB record size is used as the correct record size.
		1		Execute the error procedures specified by the USERER or ERRTBL parameters.
Checkpoint	CHKPT	Integer≤2^{17} -1		Checkpoints are taken every time this many records are accessed.
			0	No checkpoints are taken based on record count.

Parameter	Name	Value	Default	Comments
Process area	PROAR	Address, character		Location and starting character position of an area within the user program to which records are moved and from which they are retrieved with Put and Get macros. The size of the process area is assumed to be equal to the FIB record size.
File status code	FSCODE	Address, character		Location and starting byte position of a two-byte ASCII file status code field in which UFAS stores a unique status code whenever an exception condition occurs (see Section VIII).
			0,0	No file status code field.
User error procedure	USERER	Address		Location of user-supplied error procedure to which UFAS transfers control whenever an error is detected.
			0	No user error procedure.
File name	FLNAME	Address, character		Applies only to labeled files. Location and starting byte position of a 17-byte ASCII file name.
			0,0	No file name is provided.
Error table	ERRTBL	Address		Location of user-supplied table of error procedures based on processing mode. The format of the error table is: word 0, bits 0-17: address of error procedure for files in output mode. word 1, bits 0-17: address of error procedure for files in input mode. word 2, bits 0-17; address of error procedure for files in input/output mode. (bits 18-35 of each word are reserved.)

Parameter	Name	Value	Default	Comments
				When an error is detected, control is transferred to the appropriate error procedure only if the user error procedure parameter is not specified. If neither a user error procedure parameter nor an error table parameter is specified, UFAS aborts when an error is detected.
			0	No user error procedures based on processing mode.
File code	FCLST	Address		If the file resides on more than one volume, this parameter defines the location of a list of file codes assigned to the second and succeeding volumes of the file. This parameter applies only to mass storage files. The format of the file code list is: word 0, bits 0-35: number of file codes in the list, in binary. words 1-n, bits 0-17: reserved. bits 18-35: file code in ASCII.
			0	Only one file code is assigned to the file.
File organization	ORG	IND		Must be supplied.
Access mode	ACCMOD	SEQ	SEQ	Sequential mode (sequential only).
		RAN		Random mode (random only).
		DYN		Dynamic mode (sequential and random).
Percent fill	PCTFIL	Integer≤CISZ	CISZ	Number of bytes to be used in each control interval for data, as the file is being built. File must be opened for output. The remainder of the control interval is used when the file is updated. This parameter decreases overflow control interval usage as the file is updated.

Parameter	Name	Value	Default	Comments
Key pointer	KEYPTR	Address		Must be supplied on output. Location of a list of key descriptions. These descriptions define all keys to UFAS as the file is built. The list has the following format: bits 0 - indicates whether duplicate keys are allowed for this key description; if bit 0=0, this key has no duplicates; if bit 0=1, this key may have duplicates. 1-17 - key offset in record (bytes). 18-35 - key length (bytes). Each key has one entry. The list is terminated by a word of zeroes. First entry represents the prime key (bit 0 must be set off to indicate that no duplicate keys are allowed). The second entry through the nth entry represent n-1 alternate keys. Bit 0 may be set ON to indicate that an alternate may contain duplicate keys or set OFF to indicate that the alternate contains only unique keys.
			0	No key pointer specified.
Label indicator	ILABEL			Indicates the presence or absence of file labels.
		0	0	File does contain labels.
		1		File does not contain labels.

Parameter	Name	Value	Default	Comments
Inventoried local overflow	LOVINC	1 to 510	0	Controls the allocation of overflow control intervals at load time. Ratio of number of data control intervals to number of overflow control intervals. Low values will cause a larger number of control intervals to be allocated to local overflow. The $ DATA .U card can be used to override the specification of this parameter in the FIB. If this parameter is not specified or zero is specified, the total space allocated as local overflow is half the space reserved by percent fill (see Appendix E).
Inventoried general overflow	GOVINC	1 to 510	0	Controls the allocation of overflow control intervals at load time. Ratio of number of data control intervals to number of overflow control intervals. Low values will cause a larger number of control intervals to be allocated to general overflow. The $ DATA .U card can be used to override the specification of this parameter in the FIB. If this parameter is not specified or zero is specified, the default is calculated in the same manner as for LOVINC. The default for LOVINC is always calculated before the default for this parameter (see Appendix E).

Rule

The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner cause the open function to return an exception condition code to the activity.

FUNCTIONAL MACROS

Although each one of the functional macros available performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value is assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. The only exception is when a user error procedure is entered. In this instance, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of one or more files, verifies their existence, and establishes tables for file processing. DOPEN consists of the following four functions:

1. Single-Open/User-Defined FIB list

2. Single-Open/Macro-Defined FIB list

3. Multiple-Open/User-Defined FIB list

4. Multiple-Open/Macro-Defined FIB list

SINGLE-OPEN/USER-DEFINED FIB LIST

Format

```
DOPEN ⎧ FIBLS,address of FIB list              ⎫  ,[parameter-name-1,
      ⎩ FIBLSI,address of pointer to FIB list  ⎭

        parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to a set of routines used to process the file.
	SYM	Routine package SYMREF name		Package of routines needed to process the file.
	SYMI	Address		Location of pointer to the routine package.

Parameter	Name	Value	Default	Comments
Processing mode indicator				Optional
	MODE	OT	OT	Output
		IN		Input
		IO		Input/Output
	MODEI	Address		Location of processing mode indicator (2 ASCII uppercase characters in bits 0-17)

Routine Package SYMREF Name

The Unified File Access System enables the user to utilize the system in its entirety or just portions of the system as required. Selection of the UFAS routines to be linked at load time is effected by the user with a SYMREF (directory) name that designates the specific set of routines desired. Table 5-2 lists indexed file format characteristics and the corresponding directory names.

Table 5-2. UFF Indexed File Format Directories

File Format	Access Mode	Processing Mode	Directory Name (SYMREF)
Indexed	Seq, Dyn	O	.DIBLD[a] .DIBLA[b]
Indexed	Sequential	I,I/O	.DISEQ[a] .DISQA[b]
Indexed	Random	I,I/O	.DIRAN[a] .DIRNA[b]
Indexed	Dynamic	I,I/O	.DIDYN[a] .DIDYA[b]
Indexed	Dynamic	I	.DIRTV[a,c] .DIRVA[b,c]

[a] With prime keys only

[b] With prime and alternate keys

[c] When an indexed file is accessed for retrieval only, the use of these packages result in a reduction in the amount of memory used.

Rules

1. The Open macro must be successfully completed before the execution of any input/output statements for a file.

2. After the initial execution of an Open macro for a file, each subsequent DOPEN for this file must be preceded by the execution of a Close macro without the lock or unit parameters.

3. Open output is used to create an indexed file and build the indexed structure. Open input allows the indexed file to be read and open input/output allows the file to be updated.

4. The Open macro does not obtain or release the first data record.

5. For files. being opened for input or input/output, the DOPEN establishes the current record pointer so that the first data record in the file can be accessed.

6. For files opened for output, the current record pointer is initialized to point to the beginning of the user's data area in the logical data space.

7. The Open macro establishes the prime key as the key of reference.

8. If the file is empty (no data written when the file was created), attempts to retrieve records before the addition of any new records will result in the following:

   a. Attempts to retrieve records sequentially will result in an end-of-file condition.

   b. Attempts to retrive records randomly (via prime or alternate key) will result in an invalid-key condition.

Data records can be added in the prime key sequence and will be placed in the first data control interval until full. Subsequent records will be placed in overflow according to prime key sequence.

Example

Open the indexed file specified in FIBAD2 for sequential output.

```
1       8       16
_________________________________________________________
        LODM    .DMAC
        SYMREF  .DIBLD
        DOPEN   FIBLSI,LISTAD,[FIBA,FIBAD2,SYMI,ADSYM]
        .
        .
        .
LISTAD  ARG     LIST
FIBAD2  FIBMAC  .D,UFF,[PROAR,RECORD,O,ORG,IND,KEYPTR,PRIME],.I
        .
        .
        .
ADSYM   ARG     .DIBLD
LIST    BSS     3
RECORD  BSS     20
PRIME   VFD     1/0,17/0,18/4
        ZERO
```

SINGLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of only one file. It builds a FIB list in the user program immediately following the DOPEN command, and storage is allocated automatically.

Format

```
DOPEN ,, [parameter-name-1,parameter-value-1,...parameter-name-n,

ETC         parameter-value-n]
```

The following parameters and respective defaults apply.

Parameter	Name	Value	Default	Comments

Same as Single-Open/User-Defined Function.

Rules

1. Do not supply FIB list address.
2. Left bracket must be preceded by two commas.
3. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/USER-SUPPLIED FIB LIST

This function initiates the processing of from one to n files. It builds one FIB list, which is used by all files.

Format

```
DOPEN   FIBLS, FIB list address, [*, FIB address, mode,

ETC     rewind indicator, symref name, *, FIB address 2,

ETC     mode 2, rewind indicator 2, symref 2,....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required; location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (Applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required; package of routines needed to process the files

Rules

1. Space for the FIB list must be defined as 1+2n words where n is the number of files to be opened.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of from one to n files. It builds one FIB list within the user program immediately following the DOPEN command. This list is used by all files. Storage for the FIB list is automatically allocated.

Format

```
DOPEN  ,, [*, FIB address, mode, rewind indicator

ETC          symref name,*,......]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required; location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (Applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required; package of routines needed to process the files

Rules

1. Two commas must precede the left bracket.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. FIB list address should not be supplied.

6. All other rules that apply to the Single-Open/User-Defined Function.

Put Macro - DLPUT

Function

The Put macro places a logical record in the file.

Format

```
DLPUT  FIB address,VIB address,alternate return address,

       [VIB-parameter-name-1,VIB-parameter-value-1,...,

       VIB-parameter-name-n,VIB-parameter-value-n]
```

NOTE: VIB must be defined as BSS 2.

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Record size				Optional
	RSZ	Integer		Must be less than or equal to RSZ in FIB.
	RSZI	Address		Location of field containing record size in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for output or I/O at the time of the execution of this statement.

2. The current record pointer is unaffected by the execution of a DLPUT.

3. The maximum record size for a file is established at the time the file is created and cannot subsequently be changed.

4. To create a file, the file must be opened for output in the sequential or dynamic access mode. The records must be released in ascending order of prime key, or an invalid key condition is returned.

5. The value of the prime key must be unique and set by the program to the desired value prior to execution of the DLPUT function. An invalid key condition results from duplicate prime keys or from alternate keys for which duplicate keys are not allowed.

6. Records can be written to an existing indexed file in any order if opened in the random or dynamic access mode.

7. When the invalid key condition is recognized, the execution of the write function is unsuccessful and the contents of the record area are unaffected. Control is returned to the alternate return address.

8. The record is moved from the process area to the buffer. If the record size is not specified, the record size in the FIB record size word parameter is used. If no parameter is specified, the FIB maximum record size is used.

Example

Put a 27-byte record in the file specified by FIB1.

```
1      8      16
_______________________________
       DLPUT  FIB1,VIB1,ALT1,[RSZ,27]
       .
       .
       .
FIB1   FIBMAC .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
VIB1   BSS    2
ALT1   NULL
       .
       .
       .
RECORD BSS    20
PRIME  VFD    1/0,17/0,18/4
       ZERO
```

Rewrite Macro - DLRWR

Function

The Rewrite macro causes the specified record to be rewritten to the file.

Format

```
DLRWR  FIB address,VIB address,alternate return address,

         [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 1.

The parameters and resoective defaults for the Rewrite macro are follows:

Parameter	Name	Value	Default	Comments
Record size				Optional Specifies the number of characters of the record to be rewritten.
	RSZ	Integer		Value must be equal to the size of the record being replaced.
	RSZI	Address		Location of record size field in bits 0-17.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for I/O at the time of execution of this statement.

2. For files opened for sequential access, the last input/output function executed for the associated file, prior to the execution of the DLRWR function, must have been a successfully executed DLGXT. The key of reference within the data record must not be modified until the execution of DLRWR.

3. The size of the record name must be equal to the size of the record being replaced.

4. The current record pointer is not affected by the execution of a DLRWR function.

5. For an indexed file opened for random or dynamic access, UFAS logically replaces the record specified by the contents of the prime key (KEYPTR). If the file does not contain the record, an error condition is returned.

6. When the rewrite is unsuccessful, the file is unaffected.

7. When an invalid key condition is detected, control is returned to the alternate return address.

8. The content of alternate key data items of the record being rewritten may differ from that of the record being replaced. UFAS utilizes the content of the alternate key data items during the execution of the DLRWR macro to update the associated index file so that subsequent access of the data items may be effected by the specified keys of reference. If duplicate keys are not allowed for this alternate key and a duplicate key results from the attempted rewrite, an invalid key condition exists and the function is unsuccessful.

9. If the record size is not specified, the record size in the FIB record size word parameter is used. If no parameter is specified, the FIB maximum record size is used.

Example

Rewrite a 20-byte record in the file specified in the FIB. On a logical error, return to ALT.

```
1      8       16
       DLRWR   FIB,VIB,ALT, RSZI,ADRSZ
       .
       .
       .
ADRSZ  ZERO    20
FIB    FIBMAC  .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
VIB    BSS     1
ALT    NULL
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4
       ZERO
```

Delete Macro - DLDEL

Function

The Delete macro physically deletes a record from a file.

Format

DLDEL FIB address,,alternate return address, parameter-name-1

The parameters and respective defaults for the Delete macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this word is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open in the I/O mode at the time of the execution of this statement.

2. For a file opened in the sequential access mode, the statement executed for this file prior to the execution of the DLDEL function must have been a successful DLGXT.

3. For an indexed file opened for random or dynamic access, the record identified by the prime key is physically deleted from the file. If the record does not exist, an invalid key condition is returned and the alternate return is taken.

4. The current record pointer is not affected by the execution of the DLDEL function.

5. The key to be used for comparison must be in its appropriate position in the process area.

Example

Delete the record specified in FIB2 and on a logical error execute the procedure at ALT2.

```
1      8      16
       DLDEL  FIB2,,ALT2
       .
       .
       .
FIB2   FIBMAC .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
ALT2   NULL
       .
       .
       .
RECORD BSS    20
PRIME  VFD    1/0,17/0,18/4
       ZERO
```

Position Macro - DLPOS

Function

The Position macro provides for logically positioning the file for subsequent access.

Format

```
DLPOS  FIB address,VIB address,alternate return address,
       VIB-parameter-name-1,VIB-parameter-value-1,
       VIB-parameter-name-2,VIB-parameter-value-2
```

NOTE: VIB must be defined as BSS 3.

The Position macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
File beginning	BEGIN	FILE		Optional Device is positioned at beginning of file.
Forward positioning				Optional Specifies the number of logical records to be skipped over in the forward direction.
	FORW	Integer		Number of logical records to be skipped over in the forward direction.
	FORWI	Address		Location containing a 36-bit binary value representing the number of logical records to be skipped over in the forward direction.
Random positioning				Optional Positions the file to a prime key.
	FCT	EQ	EQ	Means equal to the specified key. The key in question is described by the key pointer parameter in the file information block.
		GT		Means greater than the specified key.
		GE		Means greater than or equal to the specified key.
	FCTI	Address		Location of a word having FCT value in bits 0-17 in uppercase ASCII characters.
Number of logical records skipped over	OUTNBI	Address		Optional Location in which a 36-bit binary value representing the number of logical records actually skipped over is returned. Applicable only when forward positioning is used.

Parameter	Name	Value	Default	Comments
Key offset	KEYO[1]	Integer		Byte offset from beginning of record of key to be used in comparison. Identifies key of reference and must match an offset given for a key that was described originally in KEYPTR parameter of FIB macro. If not specified, the key of reference is the prime key.
	KEYOI[1]	Address		Location of word containing the byte offset of key within the record in bits 0-17.
Key size	KEYS	Integer		Size in bytes of leftmost portion of key to be used in search. Must be less than or equal to size described for the appropriate key in KEYPTR parameter of FIB macro. Must be supplied if KEYO is used.
	KEYSI	Address		Location of word containing size of key to be used in search (bits 0-17).
Key description (offset/length)	KEYI	Address		Location of a word containing the byte offset (bits 0-17), from the beginning of the record of the key to be used, and its size (bits 18-35). This parameter is a combination of the key offset (KEYOI) and key size (KEYSI) parameters. It allows both parameters to be specified in a single word.
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

[1] If either KEYO or KEYOI is used, then KEYS or KEYSI must be supplied.

Rules

1. The designated file must be open in the input or input/output processing mode.

2. If an end-of-file is encountered before n logical records are bypassed, the number of logical records actually skipped over is returned to the user through the OUTNBI parameter; if the contents of the OUTNBI parameter do not equal the original input parameter, the value of the status field must be checked to determine if an end-of-file condition occurred.

3. The type of comparison specified (equal to, greater than, or equal to or greater than) occurs between the key associated with a record in the file and a data item defined by the prime key field (KEYPTR in FIB).

4. If no VIB parameter is specified, the relational operator EQ (equal to) is implied by default.

5. A successful execution of the DLPOS macro causes the key of reference to be established. If an alternate key is used for the comparison, the current record pointer is positioned to the first logical record currently existing in the file whose key satisfies the comparison.

6. If the comparison is not satisfied, execution is unsuccessful, the position of the current record pointer and the key of reference are undefined. An invalid key condition is returned. The alternate return is taken.

7. The key to be used for comparison must be in the appropriate position in the process area and is described in word 6 of the file information block.

8. The capability of skipping over n logical records in the backward direction is not provided.

Examples

Rewind file.

```
1      8       16
__________________________________________
       DLPOS   FIB1,VIB1,ALT1,[BEGIN,FILE]
       .
       .
       .
VIB    BSS     3
FIB1   FIBMAC  .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
ALT1   NULL
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4,36/0
```

Skip over four logical records in the forward direction.

```
1      8       16
       DLPOS   FIB3,VIB3,ALT3,[FORWI,RECNB,OUTNBI,TRUENB]
       .
       .
       .
VIB3   BSS     3
RECNB  DEC     4
TRUENB DEC     0
FIB3   FIBMAC  .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
ALT3   NULL
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4,36/0
```

Position a file to a specified alternate key whose length is 20 bytes and offset within the record is 10 bytes.

```
1      8       16
       DLPOS   FIB4,VIB4,ALT4,[KEYO,10,KEYS,20]
       .
       .
       .
VIB4   BSS     3
FIB4   FIBMAC  .D,UFF,[PROAR,RECORD,0,ORG,IND,ACCMOD,DYN,KEYPTR,PRIME],.I
       .
       .
       .
ALT4   NULL
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4
       VFD     1/1,17/10,18/20
       ZERO
```

Set the current record pointer to the record with indexed key value greater than or equal to the indexed key value specified in the FIB. If such a record does not exist, execute the code at ALT.

```
1      8       16
       DLPOS   FIB,VIB,ALT, [FCT,GE]
       .
       .
       .
VIB    BSS     3
FIB    FIBMAC  .D,UFF,[PROAR,RECORD,0,ORG,IND,ACCMOD,DYN,KEYPTR,PRIME],.I
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4
       ZERO
```

Get Next Macro - DLGXT

Function

The Get Next macro makes the next logical record available in ascending key order.

Format

DLGXT FIB address,,end-of-file address, parameter-name-1

The parameters and respective defaults for the Get Next macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this keyword is specifified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input or I/O at the time this function is executed.

2. An end-of-file condition is returned when no next logical record exists in the file. This condition is considered an unsuccessful read request. Control is returned to the end-of-file address.

3. If the position of the current record pointer is undefined, an error condition results. The current record pointer is not defined following an unsuccessful read request.

4. The order of retrieval is the ascending order of the key of reference (prime key or an alternate key if one is present). If the key of reference is an alternate key and duplicate keys are present, the records are retrieved in the same order as they were written in the file (first in, first out).

5. Records with the same duplicate key values of an alternate key, that is the key of reference, are accessed in the same order that they were released by execution of a DLPUT or DLRWR macro that created the duplicate keys.

6. If no DLPOS or DLGET macro was executed, then the key of reference is the prime key. Once the key of reference is established, it remains in effect for subsequent executions of the DLGXT macro until another key of reference is established for this file.

7. The record is moved from the buffer to the process area. The size of the record is returned to the user in the FIB record size word parameter.

8. The key of reference can be pre-established via execution of a DLGET or a DLPOS macro. If neither macro has been executed, the prime key is used.

Example

Get the next record. If end of file, go to ALT.

```
1      8      16
       DLGXT  FIB,,ALT
       .
       .
       .
FIB    FIBMAC .D,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS    20
PRIME  VFD    1/0,17/0,18/4
       ZERO
```

Get Macro - DLGET

Function

The Get macro makes available a specified record based on a prime key or alternate key.

Format

```
DLGET  FIB address,VIB address,alternate return address,
       [VIB-parameter-name-1,VIB-parameter-value-1,
       VIB-parameter-name-2,VIB-parameter-value-2]
```

NOTE: VIB must be defined as BSS 3.

The Get macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Key offset	KEYO[1]	Integer		Key offset and key size are used to identify the alternate key that is to be used to retrieve the record. If not specified, the prime key is used. The key size parameter may be used for partial key searching. Byte offset from beginning of record of key to be used in comparison. Identifies key of reference and must match an offset given for a key that was described originally in KEYPTR parameter of FIB macro.
	KEYOI[1]	Address		Location of word containing byte offset of key in record (bits 0-17).
Key size				Must be less than or equal to size described for appropriate kcv in KEYPTR parameter of FIB macro. Must be supplied if KEYO or KEYOI is used.
	KEYS	Integer		Size in bytes of leftmost portion of key to be used in search.
	KEYSI	Address		Location of word containing size of key to be used in search (bits 0-17).
Key description (offset/length)	KEYI	Address		Location of a word containing the byte offset (bits 0-17), from the beginning of the record of the key to be used, and its size (bits 18-35). This parameter is a combination of the key offset (KEY01) and key size (KEYSI) parameters. It allows both parameters to be specified in a single word.

[1] If either KEYO or KEYOI is used, then KEYS or KEYSI must be supplied.

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for input or I/O at the time this function is executed.

2. Following the unsuccessful execution of any DLGET function, the position of the current record pointer is undefined. The contents of the associated record area remain undisturbed, and the prime key and key of reference are undefined.

3. The execution of a DLGET causes the value of the key of reference to be compared with the value contained in the corresponding data item of the stored records in the file, until the first record having an equal value is found. The current record pointer is positioned to this record, which is then made available. If no record exists, execution of the function is unsuccessful. The alternate return is taken.

4. The key to be used for comparison must be in its appropriate position in the process area.

5. The record is moved from the buffer to the process area. The size of the record is returned to the user in the FIB record size word parameter.

Example

Get the record with indexed key value equal to that specified in the FIB. If it is not an active record return to ALT.

```
1      8       16
       DLGET   AA,VIB,ALT,[KEY0,30,KEYS,10,FIBI]
       .
       .
       .
FIB    FIBMAC  .D,UFF, [PROAR,RECORD,0,ORG,IND,ACCMOD,RAN,KEYPTR,PRIME],
       ETC     .I
       .
       .
       .
ALT    NULL
       .
       .
       .
AA     ARG    FIB
VIB    BSS     3
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4
       VFD     1/1,17/10,18/20
       VFD     1/0,17/30,18/10
       ZERO
```

Return Macro - UFSRET

Function

The Return macro provides a means of returning control to UFAS after a user error processing procedure is completed.

Format

```
UFSRET  ERROR
```

NOTE: User error procedures are specified in the FIB macro via the user error procedure and error table parameters.

Rule

When a user error procedure contains a UFAS macro, which can result also in a user error procedure for another file being entered by UFAS, the second procedure must conclude also with a UFSRET return for that particular function.

Example

Return to UFAS after a user error exit.

```
1      8      16
______________________________________
       UFSRET ERROR
```

Information Macro DINFO

Function

The DINFO macro either stores File Access Control Table (FACT) information or the size of each section of the FACT table into a user specified table area.

Format

```
DINFO  FIB address, VIB address, alternate return address,

       [parameter-name-1, parameter-value-1....,

       parameter-name-n, parameter-value-n]

       NOTE:  VIB must be defined as BSS 1
```

The parameters and respective defaults for the Information Macro are as follows:

Parameter	Name	Value	Default	Comments
Function	FCT	1		Store FACT section 1 into user table area. Includes words FACT-8 through FACT +98 for all file types.
		2		Store FACT section 2 into user table area. Includes words 0 through +59 for UFF Indexed files with no alternate keys: 0 through +81 for UFF Indexed with alternate keys; 0 through +17 for ANSI/IBM/GFRC and UFF tape; 0 through +41 for ISP; and 0 through +9 for H2000.
		3		Store FACT section 3 into user table area. Include words 0 through 26 (dec).
		4	4	Store the entire FACT into the user table area.
		5		Store FACT-8 through FACT +5 of section 1 into user table area. This option provides FACT information that is equivalent to the GFRC FILCB.
		6		Store the size (in binary) of FACT sections 1, 2, and 3 into a 3-word user defined table area. Zero will be given as the size of section 3 if it is non-existent.

Parameter	Name	Value	Default	Comments
Table Pointer	TBL	Address		Required. If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address in bits 0-17.

Rules:

1. The designated file must be opened in either the input, I/O or output.

2. The TBL parameter is required and must point to an area large enough to contain the requested information.

3. The user must provide an alternate return address.

4. If an error is detected during execution of the DINFO macro, UFAS will return control to the user via the alternate return except when executing the DINFO macro on any unopened file in which case UFAS will branch to a users error procedure if it is present or abort.

5. A function code of 6 requires a table size of 3 words. After a successful execution of a DINFO macro the table will contain the following values:

Word	Contents
1	Size of FACT section 1, right-justified. (In Binary)
2	Size of FACT section 2, right-justified. (In Binary)
3	Size of FACT section 3, right-justified. (In Binary) A size of zero will be given if section 3 is non-existent.

6. If no function code (FCT) parameter is specified the entire FACT will be given to the user.

7. The FACT sizes given in this specification are applicable to SR ADF2.

8. This macro may not be executed for files processed under TP or I-D-S/II.

9. The following error conditions will be identified:

a.	8-01	File not opened. (ALT return not taken)
b.	F0	A potential F0 memory fault during DINFO execution may take place because of insufficient table area size.
c.	F1	Invalid function code
d.	F2	The user asked for FACT section 3 and it does not exist.

10. Listed below in decimal are the sizes of the entire FACT for the various file formats.

UFF SEQUENTIAL Tape and Macro Store	192
UFF RELATIVE	192
UFF INDEXED	192
IBM	152
ANSI	152
H2000	136
UNLT	152
GFRC Tape and Macro Store	152
ISP	149

Example

Store the entire FACT contents of the file specified by FIBI into a table specified by TABLE.

```
1        8        16
______________________________________________________
         DINFO    FIB1,VIBI,ALTRET,[FCT,4,TBL,TABLE]
         .
         .
         .
FIB1     FIBMAC   DA,UFF,[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],KY
         .
         .
         .
RECORD   BSS      21
PRIME    VFD      1/0,17/0,18/4
         ZERO
VIBI     BSS      1
TABLE    NULL
         .
         .
         .
ALTRET   NULL
         .
         .
         .
```

Checkpoint Files Macro - DCKPF

Function

The Checkpoint Files macro establishes a checkpoint for all open files.

Format

```
DCKPF
```

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.
2. All opened files are included in the checkpoint.
3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur, since the program is not checkpointed.
4. The macro performs a buffer flush before establishing the checkpoint.
5. For any files being processed with ACCESS/MONITOR/ (see *File Management Supervisor* manual), buffers are set empty and the record pointer is set to undefined.
6. Checkpoint is taken only if protected files are present and have been opened.

Checkpoint Files and Program Macro - DCKPT

Function

The Checkpoint Files and Program macro establishes a checkpoint for all open files and for the program.

Format

DCKPT

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.
2. All open files are included in the checkpoint.
3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur.
4. The macro performs a buffer flush before establishing the checkpoint.
5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.
6. A QX file large enough to accommodate the program must be allocated, but not declared in the user program (see Program Recovery/Restart manual).
7. Checkpoint is taken only if protected files are present and have been opened.
8. The Q register (lower 18 bits) contains the status code returned by MME GECHEK as described in the General Comprehensive Operating Supervisor manual. Zero status indicates a successful checkpoint.

Rollback Files Macro - DROLF

Function

The Rollback Files macro restores the files open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the previous checkpoint.

Format

```
DROLF
```

Rules

1. A DCKPF or DCKPT macro must be executed before performing the rollback.

2. After a rollback, the current record pointer is undefined.

Rollback Files and Program Macro - DROLP

Function

The Rollback Files and Program macro restores the program and all files that are open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the checkpoint and to restart the program from that point.

Format

```
DROLP
```

Rules

1. A DCKPT macro must be executed before a rollback is executed.

2. After a rollback, the current record pointer is undefined.

Wrap-up Macro - DWRAP

Function

The Wrap-up macro closes files currently open.

Format

```
DWRAP
```

Rule

All opened files are closed. Device disposition codes are not honored.

Close Macro - DCLOS

Function

The Close macro terminates the processing of one or more files.

DCLOS consists of the following four functions:

1. Single-Close/User-Defined FIB List
2. Single-Close/Macro-Defined FIB List
3. Multiple-Close/User-Defined FIB List
4. Multiple-Close/Macro-Defined FIB List

SINGLE-CLOSE/USER-DEFINED FIB LIST

Format

```
DCLOS ⎧FIBLS,address of FIB list               ⎫ [parameter-name-1,
      ⎩FIBLSI,address of pointer to FIB list   ⎭

       parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB
Rewind indicator				Optional
	REW	RWFL	RWFL	Rewind file on close.
		LKFL		Lock file (see note).
	REWI	Address		Location of rewind indicator (4 ASCII uppercase characters).
	REWV			Rewind value.
		0	0	Rewind file.
		3		Lock file (see note).
	REWVI	Address		Location of rewind binary representation value in bits 18-35.

NOTE: Whenever the lock parameter is used, the file is disposed of as indicated by the disposition code specified on the device control card (see Control Cards Reference Manual). The lock parameter and a save or continue disposition code are mutually exclusive. If a file that has an S or C disposition code is closed with lock, the lock request is ignored, the file is not rewound or released, and no indication is given to the user that the lock request was ignored.

Rules:

1. The Close macro can only be executed for a file that was opened with DOPEN.

2. If the file was opened for sequential access in the output mode and no data exists in the file, UFAS produces a dummy record of all high values. Although the record does not occupy space on the data file, it ensures that the appropriate index entries (prime and alternate keys) are written to the index file. When opened subsequently, records can be added to the data file.

Example

Close and lock the file specified in the FIB at ADF.

```
1      8       16
       DCLOS   FIBLS,LIST3,[FIBA,ADF,REWV,3]
       .
       .
       .
LIST3  BSS     2
ADF    FIBMAC  .D,UFF[PROAR,RECORD,0,ORG,IND,KEYPTR,PRIME],.I
       .
       .
       .
RECORD BSS     20
PRIME  VFD     1/0,17/0,18/4
       ZERO
```

SINGLE-CLOSE/MACRO-DEFINED FIB LIST

Format

DCLOS ,, [parameter-name-1,parameter-value-1,...parameter-name-n,
parameter-value-n]

The following parameters and respective defaults apply:

Parameters	Name	Value	Default	Comments

Same as Single-Close/User-Defined Function

Rules

1. FIB list address is not to be supplied.
2. Left bracket must be preceded by two commas.
3. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/USER-SUPPLIED FIB LIST

This function terminates the processing of one to n files. Only one FIB list is built, which is utilized by all files to be closed.

Format

```
DCLOS  FIBLS, FIB list address ,[*, FIB address,
ETC    RewV, *, FIB address, RewV,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB address	address		Required; location of FIB, for file to be closed
Rewind Indicators			
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and lock file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules

1. Space for the FIB list must be defined as 1+n words, where n is the number of files to be closed.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/MACRO-DEFINED FIB LIST

This function terminates the processing of from one to n files. It builds one FIB list in the user program, immediately following the DCLOS command. Storage is automatically allocated, and the list is used by all files that are closed via this function.

Format

```
DCLSO ,, [*, FIB Address, REW-V,*,FIB Address2,REW-V,...]
```

The following parameters and respective defaults apply:

Parameters	Value	Default	Comments
FIB address	address		Required; location of FIB for file to be closed
REWIND INDICATOR			Rewind value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and unload file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules

1. Two commas must precede the left bracket.
2. Each list of parameters for every file must be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. The FIB list address should not be supplied.
6. All Single-Close/User-Defined Function Rules apply.

SECTION VI

INDEXED SEQUENTIAL PROCESSOR FILE ORGANIZATION

An Indexed Sequential Processor (ISP) file can contain both variable-length and fixed-length records. Each record is identified by a unique key that was entered in the prime key field when the record was created. The data file must reside on random mass storage and can be accessed sequentially, randomly, and dynamically. Files are opened, processed, and closed by UFAS in response to a FIB macro and specific functional macro requests. UFAS provides ISP file management, but does not provide the means to create ISP files.

Positioning can be effected in the sequential and dynamic access modes if the file is opened for input or input/output. When the Put macro is used to add records to a file the records are linked in the logical sequential order.

Table 6-1 lists the ISP file functions, the functional macros, the processing modes, and the access modes with applicable parameters for mass storage files.

Table 6-1. ISP File Functions and Processing Modes

ISP Files					
			Access Mode Parameters		
Function	Functional Macro	Processing Mode	Sequential	Random	Dynamic
Open	DOPEN	I,I/O	File-Id	File-Id	File-Id
Put	DLPUT	I/O	(Invalid)	Prime Key	Prime Key
Rewrite	DLRWR	I/O	Current Record	Prime Key	Prime Key
Delete	DLDEL	I/O	Current Record	Prime Key	Prime Key
Position	DLPOS	I,I/O	Prime Key	(Invalid)	Prime Key
Get Next	DLGXT	I,I/O	Next Record	(Invalid)	Next Record
Get	DLGET	I,I/O	(Invalid)	Prime Key	Prime Key
Return	UFSRET	I,I/O			
Checkpoint Files	DCKPF	I/O			
Checkpoint Files and Program	DCKPT	I/O			
Rollback Files	DROLF	I/O			
Rollback Files and Program	DROLP	I/O			
Wrap-up	DWRAP	I,I/O			
Close	DCLOS	I,I/O	File-Id	File-Id	File-Id
Info	DINFO	I,I/O	File-Id	File-Id	File-Id

FILE INFORMATION BLOCK MACRO

The file information block macro provides the user with an efficient means of communicating to the Unified File Access System the specific characteristics of a file. However, before any communication can be established with UFAS, the user must invoke the macro package with the command LODM .DMAC. All file processing information between the user and UFAS is channeled through the FIB macro. For each file to be accessed, a FIB must be coded in the user program to define the characteristics of the file and the required processing parameters. Some of the file characteristics of an ISP file are contained in the utilization record stored in the first page of both the data file and the index file.

Format

The format of the FIB macro for an ISP file is as follows:

```
FIBMAC fc,ISP,[parameter-name-1,parameter-value-1,...,

parameter-name-n,parameter-value-n],fci
```

where:

```
fc  - file code assigned to data file
fci - file code assigned to index file
```

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area, file status code, and file name. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area) multiple symbols defining the same address must be used. Default values should not be specified for parameters that are not required, instead allow the default value to be applied automatically.

The following list contains all the parameters and respective defaults available for this FIB macro:

Parameter	Name	Value	Default	Comments
Number of buffers allocated by UFAS	NBUF	3≤(Minimum) Integer≤255	$\frac{N}{CISZ}$	If CISZ is specified, N=24192 characters (4032 words).
			12	If CISZ is not specified.

Parameter	Name	Value	Default	Comments
Record size in characters	RSZ	Integer≤6144	0	Maximum record size. Must be less than or equal to the value used when the file was created.
			0	Use value from data file utilization record.
Record size word	RSZWRD	Address		Location of word containing number of characters in record. Record size to write with a Put macro. With a Get macro, UFAS returns size of record moved to process area.
			0	If no record size is specified, FIB record size is used with Put macro. With Get macro, the smallest record size of the actual record size and FIB record size is used.
Record size error indication	IRSZER			Indicates the action UFAS is to take when conflict occurs between the actual record size and that specified by the user.
			0	Record size conflicts are ignored. The smallest record size of the actual record size or FIB record size is used as the correct record size.
		1		Execute the error procedures specified by the USERER or ERRTBL parameters.
Control interval size in characters	CISZ	1920≤n≤24192		Number of characters in each page of the data file.
			0	Use values from data file utilization record.
Checkpoint	CHKPT	Integer≤2^{17} -1		Checkpoints are taken every time this many records are accessed. Does not apply with FMS ABORT/ROLLBACK/.
			0	No checkpoints are taken based on record count.

Parameter	Name	Value	Default	Comments
Process area	PROAR	Address, character		Location and starting character position of an area within the user program to which records are moved and from which they are retrieved with Put and Get macros. The size of the process area is assumed to be equal to the FIB record size and must start on a word boundary (character=0).
File status code	FSCODE	Address, character		Location and starting byte position of a two-byte ASCII file status code field in which UFAS stores a unique status code whenever an exception condition occurs (see Section VIII).
			0,0	No file status code field.
User error procedure	USERER	Address		Location of user-supplied error procedure to which UFAS transfers control whenever an error is detected.
			0	No user error procedure specified.
Error table	ERRTBL	Address		Location of user-supplied table of error procedures based on processing mode. The format of the error table is: word 0, bits 0-17: must be zero. word 1, bits 0-17: address of error procedure for files in input mode. word 2, bits 0-17: address of error procedure for files in input/output mode. (bits 18-35 of each word are reserved.) When an error is detected, control is transferred to the appropriate error procedure only if the user error procedure parameter is not specified. If neither a user error procedure parameter nor an error table parameter is specified, UFAS aborts when an error is detected.

Parameter	Name	Value	Default	Comments
			0	No user error procedures based on processing mode.
File code list	FCLST	Address		Location of a list of file codes assigned to the second and succeeding units of the data file. A maximum of 7 additional file codes may be specified. The format of the file code list is: word 0, bits 0-35: number of entries in the list, in binary. words 1-n, bits 0-17: reserved. bits 18-35: file code in ASCII.
			0	Only one file code is assigned to the file.
Access mode	ACCMOD	SEQ	SEQ	Sequential mode (sequential only).
		RAN		Random mode (random only).
		DYN		Dynamic mode (sequential and random).
Key pointer	KEYPTR	Address		Location of key list, which is a one-word entry followed by zeros. Key entry contains key off-set from beginning of record in bits 0-17. Bits 18-35 contain the key length.
			0	Key from data file utilization record is used.

Rules

1. The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner cause the open function to abort the activity automatically.

2. All pointers are set with bits 0-17 of the specified location. Values assigned to characters, such as number of characters or character offset, are based on 6-bit characters.

3. All pointers can be reset whenever required. Hence, the user can change pointers at will with the last pointer value always prevailing.

FUNCTIONAL MACROS

Even though each one of the functional macros available performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value are assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. The only exception is when a user error procedure is entered. In this instance, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of one or more files, verifies their existence, and establishes tables for file processing. DOPEN consists of the following four functions:

1. Single-Open/User-Defined FIB List
2. Single-Open/Macro-Defined FIB List
3. Multiple-Open/User-Defined FIB List
4. Multiple-Open/Macro-Defined FIB List

SINGLE-OPEN/USER-DEFINED FIB LIST

Format

```
DOPEN { FIBLS,address of FIB list              } ,[parameter-name-1,
      { FIBLSI,address of pointer to FIB list  }

      parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to a set of routines used to process the file.
	SYM	Routine package SYMREF name		Package of routines needed to process the file.
	SYMI	Address		Location of pointer to the routine package.

Parameter	Name	Value	Default	Comments
Processing mode indicator				Required
	MODE	IN		Input.
		IO		Input/Output.
	MODEI	Address		Location of processing mode indicator (2 ASCII upper-case characters in bits 0-17).

Routine Package SYMREF Name

The Unified File Access System enables the user to utilize the system in its entirety or just portions of the system as required. Selection of the UFAS routines to be linked at load time is effected by the user with a SYMREF (directory) name that designates the specific set of routines desired. Table 6-2 lists the ISP file format characteristics and the corresponding directory names.

Table 6-2. ISP File Format Directories

File Format	Access Mode	Processing Mode	Directory Name (SYMREF)
ISP	Sequential Random Dynamic	I,I/O	.DISP
ISP	Sequential (without positioning function)	I,I/O	.DISPS

Rules

1. The Open macro must be successfully completed before the execution of any input/output statements for a file.

2. After the initial execution of an Open macro for a file, each subsequent DOPEN for this file must be preceded by the execution of a Close macro without the lock or unit parameters.

3. Only open input (read) or open input/output (update) are valid processing modes. Open output (file create) is not allowed for this file format.

4. The Open macro does not obtain or release the first data record.

5. DOPEN establishes the current record pointer so that the first data record in the file can be accessed.

6. When an ISP file is opened, the collating sequence and the index page size are determined from the utilization record in the data file.

7. Values for maximum record size, data file page size, and key offset and length are obtained from the utilization records in the data and index files. UFAS applies the value from the utilization records for the preceding parameters, whenever the corresponding value in the FIB is the same as that in the utilization records or is not specified (zero).

The following table shows the action UFAS takes when the corresponding values are not equal.

Parameter	FIB Value i Versus Utilization Record Value v	Values Selected
Maximum Record size	i>v i<v	Activity is aborted. UFAS applies i as maximum record size and continues.
Data page size	i ≠ v	UFAS applies v as data page size and continues.
Key offset and length	i ≠ v	Activity is aborted.

Example

Open the ISP file specified in FIBAD2 for sequential input without the Position function.

```
1         8         16
_____________________________________________________________________
          LODM      .DMAC
          SYMREF    .DISPS
          DOPEN     FIBLS,LISTAD,[FIBA,FIBAD2,SYMI,ADSYM,MODE,IN]
          .
          .
          .
LISTAD    BSS       3
FIBAD2    FIBMAC    D.,ISP,[PROAR,RECORD,0],I.
          .
          .
          .
ADSYM     ARG      .DISPS
RECORD    BSS       14
```

SINGLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of only one file. It builds a FIB list in the user program immediately following the DOPEN command, and storage is allocated automatically.

Format

```
DOPEN   ,,[parameter-name-1,parameter-value-1,...parameter-name-n,
          parameter-value-n]

ETC
```

The following parameters and respective defaults apply.

Parameter	Name	Value	Default	Comments

Same as Single-Open/User-Defined Function.

Rules

1. Do not supply FIB list address.
2. Left bracket must be preceded by two commas.
3. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/USER-SUPPLIED FIB LIST

This function initiates the processing of from one to n files. It builds one FIB list, which is used by all files.

Format

```
DOPEN   FIBLS, FIB list address,  [*, FIB address, mode,
ETC     rewind indicator, symref name, *, FIB address 2,
ETC     mode 2, rewind indicator 2, symref 2,....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required; location of FIB for file to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (Applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required; package of routines needed to process the files

Rules

1. Space for the FIB list must be defined as 1+2n words where n is the number of files to be opened.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. All other rules that apply to the Single-Open/User-Defined Function.

MULTIPLE-OPEN/MACRO-DEFINED FIB LIST

This function initiates processing of from one to n files. It builds one FIB list within the user program immediately following the DOPEN command. This list is used by all files. Storage for the FIB list is automatically allocated.

Format

```
DOPEN  ,,[*, FIB address, mode, rewind indicator

ETC         symref name,,*.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB Address	Address		Required; location of FIB for files to be opened
PROCESSING MODE	OT	OT	Output
	IN		Input
	IO		Input/Output (applies to mass storage)
	EX		EXTEND (does not apply to GFRC on mass storage)
REWIND Indicator	Y	Y	Rewind Option
	N		DO NOT REWIND ON OPEN
Routine Package	Symref name		Required package of routines needed to process the files

Rules

1. Two commas must precede the left bracket.

2. Each list of parameters for every file must be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters must be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. FIB list address should not be supplied.

6. All other rules that apply to the Single-Open/User-Defined Function.

Put Macro - DLPUT

Function

The Put macro adds a logical record to the file.

Format

```
DLPUT  FIB address,,alternate return address,[parameter-name-1]
```

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for I/O at the time of the execution of this statement.

2. The current record pointer is unaffected by the execution of a DLPUT. After a DLPUT, the current record pointer can be set by executing a DLPOS macro with GT parameter value.

3. The maximum record size for a file is established at the time the file is created and cannot subsequently be changed. A record size smaller than the maximum (RSZ) can be specified in the record size word (RSZWRD).

4. The value of the key must be unique and set by the program to the desired value prior to execution of the DLPUT macro. An invalid key condition results from duplicate keys. The key must have the same data type and code set as the key values in the file.

5. Records can be added to an existing ISP file in any order.

6. When the invalid key condition is recognized, the execution of the write function is unsuccessful and the contents of the record area are unaffected. Control is returned to the alternate return address.

7. The record is moved from the process area to the buffer. If the record size word parameter is not specified, the maximum record size is used.

Example

Put a record in the file specified by FIB1.

```
1      8      16
       DLPUT  FIB1,,ALT1
       .
       .
       .
FIB1   FIBMAC D.,ISP,[PROAR,RECORD,0],I.
       .
       .
       .
ALT1   NULL
       .
       .
       .
RECORD BSS    14
```

Rewrite Macro - DLRWR

Function

The Rewrite macro causes the specified record to be rewritten to the file.

Format

DLRWR FIB address,,alternate return address,[parameter-name-1]

The parameters and respective defaults for the Rewrite macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open for I/O at the time of execution of this statement.

2. For files opened for sequential access, the last input/output function executed for the associated file, prior to the execution of the DLRWR function, must have been a successfully executed DLGXT. For ISP files, an error condition is returned if the key values in the record and the process area do not match.

3. The size of the record in the process area must be equal to the size of the record being replaced.

4. The current record pointer is not affected by the execution of a DLRWR macro. After a DLRWR, the current record pointer can be set by executing a DLPOS macro with GT parameter value.

5. For a file opened for random or dynamic access, UFAS logically replaces the record specified by the contents of the key (KEYPTR). If the file does not contain the record, an error condition is returned.

6. When the rewrite is unsuccessful, the file is unaffected.

7. When an invalid key condition or unsuccessful write function is detected, control is returned to the alternate return address.

8. The key to be used for comparison must be in its appropriate position in the process area and must have same data type and code set as the key values in the file.

Example

Rewrite a record in the file specified in the FIB. On a logical error, return to ALT.

```
1      8       16
------------------------
       DLRWR   FIB,,ALT
       .
       .
       .
FIB    FIBMAC  D.,ISP,[PROAR,RECORD,0],I.
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS     14
```

Delete Macro - DLDEL

Function

The Delete macro logically deletes a record from a file.

Format

```
DLDEL  FIB address,,alternate return address,[parameter-name-1]
```

The Delete macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated file must be open in the I/O mode at the time of the execution of this statement.

2. For a file opened in the sequential access mode, the statement executed for this file prior to the execution of the DLDEL macro must have been a successful DLGXT. For ISP files, an error condition is returned if the key values in the record and the process area do not match.

3. For file opened for random or dynamic access, the record identified by the key is logically deleted from the file. If the record does not exist, an invalid key condition is returned and the alternate return is taken.

4. The current record pointer is not affected by the execution of the DLDEL function. After a DLDEL, the current record pointer can be set by executing a DLPOS macro with GT parameter value.

5. The key to be used for comparison must be in its appropriate position in the process area and must have same data type and code set as the key values in the file.

Example

Delete the record specified in FIB2 and on a logical error execute the procedure at ALT2.

```
1      8       16
       DLDEL   FIB2,,ALT2
       .
       .
       .
FIB2   FIBMAC  D.,ISP,[PROAR,RECORD,0],I.
       .
       .
       .
ALT2   NULL
       .
       .
       .
RECORD BSS     14
```

Position Macro - DLPOS

Function

The Position macro provides for logically positioning the file for subsequent access.

Format

```
DLPOS  FIB address,VIB address,alternate return address,
       [VIB-parameter-name-1,VIB-parameter-value-1]
```

NOTE: VIB must be defined as BSS 2.

The Position macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Comparison type indicator				Optional Position file to the record containing the key.
	FCT	EQ	EQ	Means equal to specified key.
		GT		Means greater than specified key.
		GE		Means greater than or equal to specified key.
	FCTI	Address		Location of word having FCT value in bits 0-17 in uppercase ASCII characters.

Rules

1. The type of comparison specified (equal to, greater than, or equal to or greater than) occurs between the key associated with a record in the file and a data item defined by the key field (KEYPTR in FIB).

2. If VIB parameter is not specified, the relational operator EQ (equal to) is implied by default.

3. If the comparison is not satisfied, execution is unsuccessful, the position of the current record pointer is undefined. An invalid key condition is returned and the alternate return is taken.

4. The key to be used for comparison must be in the appropriate position in the process area and must have the same data type and code set as the key values in the file.

Example

Set the current record pointer to the record with key value greater than or equal to the key value specified in the FIB. If such a record does not exist, execute the code at ALT.

```
1      8      16
________________________________

       DLPOS  FIB,VIB,ALT, [FCT,GE]
       .
       .
       .
VIB    BSS    2
FIB    FIBMAC D.,ISP,[PROAR,RECORD,0,ACCMOD,DYN],I.
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS    14
```

Get Next Macro - DLGXT

Function

The Get Next macro makes the next logical record available in ascending key order.

Format

DLGXT FIB address,,end-of-file address,[parameter-name-1]

The parameters and respective defaults for the Get Next macro are as follows:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. If the next logically sequential record was deleted, it is bypassed.
2. An end-of-file condition is returned when no next logical record exists in the file. This condition is considered an unsuccessful read request. Control is returned to the end-of-file address.
3. If the position of the current record pointer is undefined, an error condition results. The current record pointer is not defined following an unsuccessful read request, after a checkpoint under ACCESS/MONITOR/, or rollback.
4. If present, the size of the record made available in the record size word is returned. The size made available either is the actual size or the maximum size, whichever is the smallest.
5. The record is moved from the buffer to the process area.

Example

Get the next record. If end of file, go to ALT.

```
1      8      16
---------------------
       DLGXT  FIB,,ALT
       .
       .
       .
FIB    FIBMAC D.,ISP,[PROAR,RECORD,0],I.
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS    14
```

Get Macro - DLGET

Function

The Get macro makes available a specified record based on a key value.

Format

```
DLGET  FIB address,,alternate return address,[parameter-name-1]
```

The Get macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Indirect FIB addressing	FIBI	None		Optional If this key word is specified, it implies that argument 1 is not the FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. Following the unsuccessful execution of any DLGET macro, the position of the current record pointer is undefined. The contents of the associated record area remain undisturbed.

2. The execution of a DLGET causes the value of the key to be compared with the value contained in the corresponding data item of the stored records in the file, until the first record having an equal value is found. The current record pointer is positioned to this record, which is then made available. If no record exists, execution of the function is unsuccessful and the alternate return is taken.

3. The key to be used for comparison must be in its appropriate position in the process area and must have the same data type and code set as the key values in the file.

4. The size of the record made available is returned in the record size word, if present. The size made available either is the actual size or the maximum size, whichever is the smallest.

5. The record is moved from the buffer to the process area.

Example

Get the record with key value equal to that in the process area. If it is not an active record, return to ALT.

```
1      8      16
       DLGET  FIB,,ALT
       .
       .
       .
FIB    FIBMAC D.,ISP,[PROAR,RECORD,0,ACCMOD,RAN],I.
       .
       .
       .
ALT    NULL
       .
       .
       .
RECORD BSS    14
```

Return Macro - UFSRET

Function

The Return macro provides a means of returning control to UFAS after a user error processing procedure is completed.

Format

UFSRET ERROR

NOTE: User error procedures are specified in the FIB macro via the user error procedure and error table parameters.

Rule

When a user error procedure contains a UFAS macro, which can result also in a user error procedure for another file being entered by UFAS, the second procedure must conclude also with a UFSRET return for that particular function.

Example

Return to UFAS after a user error exit.

```
1      8      16
       UFSRET ERROR
```

Checkpoint Files Macro - DCKPF

Function

The Checkpoint Files macro establishes a checkpoint for all open files.

Format

DCKPF

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.
2. All opened files are included in the checkpoint.
3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur, since the program is not checkpointed.
4. The macro performs a buffer flush before establishing the checkpoint.
5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.
6. Checkpoint is taken only if protected files are present and have been opened.

Checkpoint Files and Program Macro - DCKPT

Function

The Checkpoint Files and Program macro establishes a checkpoint for all open files and for the program.

Format

DCKPT

Rules

1. All randomly allocated files written by the program must have FMS ABORT/ROLLBACK/ protection.
2. All open files are included in the checkpoint.

3. The macro must be executed at a point from which processing of the files can be resumed if a rollback does occur.

4. The macro performs a buffer flush before establishing the checkpoint.

5. For any files being processed with ACCESS/MONITOR/ (see File Management Supervisor manual), buffers are set empty and the record pointer is set to undefined.

6. A QX file large enough to accommodate the program must be allocated, but not declared in the user program (see Program Recovery/Restart manual).

7. Checkpoint is taken only if protected files are present and have been opened.

8. The Q register (bits 18-35) contains the status code returned by MME GECHEK as described in the General Comprehensive Operating Supervisor manual. Zero status indicates a successful checkpoint.

Information Macro - DINFO

Function

The DINFO macro either stores File Access Control Table (FACT) information or the size of each section of the FACT table into a user specified table area.

Format

```
DINFO     FIB address, VIB address, alternate return address,

          [parameter-name-1, parameter-value-1....,

          parameter-name-n, parameter-value-n]

NOTE:     VIB must be defined as BSS 1.
```

The parameters and respective defaults for the Information Macro are as follows:

Parameter	Name	Value	Default	Comments
Function	FCT	1		Store FACT section 1 into user table area. Includes words FACT-8 through FACT +98 for all file types.
		2		Store FACT section 2 into user table area. Includes words 0 through +59 for UFF Indexed with no alternate keys: 0 through +81 for UFF Indexed with alternate keys; 0 through +17 for ANSI/IBM/GFRC and UFF tape; 0 through +41 for ISP; and 0 through +9 for H2000.

Parameter	Name	Value	Default	Comments
		3		Store FACT section 3 into user table area. Include words 0 through 26 (dec.)
		4	4	Store the entire FACT into the user table area.
		5		Store FACT-8 through FACT +5 of section 1 into user table area. This option provides FACT information that is equivalent to the GFRC FILCB.
		6		Store the size (in binary) of FACT sections 1, 2, and 3 into a 3-word user defined table area. Zero will be given as the size of section 3 if it is non-existent.
Table Pointer	TBL	Address		Required. The location of a table large enough to contain the requested information.
Indirect FIB Addressing	FIBI	None		Optional. If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address in bits 0-17.

Rules:

1. The designated file must be opened in either the input or I/O.
2. The TBL parameter is required and must point to an area large enough to contain the requested information.
3. The user must provide an alternate return address.
4. If an error is detected during execution of the DINFO macro, UFAS will return control to the user via the alternate return except when executing the DINFO macro on any unopened file in which case UFAS will branch to a users error procedure if it is present or abort.
5. A function code of 6 requires a table size of 3 words. After a successful execution of a DINFO macro the table will contain the following values:

Word	Contents
1	Size of FACT section one, right justified. (In Binary)
2	Size of FACT section two, right justified. (In Binary)
3	Size of FACT section three, right justified. (in Binary) A size of zero will be given if section 3 is non-existent.

6. If no function code (FCT) parameter is specified the entire FACT will be given to the user.

7. The FACT sizes given in this specification are applicable to SR ADF2.

8. This macro may not be executed for files processed under TP or I-D-S/II.

9. The following error conditions will be identified:

   a. 8-01 File not opened. (ALT return not taken)
   b. F0 A potential F0 memory fault during DINFO execution may take place because of insufficient table area size.
   c. F1 Invalid function code
   c. F2 The user asked for FACT section 3 and it does not exist.

10. Listed below, in decimal, are the size of the entire FACT for the various file formats.

UFF SEQUENTIAL Tape and Mass Store	192
UFF RELATIVE	192
UFF INDEXED	192
IBM	152
ANSI	152
H2000	136
UNLT	152
GFRC Tape and Mass Store	152
ISP	149

Example

Store the entire FACT contents of the file specified by FIBI into a table specified by TABLE.

```
1         8         16
_________________________________________________
          DINFO     FIB1,VIBI,ALTRET,[FCT,4,TBL,TABLE]
          .
          .
          .
FIB1      FIBMAC    DA,UFF,[PROAR,RECORD,0,ORG,IND,
          ETC       KEYPTR,PRIME],KY
          .
          .
          .
RECORD    BSS       21
PRIME     VFD       1/0,17/0,18/4
          ZERO
VIBI      BSS       1
TABLE     NULL
          .
          .
          .
ALTRET    NULL
          .
          .
          .
```

Rollback Files Macro - DROLF

Function

The Rollback Files macro restores the files open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the previous checkpoint.

Format

DROLF

Rules

1. A DCKPF or DCKPT macro must be executed before performing the rollback.

2. After rollback, the current record pointer is undefined.

Rollback Files and Program Macro - DROLP

Function

The Rollback Files and Program macro restores the program and all files that are open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the checkpoint and to restart the program from that point.

Format

DROLP

Rules

1. A DCKPT macro must be executed before a rollback is executed.

2. After rollback, the current record pointer is undefined.

Wrap-up Macro - DWRAP

Function

The Wrap-up macro closes files currently open.

Format

DWRAP

Rule

All opened files are closed. Device disposition codes are not honored.

Close Macro - DCLOS

Function

The Close macro terminates the processing of one or more files. DCLOS consists of the following four functions:

1. Single-Close/User-Defined FIB List

2. Single-Close/Macro-Defined FIB List

3. Multiple-Close/User-Defined FIB List

4. Multiple-Close/Macro-Defined FIB List

SINGLE-CLOSE/USER-DEFINED FIB LIST

Format

```
DCLOS { FIBLS,address of FIB list               } ,[parameter-name-1,
      { FIBLSI,address of pointer to FIB list   }

        parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Rewind indicator				Optional
	REW	RWFL	RWFL	Rewind file on close.
		LKFL		Lock file (see note).
	REWI	Address		Location of rewind indicator (4 ASCII uppercase characters).
	REWV			Rewind value.
		0	0	Rewind file.
		3		Lock file (see note).
	REWVI	Address		Location of rewind binary representation value in bits 18-35.

NOTE: Whenever the lock parameter is used, the file is disposed of as indicated by the disposition code specified on the device control card (see Control Cards Reference Manual). The lock parameter and a save or continue disposition code are mutually exclusive. If a file that has an S or a C disposition code is closed with lock, the lock request in ignored, the file is not rewound or released, and no indication is given to the user that the lock request was ignored.

Rule

The Close macro can only be executed for a file that was opened with DOPEN.

Example

Close and lock the file specified in the FIB at ADF.

```
1         8         16
          DCLOS     FIBLS,LIST3,[FIBA,ADF,REWV,3]
          .
          .
          .
LIST3     BSS       2
ADF       FIBMAC    D.,ISP,[PROAR,RECORD,0],I.
          .
          .
          .
RECORD    BSS       14
```

SINGLE-CLOSE/MACRO-DEFINED FIB LIST

Format

```
DCLOS    ,,[parameter-name-1,parameter-value-1,...parameter-name-n,
           parameter-value-n]
```

The following parameters and respective defaults apply:

Parameters	Name	Value	Default	Comments

Same as Single-Close/User-Defined Function

Rules

1. FIB list address is not to be supplied.
2. Left bracket must be preceded by two commas.
3. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/USER-SUPPLIED FIB LIST

This function terminates the processing of one to n files. Only one FIB list is built, which is utilized by all files to be closed.

Format

```
DCLOS  FIBLS, FIB list address ,[*, FIB address,
ETC    RewV, *, FIB address, RewV,*,.....]
```

The following parameters and respective defaults apply:

Parameter	Value	Default	Comments
FIB address	address		Required; location of FIB, for file to be closed
Rewind Indicators			
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and lock file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Space for the FIB list must be defined as 1+n words, where n is the number of files to be closed.

2. Each list of parameters for every file <u>must</u> be preceded by an *.

3. Key words must not appear within the brackets.

4. Parameters <u>must</u> be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.

5. All Single-Close/User-Defined Function Rules apply.

MULTIPLE-CLOSE/MACRO-DEFINED FIB LIST

This function terminates the processing of from one to n files. It builds one FIB list in the user program, immediately following the DCLOS command. Storage is automatically allocated, and the list is used by all files that are closed via this function.

Format

```
DCLOS  ,,[*,FIB Address, REW-V,*,FIB Address2,REW-V,...]
```

The following parameters and respective defaults apply:

Parameters	Value	Default	Comments
FIB address	address		Required; location of FIB for file to be closed
REWIND INDICATOR			Rewind value
	0	0	Rewind file
	1		Do not rewind file
	2		Rewind and unload file
	3		Lock file without rewind
	4		Rewind and unload file
	6		Lock file with rewind and unload
	8		Rewind unit
	9		Do not rewind unit
	10		Rewind and lock unit
	12		Rewind and unload unit

Rules:

1. Two commas _must_ precede the left bracket.
2. Each list of parameters for every file _must_ be preceded by an *.
3. Key words must not appear within the brackets.
4. Parameters _must_ be listed in predefined order. Defaults may be taken; however, if they are, the subsequent comma must be used.
5. The FIB list address should not be supplied.
6. All Single-Close/User-Defined Function Rules apply.

SECTION VII

DM-IV DATA BASE FILE ORGANIZATIONS

INTEGRATED FORMAT

An integrated file can contain variable size records on fixed-length control intervals. This file must be allocated to a mass storage device and can be accessed sequentially, randomly, and dynamically by a data base control system. Access is effected by a data base key to obtain the control interval and the record within the control interval. After the file is allocated and pre-formatted with control interval information, it remains static in size. Concurrent access is provided to support multiple user programs simultaneously. Detailed record formats and inter-relationships reside in an external control table (schema), which is paired with the integrated file in execution.

I-D-S/II performs all processing of the content of logical records and creates and maintains set relationships, contents of pointer fields, matching of key values, and owner/member status in integrated files.

UFAS provides support to I-D-S/II by performing such functions as buffer management and making calls to the Input/Output Supervisor for reading and writing control intervals of integrated files.

INDEXED FORMAT

This indexed organization is similar to the indexed file format described in Section V. Several significant differences make access to this format impossible from anywhere except DM-IV data base routines. The differences are:

- DBK is used in lieu of a page and line. DBK is interpreted by I-D-S/II routines.
- Key values may consist of other than alphanumeric characters.
- Alternate key index structures must be created with DM-IV utility Q2KEYF. See the DM-IV Utility Routines manual for a description of this utility.
- Record TYPE information defined during schema generation is stored with each record.

DUAL DATA ENVIRONMENT

Dual data environment (DDE) file formats are similar to the DM-IV indexed formats in Native mode. The data content is BCD, while the control information is byte oriented.

DM-IV indexed users should note the following restrictions:

- Alternate key access is not supported
- The record TYPE of a record added by a COBOL-68 program will be zero. An I-D-S/II user routine must be included to define the TYPE of these records.

Directory Selection

The UFAS directory selected by the validator can be overridden with the following JCL cards:

```
1      8      16
$      USE    package-name-1
$      NLOAD  package-name-2
$      OPTION LDLIB
$      EQUATE package-name-1/package-name-2/
```

The $ USE card specifies the directory package to be loaded. The $ NLOAD card inhibits the loading of the default directory package. The $ OPTION card causes the software libraries to be searched and the directory packages loaded immediately, before processing the $ EQUATE card. The $ EQUATE instructs the loader to assign the address of package-name-1 to all instructions that reference package-name-2.

The validator defaults to the update/retrieval directory. The user must override the default directory to load the data base or to reduce the amount of memory for a retrieval-only system.

Tables 7-1 and 7-2 specify the directories best suited for a minimum memory configuration.

Table 7-1. UFAS Directories For DM-IV-TP

PROCESSING MODE DATA BASE ORGANIZATION	RETRIEVAL ONLY	UPDATE/ RETRIEVAL
INTEGRATED		
w/o alt keys	.DIDS2	.DIDS2
with alt keys	.DIDTR	.DIDTU
INDEXED/ INTEGRATED		
w/o alt keys	.DIDRV	.DIDDY
with alt keys	.DIDRA	.DIDDA

Table 7-2. UFAS Directories For DM-IV (Batch Or Timesharing)

PROCESSING MODE DATA BASE ORGANIZATION	LOAD	RETRIEVAL ONLY	UPDATE/ RETRIEVAL
INTEGRATED			
w/o alt keys	.DIDS2	.DIDS2	.DIDS2
with alt keys	.DIBLA	.DID2R[c] .DIDYA	.DID2U[c] .DIDYA
INDEXED/ INTEGRATED			
w/o alt keys	.DIBLD[a]	.DIRTV	.DIDYN
with alt keys	.DIBLA[b]	.DIRVA	.DIDYA

[a] record location key only

[b] both record location key and additional access keys

[c] uses less memory than DIDYA, the validator default

SECTION VIII

FILE GENERATION AND CONCATENATION

The File Generation facility allows the user to maintain file generation groups and provides the ability to concatenate UFAS sequential files on tape and mass storage.

File Generation Group maintenance consists of the ability to define, chronologically or functionally, the following:

- Related catalog data files
- UFAS insertion of file generation numbers in cataloged output tape files
- Verification of file generation numbers in cataloged input tape files

Support of file generations and concatenation is provided for the following file formats and media types.

File Format	Media Type
GFRC	TAPE TAPE7 TAPE9 PRMFL
ANSI	TAPE9
IBM	TAPE9
UFF	TAPE9

RESTRICTIONS

Unlabeled tape file, mixed TAPEs and PRMFL concatenation sequences are not supported.

All File Generation Group member files and file concatenation element member files must: (1) be sequential files; (2) have identical block size; (3) be in the same recording mode; (4) be the same media type; and (5) have the same record format. GFRC sequential mass store files to be concatenated must be of the same size and consist of single extents. In addition, if code set transliteration is specified, the following requirements apply:

- File Generation entries (output) can specify any allowable internal code set, but the external code set must be the code set of the defined File Generation Group of which this entry is to be a member.

- Input transliteration can be from the external code set of the defined File Generation Group to any allowable internal code set.
- Input code set transliteration, if specified for file concatenation, must be the same for all files in the defined concatenation set.

Multi-file tape volumes and input tape label name verification are not allowed.

User label exits are being serviced only to the extent that this feature is supported by UFAS (i.e., limited to GFRC labeled tape files).

Concatenated files can be opened (DOPEN) for input only. The user must code a single OPEN (DOPEN) and a single CLOSE (DCLOS) for each file concatenation set. Intervening CLOSE and OPEN services are provided by UFAS. The concatenated file must be opened with rewind to insure proper file position at the start of each activity.

The position macro (DLPOS) can be used only if the number of records to be skipped in the forward or backward direction defines an extent of the current file in the concatenation set equal to, or less than, the number of records processed (backspace) or the number of records known to remain in the file (forward-space).

FILE GENERATIONS

The characteristics of each file in the File Generation Group are maintained in a control file called the File Group Index. The user can select any specific file or any number of group member files for concatenation in any prescribed input order (i.e., current and prior generations).

New files can be entered into a File Generation Group as additions to the defined File Group Index or as replacements in the defined file notation definition (i.e., future generations).

The File Group Index is created and maintained by the utility program, FGCRE. For a detailed explanation of the use of this program, refer to File Generation Utility reference manual, order number DG79.

Files are allocated within the File Generation Group via the $ FILGP control card.

FILE CONCATENATION

Files that are members of a defined File Group Index are concatenated as a result of use of the $ FILGP control card.

Example:

```
        $ FILGP  IN,X1D,R,CAT/FILE,(-6,-4,-3,-1)
```

In which case, cataloged files from the File Group Index identified by CAT/FILE generations, minus 6,4,3, and 1, become input to this activity as a single continuous file.

Files can be concatenated by building a group of file control cards. The file code field of the first file control card is the file code expected by the application program.

Subsequent files are concatenated with the first file by a double quote character (") in the file code field of the control card.

This method permits the concatenation of logical sequences of uncataloged files.

Example:

```
$    TAPE9   IN,X1D,,12345,,,,DEN16
$    TAPE9   ",,,23456,,,,DEN16
$    TAPE9   ",,,40171,,,,DEN16
```

In this example, tape files resident on reel serial numbers 12345, 23456, and 40171 will become input to this activity as a single continuous file.

SECTION IX

TIME SHARING ENVIRONMENT

A user program can be executed under the Unified File Access System in either batch or time sharing mode. However, not all file formats supported in the batch mode are available in the time sharing mode. The file formats supported in the time sharing mode are the GFRC linked mass storage and the UFF of sequential, relative, indexed, and integrated files. Tape file formats, Indexed Sequential Processor file formats, and the building of UFF indexed alternate keys are not supported in the time sharing mode and no label processing is provided for the file formats supported.

The directory names (symbol references) used to select the UFAS routines to be linked are the same for both the batch mode and time sharing mode.

FILE SPECIFICATION

If the file code (fc - two ASCII characters) specified in the file information block macro corresponds to a file already contained in the Available File Table (AFT), the file contained in the AFT will be operated upon. If the file selected is an indexed file, and the file code of the data file is in the AFT, then the file code assigned to the index file in the file information block macro must be also in the AFT.

If the file code is not found in the time sharing Available File Table, by UFAS, the user must supply the operand ASCII descriptor (ADSC9) for the catalog or file string as the file name parameter, FLNAME, in the file information block macro. The file name descriptor, FLNAME, appears in word 9 of the FIB.

The file description must have one of the following formats:

- Filename
- Filename$Password
- USERID/Filename
- /Catalogname/Filename
- USERID/Catalogname$Password/.../Filename$Password,Permissions

- Filename "Alternatename",Permissions

- Filename,Permissions

- *Filename

- *

The input strings for these formats may be in either lowercase ASCII or uppercase ASCII characters. The scan of the file descriptor string is terminated by any non-valid character (see <u>File Management Supervisor</u> manual for valid file name characters), excluding comma, slant, dollar sign, quotation marks, or blanks. The two file strings for an indexed file must be given as: data file string; index file string. The ASCII descriptor in FLNAME must have a total character count, that is, it must include the count of both file strings plus one for the character ";".

FILE ACCESSING

File name accessing when the file is not specified in the AFT or is in the AFT, but was not entered by UFAS, is governed by the following rules:

1. If the catalog or file description has a valid alternate name, this name becomes the file name in the AFT.

2. If the catalog or file description has a file name with less than eight characters but no alternate name, the file name is entered in the AFT.

3. Descriptions containing a name with eight characters or less, preceded by an asterisk, are entered in the AFT only if the description occurs as a separate file name without user identification, permissions, or alternate name. The asterisk as a name by itself is converted to *src (current file) and entered in the AFT.

4. A file name preceded by a slant (/) cannot be a temporary file.

5. If no permissions are specified in the file description, the default is:

   a. Read - if the user identification in the catalog or file string differs from the current user identification.

   b. Read and Write - if the user identification is the same as that of the current user.

6. If a file being opened has a file name or alternate name of a file already entered in the AFT by UFAS, the following occurs:

   A unique alternate name is generated as the AFT name with the format *tnnnnnn, where n represents an ASCII numeric string ranging from 1 to 999999. Then, the file is accessed again. If the status is "file busy" an error has occurred; otherwise, the generated alternate name becomes the valid name assigned to the file.

7. If a file name or alternate name is the same as that of another file in the AFT but not entered by UFAS, the following occurs:

   a. If the file referenced in the FIB has no intermediate catalogs and the file specified in the AFT is a temporary file, the file already in the AFT is used.

b. If the requested file has intermediate catalogs and the file specified in the AFT is a temporary file, an error message is issued at the terminal.

c. If the file requested has intermediate catalogs and the file specified in the AFT is a permanent file, the file originally in the AFT is removed.

8. If a file description has a name with more than eight characters and an alternate name has not been specified, an alternate name with less than nine characters is generated also having the format *tnnnnnn.

9. UFAS creates:

   a. A temporary file if a file with no intermediate catalogs was specified or the file name includes an asterisk.

   b. A permanent file if the file name is a valid permanent file name.

10. If errors are detected in enforcing the above rules, error messages are given to the users terminal. If a users error routine has been specified, via the USERER parameter, it will be invoked with a major status of 8 and a substatus of 91.

TERMINAL INTERFACE

UFAS does not have the capability to write to a file assigned to SYSOUT in the time sharing mode. Errors that result in aborts cause error messages to be issued at the terminal.

Input and output files will be assigned to the terminal under the following conditions at the time the file is opened:

- If the file name is SYSIN or SYSPRINT and is not specified in the AFT.
- If the file name descriptor, FLNAME, in the FIB macro is null.
- If the file name is ** .

When the Open macro assigns a file to the terminal, a major status 9 and substatus 6 are specified in the file status word, FSCODE, of the FIB. For an input file assigned to the terminal, the Get Next macro causes a "?" and the file code characters to be issued to the terminal as "?FC". In response, the terminal user enters a string of ASCII characters followed by a carriage return. This string of characters is moved to the process area designated as PROAR in the FIB macro. An end-of-file status is returned to the user program upon receipt of a single carriage return character or the control-shift-L character (octal 034).

When a Put macro is issued by the user program to an output file assigned to the terminal, an ASCII string of characters is expected in the process area. This ASCII string will be output to the terminal. The size of the string moved from the process area is one of the following:

- Record size from word 0 of the Variable Information Block (VIB).
- Record size word (RSZWRD) from FIB macro.
- Maximum record size (RSZ) from FIB macro.

After the output string is sent to the terminal, the automatic slew of a line occurs. Any additional line control required must be included in the character counts.

The following example in Figure 9-1 illustrates the creation of an H* file from a batch activity for execution in the time sharing environment. Note that, with the exception of the changes indicated in the figure, the time sharing job is the same as the batch job.

```
1      8       16
$      IDENT   Account#, Name
$      LOWLOAD 36
$      OPTION  SAVE/TSSHSTAR,NOGO,NOSETU
$      GMAP    NDECK,EISF
$      LIMITS  ,30K
       LODM    .DMAC
       SYMREF  .DSEQ,.DVLRI,.DVLPO
       SYMDEF  TSSUF
       LODM    .G3TSM
       .SSDRL
TSSUF  NULL
       DRL     DEFIL         TEMP FILE FOR ABORT DUMP  (OPTIONL)
       ZERO    ARGS,STAT
*
       DOPEN   FIBLS,FIBLST,(FIBA,FIBIN,SYM,.DVLRI,MODE,IN)
       DOPEN   FIBLS,FIBLST,(FIBA,FIBOT,SYM,.DVLPO)
*
       MLR     ,,040
       ADSC9   MSG1,0,52
       ADSC9   OTPROC+1,0,52
       DLPUT   FIBOT,VIB
*
       MLR     ,,040
       ADSC9   MSG2,0,36
       ADSC9   OTPROC+1,0,52
       DLPUT   FIBOT,VIB
*
       MLR     ,,040
       ADSC9   MSG3,0,44
       ADSC9   OTPROC+1,0,52
       DLPUT   FIBOT,VIB
*
LOOP   NULL
       MLR     ,,040
       ADSC9   0,0,0         CLEAR INPUT PROCESS AREA
       ADSC9   INPROC,0,56
*
       DLGXT   FIBIN,VIB,RETRN
*
       CMPC
       ADSC9   INPROC,0,6
       ADSC9   ENDFLG,0,6
       TZE     DONE
*
       MLR     ,,040
       ADSC9   INPROC,0,52   MOVE INPUT DATA TO
       ADSC9   OTPROC+1,0,52  OUTPUT AREA
*
```

Figure 9-1. Batch Activity To Create H* File For Execution In Time Sharing Environment

```
1      8       16
       DLPUT   FIBOT,VIB
       TRA     LOOP
*
*
DONE   NULL
       MLR     ,,040
       ADSC9   ENDMSG,0,36
       ADSC9   OTPROC+1,0,52
       DLPUT   FIBOT,VIB
*
       DCLOS   FIBLS,FIBLST,(FIBA,FIBIN)
       DCLOS   FIGLS,FIBLST,(FIBA,FIBOT)
*
       DRL     RETURN            (GEFINI...)
*
RETRN  NULL
       DRL     ABORT             (GEBORT...)
*
*
*
FIBLST BSS     3
VIB    BSS     2
STAT   BSS     2
ARGS   UASCI   2,ABRT
       OCT     5
MSG1   OCT     015007007007      CR,BELL,BELL,BELL
       UASCI   9,TSS/UFAS - ENTER DATA FOLLOWING 'IN?
       UASCI   3,' PROMPT.
MSG2   UASCI   9,PROGRAM WILL VERIFY RECEIPT OF DATA.
MSG3   UASCI   9,TO TERMINATE, ENTER '***END' OR HIT
       UASCI   2,RETURN.
ENDMSG UASCI   9,*** END OF TSS/UFAS EXERCISE ***
INPROC UASCI   9,
       UASCI   5,
OTPROC OCT     015007007007      CR,BELL,BELL,BELL
       UASCI   9,
       UASCI   4,
ENDFLG UASCI   2,***END
FIBIN  FIBMAC  IN,GFRC,[NBUF,2,RSZ,56,CISZ,1280,PROAR,INPROC,0,
       ETC     MEDCOD,6,FLNAME,INFC,0]
FIBOT  FIBMAC  OT,GFRC,[NBUF,2,RSZ,56,CISZ,1280,PROAR,OTPROC,0,
       ETC     MEDCOD,6,FLNAME,OTFC,0]
INFC   ADSC9   SYSIN,0,5
OTFC   ADSC9   SYSPRT,0,8
SYSIN  UASCI   2,SYSIN
SYSPRT UASCI   2,SYSPRINT
       END
       EXECUTE DUMP
$      PRMFL   H*,W,R,UFAS/DAP/DAPHSTAR
$      ENDJOB
```

Figure 9-1 (cont). Batch Activity To Create H* File For Execution In Time Sharing Environment

Commands commonly used to load an H* file for execution in a time sharing environment include:

- LODT command
- LODX command
- FORTRAN run command FRN

SECTION X

EXCEPTION AND ERROR CONDITIONS

The exception and error conditions defined in this section comprise the exception condition major status and substatus codes with corresponding messages where applicable, label processing messages, and console messages.

EXCEPTION CONDITIONS AND STATUS CODES

For every exception condition, as well as successful completion, there is a corresponding major status and substatus code in a two-byte ASCII field. The major status code is either 0, 1, 2, 3, 5, 8, 9 or F ASCII and identifies the type of error. The substatus code, which is also expressed in ASCII (except the substatus for major status 8 which is in binary), identifies the specific error. If the user indicates in the FIB that a file status code field pointer is provided, then that field is initialized each time with the appropriate value (including 0x for successful).

UFAS always returns control to the user if appropriate exits are provided and only aborts if instructed to do so, or if no exits are provided. When a function is successfully completed, UFAS makes a normal return. If an error with major status 1, 2, 5, or F is detected, UFAS takes an alternate exit. For errors with major status 1 or 2 (when the alternate exit is not specified), and 3 or 9, transfer is made to the file name error exit, if one is provided, or to the processing mode error exit. If no file name or processing mode error exit is specified a message is issued and the activity is aborted. When an error with major status 8 is detected, an appropriate error message is issued before transfer is attempted to the error exit.

When control is transferred to a user error procedure as a result of an error with major status 1, 2, 3, or 9, UFAS inhibits file processing until the user responds by indicating the action to be taken. If transfer is caused by an error with major status 8, the file is permanently inhibited. In both instances, upon termination of the user error procedure, the user is required to return control to UFAS via macro UFSRET, indicating what action is to be taken; that is, if recovery should be attempted, the function that was interrupted should be terminated and control returned, or the activity aborted.

The action to be taken is indicated by one of the following responses and is effected by changing only the substatus code field. The major status code field may remain unchanged.

Major Status Field	Substatus Field Response	Action
1 or 2	R	Return to user taking normal exit.
	S or P	Abort
	(field unchanged)	Return to user taking normal exit.
3 or 9	R	Return to user taking normal exit.
	P	Process record or block as normal.
	S	Skip the block.
	(field unchanged)	Abort.
8	R	Return to user taking normal exit.
	P	Abort
	S	Abort
	(field unchanged)	Abort

All files may continue to be referenced except those that are inhibited. Calls to inhibited files, except to close the files, cause the error procedure to be reiterated and an additional message to be issued indicating the status of the file.

Termination of an activity may occur any time the user program has control. If the Wrapup function is needed to close open files, this may be accomplished with the DWRAP macro or the system's normal termination vector (.CWRAP).

Major Status Code	Substatus Code			
0	0	Meaning:	Normal termination.	Successful Completion
0	2		Duplicate alternate keys where duplicate keys are allowed for this file (not applicable to ISP files). On read, two successive records contain the same alternate key values for a given key of reference. On write or rewrite, the record just written creates a duplicate key value for at least one alternate key.	
1	0	Meaning:	An end-of-file condition was detected.	Logical EOF
2	1	Meaning:	Ascending sequence requirements were violated on a write or the key value was changed between a successful read and next rewrite.	Invalid Key Condition
2	2		An attempt was made to write or rewrite a record that would generate a duplicate key. This condition occurs for prime keys and alternate keys only when duplicate keys are not allowed.	
2	3		No record found. This condition is also invoked for all functions, except write when a boundary violation occurs.	
2	4		An attempt was made to write beyond the limits of a file space.	
3	0	Meaning:	Reserved.	
3	1		An unrecognizable hardware error; i.e., one that is not defined by code 9x.	Input/Output Processing
3	4		MME GEMORE denial on linked mass storage file.	
5	0	Meaning:	No User Label was found. This condition is invoked by the Read User Label Record function. A UFSRET (USERLB) must be issued to return to UFAS.	ANS, UFF, or IBM User Label processing

Major Status Code	Substatus Code		
8	00	Message:	IC&I STACK OVERFLOW - TOO MANY NESTED CALLS ON FILE xx
		Meaning:	Too many outstanding label or error exits caused internal stack overflow.
8	01	Message:	I/O ATTEMPTED ON UNOPENED FILE xx
		Meaning:	Input/output was attempted on an unopened file.
8	02	Message:	I/O ATTEMPTED WHILE A RETURN FROM AN ERROR PROCEDURE OR LABEL EXIT IS OUTSTANDING ON FILE xx
		Meaning:	Control was transferred at the user label or error exit level; access of the file is no longer possible until control is returned to UFAS via the UFSRET macro.
8	03	Message:	ANSI BLOCK SEQUENCE INDICATOR ERROR ON FILE xx
		Meaning:	Block sequence indicator in block control word does not match internal block serial counter.
8	04	Message:	REQUIRED VIB POINTER IS MISSING ON CALL TO FILE xx
		Meaning:	Variable information block was not specified for a required function.
8	05	Message:	INVALID SPANNED/PARTITIONED RECORD SEGMENT ENCOUNTERED ON FILE xx
		Meaning:	Record segments are out of order in a spanned/partitioned file.
8	06	Message:	INPUT RECORD SIZE EXCEEDS FIB RECORD SIZE ON FILE xx
		Meaning:	Actual record size is larger than the maximum size specified in the FIB.
8	07	Message:	RETURN FROM USER LABEL EXIT OR ERROR PROCEDURE IS NOT IN CORRECT POP-UP SEQUENCE FOR FILE xx
		Meaning:	Return made by a user procedure from nested label exits was not in reverse sequence to the order the procedures were called; that is, the last procedure to gain control should be the first to return control.
8	08	Message:	UNRECOGNIZABLE LABEL DETECTED ON FILE xx
		Meaning:	Invalid label was used on file.

Major Status Code	Substatus Code		
8	09	Message:	RETENTION PERIOD DESCRIPTOR IN FIB IS INVALID FOR FILE xx
		Meaning:	Incorrect retention period was specified.
8	10	Message:	BLOCK SERIAL ERROR READING FILE xx
		Meaning:	Block serial number in block control word does not match internal block serial counter.
8	11	Message:	INVALID TAPE MARK ON FILE xx
		Meaning:	Non-standard end-of-file mark was encountered in data file.
8	12	Message:	ERROR WHILE SKIPPING CHECKPOINT RECORDS ON FILE xx
		Meaning:	Partial checkpoint delimiter (zero tape mark) was found in data file.
8	13	Message:	BLOCK LENGTH ERROR READING FILE xx
		Meaning:	Logical block size as calculated does not match the size specified in the block control word.
8	14	Message:	ON RETURN FROM USER ERROR PROCEDURE THE MINOR STATUS CODE RESPONSE IS INVALID FOR FILE xx
		Meaning:	User response was invalid for the error type detected.
8	15	Message:	UNABLE TO RECOGNIZE IO STATUS ON FILE xx
		Meaning:	Unexpected major status or substatus was encountered in IOS status words. See the *File Management Supervisor* manual for status explanation.
8	16	Message:	CHECK ALERT WHILE READING MASS STORE FILE xx
		Meaning:	Check alert status was encountered when reading mass storage file.
8	17	Message:	BLANK TAPE READING FILE xx
		Meaning:	No bits were detected during a read function.
8	18	Message:	READ ATTEMPTED AFTER END OF FILE ON FILE xx
		Meaning:	Get Next function is not allowed after an EOF condition until the current record pointer is re-established.

Major Status Code	Substatus Code		
8	19	Message:	IMPROPER MEDIA CODE VALUE SPECIFIED FOR FILE xx
		Meaning:	Media code was not defined or implemented.

Major Status Code	Substatus Code		
8	20	Message:	CONTROL CARD FROM .U FILE HAS INVALID/MISSING DELIMITER xx
		Meaning:	Delimiter is missing on control card.
8	21	Message:	ATTEMPT MADE TO OPEN THE ALREADY OPEN FILE xx
		Meaning:	An attempt was made to open an open file before it was closed.
8	22	Message:	CANNOT SPECIFY BLOCK SERIAL INDICATORS WHEN MODE IS OUTPUT OR EXTENDED FOR FILE xx
		Meaning:	UFAS can only read, not build, ANSI tape files containing block sequence indicators.
8	23	Message:	MIXED DEVICE TYPES ASSIGNED ON FILE xx
		Meaning:	More than one device was assigned to file, but not of the same type.
8	24	Message:	DEVICE ILLEGAL OR MISSING ON FILE xx
		Meaning:	Device code is not known to operating system. If ISP file, can be one of the following: ● Device or allocation is not random. ● Number of additional data file codes in FIB file code list does not match the number in the data file utilization record. ● Device is not assigned for index file.
8	25	Message:	DEVICE-TYPE/FILE-CONTROL CONFLICT ON FILE xx
		Meaning:	Abort was caused by one of the following conditions: ● User assigned 7-track tape and format is not GFRC. ● User assigned SYSOUT, but format is not GFRC. ● User assigned tape device, but file organization is not sequential. ● User assigned random device, but file format is not UFF or ISP. ● User assigned linked mass storage device, but file format is not GFRC. ● User assigned linked mass storage device, but fixed-length records were specified or BSN was not specified.

Major Status Code	Substatus Code		
			• If ISP file, the file assigned as the index or data file is not a valid ISP index or data file. • File format is GFRC, but file organization is not sequential. • Code set is ASCII, but directed to a dedicated BCD printer.
8	26	Message:	VARIABLE LENGTH RECORDS OR 9-BIT EXT CODE SET NOT SPECIFIED FOR MEDIA CODE 6 FILE xx
		Meaning:	Media code 6 GFRC file required VLR specification or 9-bit external code.
8	27	Message:	DATA BASE KEY (HIGH OR LOW) HAS BEEN CHANGED FOR FILE xx
		Meaning:	Data base key specified by user does not match key in the attribute block.
8	28	Message:	KEY OUTSIDE OF RECORD AREA FOR DATA FILE xx
		Meaning:	Improper positioning of key. Every record in an ISP file must be large enough to accommodate the key. The smallest record in the file must contain at least n characters; i.e., n = Key size + Key offset
8	29	Message:	PAGE MODE DOES NOT AGREE WITH INPUT FILE xx
		Meaning:	Control interval format of the input file does not agree with description specified for I-D-S/II.
8	30	Message:	AREA NAME DOES NOT AGREE WITH INPUT FILE xx
		Meaning:	Area name of input file does not agree with that specified for I-D-S/II.
8	31	Message:	INVALID RECORD SIZE ON FILE xx
		Meaning:	Record size does not match record size in FIB. If ISP file, record size in the record size word is not large enough to contain the entire key.
8	32	Message:	ERROR ENCOUNTERED WHILE ATTEMPTING TO BACKSPACE FILE xx
		Meaning:	Unrecognizable I/O error was encountered while attempting to backspace the file.

Major Status Code	Substatus Code		
8	33	Message:	CONTROL CARD FROM .U FILE HAS UNRECOGNIZABLE KEYWORD xx
		Meaning:	Unrecognizable keyword was specified on control card.

Major Status Code	Substatus Code		
8	34	Message:	CLOSE ATTEMPTED ON UNOPENED FILE xx
		Meaning:	Closing of an unopened file was attempted.
8	35	Message:	PROCESS AREA POINTER IN FIB MISSING ON FILE xx
		Meaning:	FIB pointer was not specified on file.
8	36	Message:	DBK'S/CALC-HDR OR LINES/PAGE DOES NOT AGREE WITH INPUT FILE xx
		Meaning:	Incompatibility of the I-D-S/II description with that of the input file.
8	37	Message:	CHECKPOINT FILE NOT OPEN AS OUTPUT
		Meaning:	Checkpoint file was not open in output processing mode.
8	38	Message:	CHECKPOINT FILE ASSIGNED TO ILLEGAL PERIPHERAL
		Meaning:	Checkpoint file must be random, linked mass storage, or magnetic tape.
8	39	Message:	TWO SUCCESSIVE WRITE ERRORS ON CHECKPOINT, CHECK FILE SIZE xx
		Meaning:	Two consecutive write errors were encountered in file.
8	40	Message:	CHECKPOINT FILE NOT ALLOCATED
		Meaning:	Checkpoints were requested when the file was opened, but the file was not allocated/open when the dump was requested.
8	41	Message:	FILE CODE FOR INDEX NOT SPECIFIED FOR INDEXED FILE xx
		Meaning:	File code missing for index of indexed file.
8	42	Message:	INDEXED FILE'S INDEX IS NOT VALID FOR FILE xx
		Meaning:	The date or time of day on the allocated index file does not match the date or time of day on the allocated data file, or one or more occurrences of alternate key types did not exist when the key file was built.
8	43	Message:	VARIABLE LENGTH RECORDS OR 6-BIT EXT CODE SET REQUIRED FOR MEDIA CODE 3 FILE xx
		Meaning:	User did not specify VLR or 6-bit external code for media code 3.
8	44	Message:	FILE FORMAT NOT UFF ON FILE xx
		Meaning:	File allocated is not an UFF file.

Major Status Code	Substatus Code		
8	45	Message:	DIRECTORY IN ERROR ON FILE xx
		Meaning:	User specified wrong directory in open FIB list.
8	46	Message:	USER HIGH PAGE IS INVALID FOR FILE xx
		Meaning:	High data base key does not agree with that of input file.
8	47	Message:	LOW PAGE IN AREA DOES NOT AGREE WITH INPUT FILE xx
		Meaning:	Low data base key does not agree with that of input file.
8	48	Message:	BUFFER POOL TABLE OVERFLOW
		Meaning:	An attempt was made to record more than 20 entries in the buffer pool table.
8	49	Message:	OPEN EXTEND IS ILLEGAL ON UNLABELED FILE xx
		Meaning:	Extend mode is not allowed for unlabeled file.
8	50	Message:	PROCESS MODE = EXTEND IS ILLEGAL ON A GFRC CREATED FILE xx
		Meaning:	Extend mode is not allowed for GFRC file not created by UFAS.
8	51	Message:	EOF MARK ENCOUNTERED INSTEAD OF INPUT LABEL ON FILE xx
		Meaning:	Required header label is missing.
8	52	Message:	INTERNAL ERROR - (LABEL PROCESSING) FOR FILE xx
		Meaning:	Caused by one of the following: ● Unexpected hardware error was encountered. ● Userlabel (UVL) request is invalid for IBM files. ● Requested function (DLRLB or DLWLB) is invalid for this userlabel exit. ● User Label record indentification is invalid. Must be UVL,UHL or UTL and defined as UASCI. ● An input file contains an invalid label structure.
8	53	Message:	ILLEGAL 'PERCENT FILL' SPECIFIED FOR FILE xx
		Meaning:	Specified percent fill is larger than the control interval.

Major Status Code	Substatus Code		
8	54	Message:	LABEL EXIT PROCESSING RECURSION LIMIT EXCEEDED ON FILE xx
		Meaning:	More than 10 outstanding label exits were specified.
8	55	Message:	VARIABLE LENGTH RECORDS OR 9-BIT EXT CODE SET NOT SPECIFIED FOR MEDIA CODE 7 FILE xx
		Meaning:	Media code 7 GFRC file requires VLR specification or 9-bit external code.
8	56	Message:	AN INVALID KEY CONDITION, REJECTED BY USER, WAS DETECTED ON FILE xx
		Meaning:	Abort was caused by one of the following conditions: ● On invalid key condition, user did not provide an alternate exit or an error procedure. ● User returned to UFAS from an error procedure with an invalid response in the substatus field.
8	57	Message:	RECORD KEY EXCEEDS FILE SPACE LIMITS ON FILE xx
		Meaning:	Key supplied by user was either negative or beyond the limits of the file. This error will occur in DMIV-TP environment when local or general inventoried overflow is not available. Since TP does not update the attribute block, general overflow is not used.
8	58	Message:	FIB HAS NO USER ERROR PROCEDURE POINTER FOR FILENAME OR PROCESSING MODE FOR FILE xx
		Meaning:	User failed to provide error procedure for file name or processing mode.
8	59	Message:	NO UPDATE PROTECTION SPECIFIED FOR RANDOM FILE xx
		Meaning:	Within a given run unit, when any permanent random file has update protection, all permanent random files must have update protection.
8	60	Message:	FILE SPACE EXHAUSTED; GEMORE REQUEST DENIED ON FILE xx
		Meaning:	File space for the specified file was increased up to the maximum size given when the file was created.
8	61	Message:	CANNOT SPECIFY PARTITIONED RECORDS JOINTLY WITH MEDIA CODE 3 or 7 OR WITH FIXED LENGTH RECORDS ON FILE xx

Major Status Code	Substatus Code		
		Meaning:	Media code 3 or 7 GFRC file requires VLR specification.
8	62	Message:	READ NEXT ATTEMPTED AFTER INV KEY CONDITION ON FILE xx
		Meaning:	Get Next function is not allowed after an invalid key condition until the current record pointer is re-established.
8	63	Message:	ILLEGAL FILE PROTECTION OPTION SPECIFIED FOR FILE xx
		Meaning:	FMS detected an improper file protection option while a before-image was being taken
8	64	Message:	UNEXPECTED FILE MARK ON FILE xx
		Meaning:	Unexpected file mark was encountered during processing of the file.
8	65	Message:	TOO MANY PAGES LOCKED ON FILE xx
		Meaning:	Number of available buffers is exhausted.
8	66	Message:	END OF TAPE ENCOUNTERED ON UNLABELED OUTPUT FILE xx
		Meaning:	An end of tape was detected on an unlabeled file.
8	67	Message:	REQUEST FOR MORE MEMORY DENIED BY GEMORE - FILE xx
		Meaning:	Work space allocated to UFAS for internal tables and buffers has been depleted.
8	68	Message:	NO BUFFERS OR TOO MANY BUFFERS ASSIGNED FOR FILE xx
		Meaning:	The number of buffers assigned is outside the acceptable limits.
8	69	Message:	NO RECORD SIZE SPECIFIED FOR FILE xx
		Meaning:	Record size was not specified.
8	70	Message:	RECORD SIZE IS GREATER THAN PAGE SIZE ON FILE xx
		Meaning:	A single record and the control information do not fit on the control interval.
8	71	Message:	INVALID FILE FORMAT SPECIFIED FOR FILE xx
		Meaning:	File format specified for file is invalid.
8	72	Message:	INVALID FILE ORGANIZATION SPECIFIED FOR FILE xx
		Meaning:	File organization specified for file is invalid.
8	73	Message:	INVALID ACCESS MODE SPECIFIED FOR FILE xx

Major Status Code	Substatus Code		
		Meaning:	Access mode specified for file is invalid.
8	74	Message:	INVALID PROCESSING MODE SPECIFIED FOR FILE xx
		Meaning:	File processing mode specified for file is invalid.
8	75	Message:	PAGE SIZE IN FIB NOT EQUAL TO FMS PAGE SIZE ON FILE xx
		Meaning:	Control interval (page) size in the FIB is not the same as the control interval size specified for FMS when the file was created.
8	76	Message:	OPEN OUTPUT WITH FMS PROTECTION REQUIRES 'LOAD' ALLOCATION ON FILE xx
		Meaning:	FMS protection is not allowed during creation of a file. Override by using load permissions on the $ PRMFL card.
8	77	Message:	OUTPUT RECORD SIZE EXCEEDS MAX SPECIFIED ON FILE xx
		Meaning:	Record size specified for Put macro (VIB) is greater than maximum record size specified in the FIB.
8	78	Message:	MUST RE-ESTABLISH CURRENT RECORD POINTER AFTER CHECKPOINT FILE xx
		Meaning:	When files with access/monitor protection are present and checkpoint is performed, the current record pointer is cleared. This pointer must be re-established before another sequential Get function is performed.
8	79	Message:	AN END OF FILE CONDITION, REJECTED BY USER, WAS DETECTED ON FILE xx
		Meaning:	Abort was caused by one of the following conditions: ● On end-of-file condition, user did not provide an alternate exit or an error procedure. ● User returned to UFAS from an error procedure with an invalid response in the substatus field.
8	81	Message:	NO PAGE SIZE OR INVALID PAGE SIZE SPECIFIED FOR FILE xx
		Meaning:	Control interval size specified is either zero or greater than 4096 words.
8	82	Message:	INVALID RECORD TYPE SPECIFIED FOR FILE xx

Major Status Code	Substatus Code		
		Meaning:	Type of record (fixed or variable) specified does not match the type specified in the tape label (HDR2).
8	84	Message:	DENSITY ERROR ON FILE xx
		Meaning:	Incorrect density was specified.
8	85	Message:	OPERATOR ABORTED HEADER/TRAILER LABEL ERROR MESSAGE ON FILE xx
		Meaning:	Operator responded with the abort option to a console message.
8	86	Message:	POSITIONING REQUIRES FUNCTION CODE IN VIB FOR FILE xx
		Meaning:	File positioning requires a non-zero content in word 1 of VIB.
8	87	Message:	CURRENT RECORD POINTER UNDEFINED xx
		Meaning:	A Rewrite or Delete function must be preceded by a Read Next function in the sequential access mode.
8	88	Message:	AN ATTEMPT TO REWRITE A RECORD WITH DIFFERENT RECORD SIZE OCCURRED ON FILE xx
		Meaning:	Record size provided for the Rewrite function, either in the FIB or VIB, must be equal to the record size of the record being replaced.
8	89	Message:	DATA FILE FILE-CODES NOT GIVEN IN SAME ORDER AS AT FILE BUILD FOR FILE xx
		Meaning:	Sequence of data file, file codes is different from that specified when file was created. If ISP file, the page read was not the one requested because file codes were out of order, or data or index file pages were destroyed.
8	90	Message:	INVALID KEY TYPE SPECIFIED FOR FILE xx
		Meaning:	Incorrect key was specified for file.
8	91	Message:	ILLEGAL CALL ON FILE xx
		Meaning:	User tried to execute a function that is not allowed by the specified processing mode and access mode, or an error was found in the catalog file description when UFAS tried to put the users filename in the Time Sharing Available File Table.
8	92	Message:	OPERATOR RESPONDED "E" TO LABEL ERROR MESSAGE ON FILE xx

Major Status Code	Substatus Code		
		Meaning:	Operator forced an end-of-file condition on the input file.
8	93	Message:	LATERAL PARITY ON TAPE, FILE xx
		Meaning:	Incorrect parity was detected across input tape file.
8	94	Message:	LONGITUDINAL PARITY ON TAPE, FILE xx
		Meaning:	Incorrect parity was detected along length of tape.
8	95	Message:	ERROR ON CHECKPOINT FILE
		Meaning:	Checkpoint was attempted while a courtesy call was outstanding.
8	96	Message:	NO ARGUMENT OR INVALID ARGUMENT IN FIB LIST FOR FILE xx
		Meaning:	Either the FIB pointer or directory pointer is missing in the FIB list.
8	97	Message:	INTERNAL OR EXTERNAL DATA TYPE IS INVALID FOR FILE xx
		Meaning:	Invalid code value was specified for character set.
8	98	Message:	THE BEGINNING OF VOLUME WAS ENCOUNTERED ON FILE xx
		Meaning:	Beginning of a volume was reached during backward file positioning.
8	99	Message:	INVALID INPUT ARGUMENT FOR FILE xx
		Meaning:	One of the VIB input arguments is incorrect.
8	100	Message:	ILLEGAL COMBINATION OF CALLS FOR FILE xx
		Meaning:	Combination of primitive calls is not acceptable (for example, forward spacing by 0 record twice for an indexed sequential).
8	101	Message:	INVALID KEY OFFSET AND KEY LENGTH GIVEN FOR ISP FILE xx
		Meaning:	Values given in the key list entry (key pointer in the FIB) do not match those in the data file utilization record.
8	102	Message:	UNABLE TO LOCATE REQUIRED KEY IN INDEX FILE xx
		Meaning:	Index file was destroyed. File must be rebuilt by unloading and loading the file with the ISP utility XUTIL.
8	103	Message:	MISPLACED 1ERI TRAILER LABEL ON FILE xx
		Meaning:	End-of-reel indicator was misplaced.

Major Status Code	Substatus Code		
8	104	Message:	DATA BLOCK HAS LABEL PARITY ON FILE xx
		Meaning:	Parity error occurred when label was read.
8	105	Message:	MAXIMUM REC SIZE IS LESS THAN ACTUAL REC SIZE ON FILE xx
		Meaning:	Incorrect record size was specified.
8	106	Message:	ACT REC SZ IS LESS THAN USER SPECIFIED SZ ON FLR FILE xx
		Meaning:	Incorrect fixed-length record size was specified.
8	107	Message:	LAST REC IN CURR BLK IS LESS THAN FIXED SIZE ON FILE xx
		Meaning:	Last record is shorter than the defined fixed-length record size.
8	108	Message:	NO ALTERNATE KEY DESCRIPTIONS PROVIDED IN FIB (SEE KEY PTR) ON FILE xx
		Meaning:	Alternate key is missing in key description list.
8	109	Message:	INVALID MAXIMUM RECORD SIZE GIVEN FOR ISP FILE xx
		Meaning:	Record size given in FIB exceeds the maximum size of the data file utilization record specified when the file was created. The value of the record size in the FIB must be less or equal to the value specified in the utilization record.
8	110	Message:	BANNER CHARS NOT ALLOWED WITH H2000 MOD4 FORMAT ON FILE xx
		Meaning:	Banner characters are allowed only with the H2000 Mod1 file format.
8	111	Message:	INVALID FILE SEQUENCE NUMBER SPECIFIED IN FIB FOR FILE xx
		Meaning:	File sequence number specified in FIB was incorrect.
8	113	Message:	ILLEGAL BANNER CHAR FOR FILE xx
		Meaning:	Banner character provided by the user is not within the legal range.
8	114	Message:	TOO FEW RECORDS IN CI CAUSE TOO MANY INDEX LEVELS; INCREASE CI SIZE FOR FILE xx
		Meaning:	Key size is almost as large as control interval size of index file, or the control interval size is too small to contain two alternate keys for a UFF Indexed or Integrated file, thus creating a control interval table overflow. Increase control interval size.

Major Status Code	Substatus Code		
8	116	Message:	ILLEGAL ATTEMPT TO OPEN IDS-2 CREATED FILE xx
		Meaning:	Files created through I-D-S/II can only be accessed through I-D-S/II. No other files can be accessed by I-D-S/II.
8	117	Message:	NO CURRENT CI FOR FILE xx
		Meaning:	UFAS attempted a function on a non-existent current control interval.
8	118	Message:	CI NOT LOCKED FOR FILE xx
		Meaning:	UFAS attempted a function on a locked control interval, but the control interval was not locked.
8	119	Message:	LOCK TABLE FULL FILE xx
		Meaning:	UFAS attempted to lock more pages than the current table space allowed.
8	120	Message:	LABEL PROCESSING NOT A TSS FEATURE
		Meaning:	An attempt was made to open a labeled file in the time sharing environment. Label processing is not available in the time sharing environment.
8	121	Message:	THIS FILE FORMAT NOT SUPPORTED BY TSS UFAS
		Meaning:	An attempt was made to open a file with a format not supported in the time sharing environment.
8	123	Message:	FILE CODE LIST NOT PERMISSIBLE IN TSS UFAS
		Meaning:	A multi-volume file cannot be specified in the time sharing environment.
8	124	Message:	INVALID ALTKEY DESCRIP OR INDEX-ID= aaaaaabbbbbb ON FILE xx
		Meaning:	The current access of an indexed or integrated file with an alternate key used a key description (or index-id) that was not provided at the time the file was built. The field aaaaaabbbbbb represents either the 12-character alternate key description (aaaaaa = offset, bbbbbb = length) or the 3 character I-D-S/II index-id.
8	125	Message:	RECRD TO DEL OR REWR NOT FOUND IN ALT KEY INDEX FILE xx
		Meaning:	An attempt was made to delete or modify a non-existing alternate key value.

8 126 Message: GENERAL INVENTORIED OVERFLOW SPACE EXHAUSTED FOR FILE xx

Meaning: An attempt was made under TP to update an indexed file whose inventoried overflow was exhausted. This condition occurs when insufficient or no inventory overflow (local or general) was defined when the file was built. Rebuild the file using the LOVI and GOVI options.

Major Status Code	Substatus Code		
8	127	Message:	FILE SIZE TOO SMALL FOR ATTRIBUTES +1 CI ON FILE xx
		Meaning:	Not enough file space was provided for the file attributes and at least one control interval.
8	128	Message:	FILE CONCATENATION TAPE ACCESS DENIED
		Meaning:	Unable to obtain allocation of the next required tape file in concatenation sequence.
8	129	Message:	FILE CONCATENATION NON-BUSY PERMFL ACCESS DENIED
		Meaning:	Unable to obtain allocation of the next required PRMFL in concatenation sequence.
8	130	Message:	INVALID USER LABEL EXIT PARAMETERS
		Meaning:	ANS, UFF or IBM User label processer have been requested with incorrect (FIBMAC) parameters.
8	131	Message:	I/O ERROR ON W* FILE
		Meaning:	An unrecoverable error has occured while attempting to read the W* file.
8	132	Message:	EXCEEDED 30 FILES IN W* FILE
		Meaning:	The maximum allowable number of files that can be described in the W* file has been exceeded.
8	133	Message:	ERRORS IN W* PARAMETERS
		Meaning:	Format or syntax errors were discovered in processing the W* file parameters.
8	134	Message:	I/O ERROR ON WRITE TO CONSOLE
		Meaning:	An I/O error occurred on the system console in processing a T* parameter card contained in the W* file.
8	135	Message:	GEMORE ERROR ON "NULL" FILE (RERUN)
		Meaning:	A GEMORE denial occurred while attempting to allocate a null file during a rerun.
8	136	Measage:	ERROR WRITING CHECKPOINT
		Meaning:	A hardware error occurred while attempting to write a checkpoint record to a data tape.
8	137	Message:	ALLOCATED OUTPUT FILE NOT IN RERUN LIST
		Meaning:	During rerun, output files have been allocated that were not specified in an OUT parameter.

8	138	Message:	NO USER ADDRESS GIVEN FOR EXTERNAL LABELS	
		Meaning:	The L option was specified in the W* file without the address of the procedure to process labels passed to UFAS. Use the DXLABL macro to pass the address to UFAS.	
8	139	Message:	END OF TAPE BEFORE REACHING BLOCK COUNT	
		Meaning:	The tape volume is too short to hold the specified number of blocks.	
8	140	Message:	INTERNAL UFAS ERROR	
		Meaning:	UFAS has encountered an internal error. No recovery is possible.	
9	0	Meaning:	Reserved.	
9	1		Block serial number error.	File Processing Conditions
9	2		Block length error on read.	
9	3		Blank tape on read.	
9	4		Rewrite record size error.	
9	5		VIB or record size word error.	
9	6		File Mark found on H2000 Input File, or File assigned to a terminal during special Time Sharing testing.	
9	7		Incorrect I/O status after Backspace command.	
9	8		Misplaced 1ERI Trailer Label found.	
9	9		Unidentified record found on H2000 tape.	
9	A		Record length < maximum length and not VLR file.	
9	B		Record length > maximum record length.	
9	C		Incorrect I/O status after attempted WRITE.	
9	D		Actual record length not equal to record length for FIXED LENGTH FILE.	
F	0	Meaning:	An F0 memory fault may take place during DINFO execution due to insufficient table area size.	
F	1	Meaning:	Invalid function code during DINFO execution.	
F	2	Meaning:	File is not labeled and the user asked for the contents of FACT section three.	

LABEL PROCESSING MESSAGES

The label processing messages provide a log of the sequence of files processed and are printed on the execution report whenever a volume of a single-file multi-volume is processed.

Message	Meaning
OPERATOR STARTED WITH #XXXXX FOR FILE CODE YY REEL SEQ #QQQ FILE SEQ #SSS OUTPUT	Operator loaded output tape reel with: Reel serial number - XXXXX File Code - YY Reel sequence number - WWW File sequence number - SSS
OPERATOR CONTINUED WITH #XXXXX FOR FILE CODE YY REEL SEQ #QQQ FILE SEQ #SSS	Operator continued multi-reel output file with: Reel serial number - XXXXX File code - YY Reel sequence number - QQQ File sequence number - SSS
INPUT STARTED WITH #XXXXX FOR FILE YY REEL SEQ #QQQ FILE SEQ #SSS	Input file was started with: Reel serial number - XXXXX File code - YY Reel sequence number - QQQ File sequence number - SSS
INPUT CONTINUED WITH #XXXXX FOR FILE CODE YY REEL SEQ #QQQ FILE SEQ #SSS	Operator continued multi-reel input file with: Reel serial number - XXXXX File code - YY Reel sequence number - QQQ File sequence number - SSS

CONSOLE MESSAGES

The following list of console messages, which are applicable to the H2000 file formats, contains the definition of the message and the operator action required to correct the error condition:

Message

BCR MT D#iccdd sssss-aa L=nnnnnn F=mmmmmm CEA where:

BCR	- block count error
MT	- magnetic tape subsystem
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- job sequence number (SNUMB)
aa	- activity number
L=nnnnnn	- trailer label block count
F=mmmmmm	- internal block count

Options

C - continue execution, ignoring the label block
E - force an end-of-file condition
A - abort program

Meaning

The block count in the trailer label does not agree with the internal block count.

Action

The operator may select option C, E, or A.

Message

BTIH MT D#iccdd sssss-aa   NA

where:

BTIH	- blank tape on input header
MT	- magnetic tape subsystem
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- SNUMB
aa	- activity number

Options

N - dismount tape and mount new tape to continue
A - abort the program

Meaning

A tapefile was opened as input and the first physical record read is blank or unreadable, or the density is incorrect for $ TAPE27/29 card.

Action

The operator may select option N or A.

Message

```
ƀƀƀƀƀBTIT  MT  D#iccdd  sssss-aa  E
```

where:

BTIT	- blank tape on input trailer
MT	- magnetic tape subsystem
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- SNUMB
aa	- activity number

Option

E - end-of-file condition

Meaning

The input trailer label is invalid or unrecognizable.

Action

None.

Message

```
IHLR  MT D#iccdd  sssss-aa
LABEL IS file ser # reel seq # filename rrrrr
SHOULD BE file ser # reel seq # filename NAIC
```

where:

IHLR	-	input header label error
MT	-	magnetic tape subsystem
D#i	-	IOM number
cc	-	channel
dd	-	device number
sssss	-	SNUMB
aa	-	activity number
LABEL IS	-	information from input header label
SHOULD BE	-	internal information

Options

N - dismount tape and mount new tape to continue
A - abort the program
I - ignore any subsequent IHLR processing
C - continue execution, ignoring the label check

Meaning

An error exists on the input header label. The information on the tape label does not agree with that generated internally.

Action

The operator may select option N, A, I, or C.

Message

```
ƀƀƀƀƀLABEL MT D#iccdd sssss-aa rrrrr
```

where:

```
MT    - magnetic tape subsystem
D#i   - IOM number
cc    - channel
dd    - device number
sssss - SNUMB
aa    - activity number
rrrrr - reel number
```

Meaning

Informs operator that the file opened as output has a retention period for one or more days so that an external label identifying the data may be prepared.

Action

None.

Message

```
OHLR  MT  D#iccdd  sssss-aa  rrrrr  literal  reply
```

where:

```
OHLR  - output header label error
MT    - magnetic tape subsystem
D#i   - IOM number
cc    - channel
dd    - device number
sssss - SNUMB
aa    - activity number
rrrrr - reel number

Literals (type of discrepancy)

NOTLAB - output tape has no label.
PREV   - output reel number entered in response to a LOCATE message
         does not agree with the number of the mounted reel.
VERIFY - file serial number in the label does not agree with
         the file serial number provided on tape control card.
UNEXP  - output header label indicates unexpired retention.
PERM   - output header label indicates permanent retention.

Reply (combination of the character options available,
      indicating that operator response is needed)
```

Options

N - dismount tape and mount new tape to continue
A - abort program
C - continue execution, ignoring the label check
I - accept the mounted tape
- build a new label using new reel number

Meaning

A discrepancy was detected in the output header label. The type of discrepancy is indicated by the literal.

Action

The operator may select any of the options N, A, C, I, or #.

Message

```
RCE  MT  D#iccdd  sssss-aa  L=nnnnnnnnnn F=mmmmmmmmmm CEA
```

where:

RCE	- record count error
MT	- magnetic tape subsystem
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- SNUMB
aa	- activity number
L=nnnnnnnnnn	- trailer label record count
F=mmmmmmmmmm	- internal record count

Options

C - continue executjon, ignoring the label check
E - force an end-of-file condition
A - abort program

Meaning

The logical record count in the trailer label does not agree with the internal record count.

Action

The operator may select option C, E, or A.

Message

```
RDY MT D#iccdd sssss-aa filename rrrr yyyy processing mode reply
```

where:

RDY	- ready
MT	- magnetic tape subsystem
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- SNUMB
aa	- activity number
filename	- file name from $ TAPE27/29 card
rrrr	- reel serial number
yyyy	- reel sequence number
processing mode	- input or output
reply	- character options available, indicating that operator response is needed

Options

C - continue execution
E - force an end-of-file condition (input only)

Meaning

Instructs the operator to mount the indicated tape reel and ready the device.

Action

The operator may select option C or E for input, but can only use C for output.

NOTE: The literal PREV may occur in the sequence number field of the output message when the operator has mounted a reel (part of a multi-reel set) that is too far along in the set. The operator should select the correct tape from earlier in the set, mount it, and respond "C".

Message

TRL ER D#iccdd sssss-aa filename rrrr yyyy processing mode only

where:

TRL ER	- Trailer label error
D#i	- IOM number
cc	- channel
dd	- device number
sssss	- SNUMB
aa	- activity number
filename	- file name from $ TAPE 27/29 card
rrrr	- reel serial number
yyyy	- reel sequence number
processing mode	- input or output
reply	- character options available, indicating operator response is needed

Options:

C - continue execution
F - force an end-of-file condition(input only)
A - abort program

Meaning

Informs the operator that the Trailer Label groups on the input reel jus dismounted was incorrect or missing.

Action

The operator may continue processing by mounting the indicated reel, an responding with a "C".

The operator may set and end-fo-file condition and continue processing b responding with an "E".

The operator may abort the program by responding with an "A".

Message

USW MT D#iccd sssss-aa (D#iccd) filename SEQ#xxx
rrrr processing mode

where:

USW	-	unit switch
MT	-	magnetic tape subsystem
D#i	-	IOM number
cc	-	channel
dd	-	device number
sssss	-	SNUMB
aa	-	activity number
(D#iccdd)	-	alternate IOM channel, and device number
filename	-	file name from $ TAPE27/29 card
SEQ#xxx	-	reel sequence number
rrrrr	-	reel number
processing mode	-	input or output

Meaning

The alternate or secondary device was in a ready state during unit switching and an automatic unit switch occurred.

Action

None.

Message

VERIFY D#iccdd sssss-aa filename ENDING rr mode reply

where:

VERIFY	-	request from operator to verify
D#i	-	IOM number
cc	-	channel
dd	-	device number
sssss	-	SNUMB
aa	-	activity number
filename	-	filename if present on tape card
ENDING	-	operator ended a previous RDY.MT message
rrr	-	reel sequence number
mode	-	processing mode: IN, OUT, or EXT
reply	-	operator response is needed

reply Options:

Y - Yes, end with last volume.
N - No, do not end. Reissue
RDY.MT message.

Meaning:

Requests that the operator verify the ending of the input file.

Action:

A yes (Y) response causes an end-of-file condition to be set-up for the input file. A no (N) response causes the RDY.MT message to be reissued. Processing can continue when the next volume is mounted.

SECTION XI

DEDICATED PRINTER CAPABILITY

UFAS provides an interface to a dedicated online printer for those applications requiring forms control. This capability is available for BCD and ASCII output.

To use a dedicated printer, the user program must create a FIB with the FIBMAC macro and must open the printer (which is treated as another file) with the DOPEN macro. The DLPUT macro writes data to the printer. The printer is defined as a GFRC sequential file.

ALLOCATING A PRINTER

The printer must be allocated as a dedicated device via an external control card.

An external control card must be used to direct the Peripheral Allocator to allocate an online printer and to provide information to UFAS. The printer cannot be assigned to SYSOUT.

The control card is:

```
$ PRINT FC,...
```

where:

FC is the file code used in the UFAS macros to reference the printer.

See the Control Cards Reference Manual "Control Card Descriptions" section for full details on the parameters and use of the $ PRINT card.

After the printer is allocated, the user program may direct output to it. The printer must be specified as a GFRC Sequential file with any of the following media codes:

Media Code	Meaning
3	BCD Print line image
7	ASCII Print line image
9	BCD Print line image with two-character file code embedded
13	ASCII Print line image with two-character file code embedded

PROCEDURES

The COBOL-74 or GMAP user developing programs that write to a dedicated printer should be aware of the following:

- The user program is responsible for processing the Printer Button Interface (PBI) status returned from UFAS.
- While the COBOL-74 FIL-STATS field (or GMAP FSCODE field) is optional, it must be specified if the user wants to know which PBI was pressed by the operator. This information allows the user program to perform different functions depending on what status is returned.
- An error procedure for the output file is optional. If it is provided, any PBI status that is detected causes control to be passed to the error procedure. No commands to the printer are allowed while the error procedure is in control.
- The decision of whether to specify buffers (COBOL-74 Reserve clause or FIBMAC NBUF parameter) affects the trade-off between performance and control.
    - If no buffers are specified, UFAS defaults to one System Standard Format (SSF) buffer. Data is written to the printer when the buffer is full or when 64 print/slew lines have been sent to UFAS. The advantage of this approach is higher performance of the print job. The disadvantage is less control because PBI status is returned only after the entire buffer is written. The status could represent an interrupt that occurred while printing one of the first records in the buffer. This limits the flexibility of a program to perform logical routines in response to a PBI.
    - If any number of buffers are specified, UFAS writes each record as it is received from the user program. This approach gives more control to the user program because status is returned after each record. The disadvantage of this approach is the reduction in performance due to the increased number of printer connects.
- After the user program has issued all its writes, the printer can be deallocated before the end of the activity by using a close-with-lock option. See the COBOL-74 Reference Manual for information on the close option.

PRINTER STATUS

If the user program provides a FIL-STATS or FSCODE field, a two-byte ASCII code is returned after each write to the printer. A code of 00 indicates all processing was completed. Other codes represent the following PBI statuses:

Major Status (byte 1) (octal)	Substatus (byte 2) (octal)		Meaning
always 0 (060)	0	(060)	normal
	1	(061)	print 1 line
	2	(062)	forward space
	3	(063)	forward T.O.P.
	4	(064)	invalid line
	5	(065)	reverse/rewind
	6	(066)	backspace
	7	(067)	backspace T.O.P.

It is important to note that any print command that is interrupted by a manual halt will be completed before the PBI status is returned to the user program. Therefore, the operator/user should be aware of possible forms adjustment when returning the printer to the ready state. This condition is more evident when using the multiple buffering option.

FILE INFORMATION BLOCK MACRO

The file information block macro provides the GMAP user with an efficient means of communicating the specific characteristics of a printer file to UFAS. However, before any communication can be established with UFAS, the user must invoke the macro package with the command LODM .DMAC.

All file processing information between the user program and UFAS is channeled through the FIB macro. For each printer to be accessed, a FIB macro must be coded in the user program to define the characteristics of the file and the required processing parameters.

Format

The FIB macro for a sequential file has the following format:

```
FIBMAC fc,GFRC,[parameter-name-1,parameter-value-1,...,

           parameter-name-n,parameter-value-n]
```

where:

fc - file code assigned to file

GFRC - is the only applicable file format indicator

Multiple parameter-name and parameter-value combinations constitute a keyword-argument arrangement. These combinations may appear in any order within the brackets. If a combination of parameter-name and parameter-value is omitted, a default value is assumed. If the number of combinations exceeds one line, an ETC card must be used to continue on the following line.

Only two parameter-values are required for the process area (file status code and file name) but three are required for the retention period. Also, no symbol that defines an address can be repeated in another FIB macro. In order to repeat the same address in more than one FIB macro (e.g., a common process area), multiple symbols defining the same address must be used. Do not specify default values for parameters that are not required; instead allow the default value to be applied automatically.

The following list contains all the parameters and their respective defaults which are applied if the parameter name and value are not specified:

Parameter	Name	Value	Default	Comments
Number of Buffers Allocated by UFAS	NBUF	Integer <u><</u>255	1	For a dedicated printer
		Integer <u><</u>2 (tape)		
External code	EXTCOD			Code set of the data on the external device. Also, used with the internal code set parameter to establish the size of the characters within the record and the process area, so that the records may be converted to this code set as they are moved to the buffer.
		ASC	ASC	ASCII character set.
		GBCD		GBCD (GFRC BCD).
Internal code	INTCOD			Code set of the data in the process area. Records are converted to this code set as they are moved to the process area.
		ASC	ASC	ASCII character set.
		EBCD		EBCDIC character set.
		GBCD		GBCD character set.
		HBCD		HBCD character set.
		JIS		JIS character set.
Process area	PROAR	Address, character		A required parameter indicating the location and starting character or byte position of an area within the user program from which records are retrieved. For GFRC Files, character position is assumed to be zero. The size of the process area is assumed to be equal to the FIB record size. If this parameter is not specified, the in-place processing parameter must be specified.
File status code	FSCODE	Address, character		Location and starting byte position of a two-byte ASCII file status code field in which UFAS stores a unique status code whenever a PBI condition occurs.
			0,0	No file status code field.

Parameter	Name	Value	Default	Comments
User error Procedure	USERER	address		Location of user-supplied error procedure to which UFAS transfers control whenever a fatal error is detected.
			0	No user error procedure.
Media code	MEDCOD			
		3		BCD print line image.
		7		ASCII print line image.
		9		BCD print line image with report code as first two characters of a logical record.
		13		ASCII print line image with report code as first two characters of a logical record.

Rules

1. The file information block has several assigned fields. If a field is not applicable to a particular situation, it is ignored. Fields that are set in a conflicting manner generate an error.

2. A process area is required.

FUNCTIONAL MACROS

Even though each one of the functional macros available performs a specific function and has its own format and set of rules, which are described separately in detail, certain conventions and rules are common to all functions.

The Open and Close macros contain the information that is necessary to open or close one or more files with one statement. A common FIB list is built and is used by all files to be either opened or closed. The user can either reserve space in the program for the FIB list, or can allow the list to be built by the macro.

In order to execute a functional macro, information pertinent to the function must be supplied in a variable information block (VIB). The VIB address parameter specifies where the variable information is to be located. This parameter may be either zero or omitted if no parameters are required. If parameters are required, the user supplies values in the macro parameter fields to indicate the values to be entered in the VIB for execution of the function.

Parameters are expressed in pairs containing a keyword and value. Commas must be used to separate each parameter name from the parameter value and two consecutive pairs of parameters. Parameter names ending in "I" indicate an indirect reference to the parameter value. Parameters may be specified in any order. If a parameter and its value are omitted, the default value is assumed.

Normally, registers are saved when UFAS is entered and restored when UFAS returns control to the user, except index register 1 which is the transfer register. However, when a user error or label exit procedure is entered, the user registers are not restored because UFAS is relinquishing control only temporarily to allow the user to process the exception condition. When control is returned to UFAS (at the completion of the error or label procedure), the saved registers are still valid.

Open Macro - DOPEN

Function

The Open macro initiates the processing of a printer, verifies its existence, and establishes tables for processing.

Format

```
DOPEN { FIBLS,address of FIB list               }   ,[parameter-name-1,
      { FIBLSI,address of pointer to FIB list   }

         parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as Block Starting Symbol (BSS) 3.

The parameters and respective defaults for this function are as follows:

Parameter	Name	Value	Default	Comments
FIB				Required Location of FIB for file to be opened.
	FIBA	Address		Location of FIB.
	FIBAI	Address		Location of pointer to FIB.
Routine package SYMREF name				Required Location of a UFAS symbol reference (SYMREF) that corresponds to a set of routines used to process the file.
	SYM	DPRNT		Package of routines needed to process the file.
	SYMI	Address		Location of the pointer to the DPRNT package.
Processing mode indicator				
	MODE	OT	OT	Output
	MODEI	Address		Location of processing mode (OT) indicator (2 ASCII uppercase characters in bits 0-17).

Rules

1. The Open macro must be executed before the execution of any input/output statements for a file.

2. After the initial execution of an Open macro for a file, each subsequent DOPEN for this file must be preceded by the execution of a Close macro.

3. The Open macro does not obtain or release the first data record.

Example

Open a printer. This Open macro contains the address of the FIB list (LIST) in which the FIB address (FIB) is stored. Note that the MODEI parameter references a location that contains the processing mode indicator "OT".

```
1      8      16
______________________________________________________________
       LODM   .DMAC
       DOPEN  FIBLS,LIST,[FIBA,FIB,SYM,DPRNT,MODEI,ACCMOD]
       .
       .
       .
LIST   BSS    3
FIB    FIBMAC GO,GFRC,[,PROAR,CARD,0,FSCODE,STAT,0,
       ETC    EXTCOD,ASC,INTCOD,ASC,USERER,ERR]
       .
       .
       .
ACCOM  UASCI  1,OT
CARD   BSS    20
```

Put Macro - DLPUT

Function

The Put macro places a logical record in the file.

Format

```
DLPUT  FIB address,VIB address,,

       [VIB-parameter-name-1,VIB-parameter-value-1,...,

       VIB-parameter-name-n,VIB-parameter-value-n]
```

NOTE: VIB must be defined as BSS 2.

The Put macro has the following parameters and respective defaults:

Parameter	Name	Value	Default	Comments
Record size				Optional Used with variable-length records, but ignored with fixed-length records.
	RSZ	Integer		Must be less than or equal to RSZ in FIB.
	RSZI	Address		Location of field containing record size in bits 0-17.
Report code				Optional Applies only to GFRC files. If a negative value is supplied, a default report code, assigned when the file is opened, is used.
	RPCOD	1≤Integer <63	0	
	RPCODI	Address		Location of report code field in bits 18-35.
Number of lines to slew				Required Only if slew code SLCOD(I) is 4 or 5. Applies only to System Output and GFRC files.
	SLEWN	Address		Location of word containing number of lines to slew in bits 18-35.
Slew code				Optional Applies only to GFRC files. Specifies spacing before or after print.
	SLCOD	0	0	Print after slewing one line.
		1		Print before slewing any lines.
		2		Print before top of page.
		3		Print after top of page.
		4		Print before slewing the number of lines specified in SLEWN.
		5		Print after slewing the number of lines specified in SLEWN.
	SLCODI	Address		Location of word containing slew code in bits 18-20 (bits 21-35 must be zero).
Indirect FIB addressing	FIBI	None		Optional If this keyword is specified, it implies that argument 1 is not the actual FIB address, but is a pointer to a location that contains the FIB address.

Rules

1. The designated printer must be open at the time of the execution of this statement.

2. The record is moved from the process area to the buffer. If an output transliteration table is specified, the entire record is transliterated from the internal code set to the external code set as it is moved.

Example

Send a 70-character line to the printer specified by FIB2. The line includes a slew code to print the line before slewing (15 lines) and a report code of 25.

```
1      8        16
_____________________________________________________________________
       DLPUT    FIB2,VIB2,,[RSZI,ADDSIZ,SLEWN,SL15,SLCOD,4,FIBI]
       .
       .
       .
FIB2   ARG      FIBX
       .
       .
       .
VIB2   BSS      2
       .
       .
       .
ADDSIZ ZERO     70,0
SL15   DEC      15
FIBX   FIBMAC   PR,GFRC,[PROAR,LINE,0,MEDCOD,7]
       .
       .
       .
LINE   BSS      40
```

Close Macro - DCLOS

Functions

The Close macro terminates the processing of a dedicated printer.

Format

```
DCLOS { FIBLS,address of FIB list              }  ,[parameter-name-1,
      { FIBLSI,address of pointer to FIB list  }

        parameter-value-1,...,parameter-name-n,parameter-value-n]
```

NOTE: FIB list must be defined as BSS 2.

The following parameters and respective defaults apply:

Parameter	Name	Value	Default	Comments
FIB address				Required Location of FIB for file to be closed.
	FIBA	Address		Location of FIB
	FIBAI	Address		Location of pointer to FIB
	REW	RWFL	RWFL	Rewind file
		RLFL		Rewind and lock file
	REWI	Address		Location of rewind indicator
	REWV	0	0	Rewind file
		2		Rewind and lock

Rules:

1. The Close macro can only be executed for a printer that was opened with DOPEN.

Example

Close and deallocate the printer specified by the file code in the FIB.

```
1      8       16
       DCLOS   FIBLS,LIST2,[FIBAI,ADDFIB,REW,RLFL]
       .
       .
       .
ADDFIB ARG     FIB
LIST2  BSS     2
       .
       .
       .
FIB    FIBMAC  CF,GFRC,[PROAR,RECORD,0]
RECORD BSS     20
```

SECTION XII

RERUN AND TAPE UTILITIES

The rerun and tape utility functions provide labeled input tape density determination, tape management by file name (in lieu of serial number), restart capabilities, rerun capabilities, external tape label creation, checkpoint notification, and logical and physical record count display for sequential files.

The functions are:

- Adjust Density -- UFAS attempts to set labeled input ANSI, IBM, UFF, GFRC, and H2000 tape handlers to the correct density when the user specified or system default density is incorrect.
- File Name Identified Tape (FNIT) -- This feature permits the user to manage his tape library using file names in lieu of the conventional tape reel serial number system. This feature is applicable to labeled ANSI, IBM, UFF, and GFRC single-file, single- or multi-extent tapes.
- Rerun Functions:
  - Defective Volume Replacement -- This feature allows the user to recreate defective volumes of a sequential single- or multi-volume ANSI, UFF, or GFRC labeled tape file.
  - Activity Restart -- This feature allows the user to restart an activity from a defined checkpoint on tape or a permanent mass store file. Only labeled ANSI, UFF, and GFRC single- or multi-volume tape files are checkpointed.

    The COBOL user can designate an alternate checkpoint file. The alternate file must be on mass storage with a file code of "RX". When this file is specified, checkpoints are written to both files alternately. This feature requires the use of the COBOL "RERUN" sntax.
  - Volume Size -- This feature provides the capability to specify the number of physical records (blocks) to be written to each volume of an output ANSI, UFF, or GFRC labeled, single- or multi-volume tape file.

    The volume Size feature must be invoked by the user program if Defective Volume Replacement is to be performed.
  - User External Labels -- This feature provides an exit from UFAS to a user defined procedure to allow the user program to print external tape labels on the display media (console typewriter or printer) of the user's choice.

    UFAS passes tape label data to a user supplied label data area. This feature is applicable to both labeled and unlabeled ANSI, UFF, and GFRC single- or multi-volume tape file.

    - Checkpoint Record Tracking -- This feature provides checkpoint tracking messages on the Execution Report ($$). Checkpoints are identified by file code and cumulative count.

- Physical and Logical Record Tracking -- The physical (block) and logical (record) content of input and output sequential tape and mass storage files processed by UFAS are displayed by file code on the Execution Report ($$) at file CLOSE. This feature is applicable to all sequential file formats.

- Dedicated Tape -- This feature allows the operator to assign tape units to a particular multi-volume file and release tape units so assigned.

These functions are invoked through an Executive interface module. User programs are required to affect loading of the required sub-routines by use of the $ USE .DRUOA directive. .DRUOA is the primary SYMDEF in the Executive module.

ADJUST DENSITY

The Adjust Density feature requires no external user interface. UFAS Label Processor (SSAs) automatically attempt to determine proper density while reading input tape labels.

FILE NAME IDENTIFIED TAPE (FNIT)

The user applications must provide a tape serial number of "99999" and a file name in the form:

aaaaaaaa-nnn

where:

The whole name can contain a maximum of 12 characters. aaaaaaa can be a 1- to 8-character alphanumeric string. The hyphen (-) is a required separator. nnn is three character numeric string that specifies the generation number of the file.

The name is specified in the JCL $ TAPE cards.

File names supplied via the $ TAPE cards are placed in the ANSI, IBM, UFF, and GFRC output file header labels. A file name supplied for input files are verified against the name in the file header label.

Example

```
1      8       16
$      TAPE9   IN,A1D,,99999,,TAPEFILE-001,,DEN17
```

RERUN FUNCTIONS

Defective Volume Replacement

The defective volume replacement function is invoked by executing program "S31A". The function is controlled by the RERUN directive. The program directives are placed in the temporary data file V* which must be allocated by the user.

The format of the RERUN directive is:

```
RERUN,FCOD/fc/
     [,NVOL/nn/]
     [,NEXT/sssss/]
     [,CPY/fc[,nnnnn][,ppppp]/]
     [,FNAME/filename/]
     [,OUT/fc1[,...]/]
```

where:

FCOD/fc/ is a required parameter that specifies the field code of the defective volume that is to be replaced.

NVOL/nn/ is an optional parameter that specifies the number of defective volumes that are to be replaced. If this parameter is omitted, the default is 1.

NEXT/sssss/ is an optional parameter that specifies the serial number of the volume following the last specified defective volume. If this parameter is omitted, the default is 99999. If the number is specified, it is placed in the trailer label of the last recreated volume (GFRC and UFF file formats only).

CPY/fc,nnnnn,ppppp/ is an optional parameter that specifies the file code of the defective volume set. nnnnn specifies the tape serial number of the new volume. If omitted, this parameter defaults to 99999. ppppp specifies the tape serial number of the volume that precedes the volume or volumes being recreated. If omitted, this parameter defaults to 99999.

FNAME/filename/ is an optional parameter that specifies the file name contained in the checkpoint tape header label. If it is specified, it is compared with the file name from the tape header label. This option is applicable only to non-FNIT tapes.

OUT/fc/ is an optional parameter that specifies the file code or codes from the restarted activity. A maximum of 14 files can be specified. The output files must be allocated via the $ TAPE or the $ PRMFL JCL cards.

The directives consist of keywords and values. Keywords are separated from their associated optional values by beginning and ending slashes (/). Keywords as well as multiple values within a keyword are separated by commas. A blank preceded by a slash terminates the directive.

Directives must begin in column 1.

If the optional CPY directive is not used, the file beginning label (or label set) and the checkpoint record on the original defective volume are skipped over and the recreated data is written over the original volume.

The JCL required to invoke the defective volume replacement is:

```
1      8        16
$      IDENT
$      PROGRAM  S31A
$      LIMITS   (see following explanation)
$      DATA     V*
RERUN...
$      TAPE     ...
$      PRMFL    ...
       .
       .
       .
$      ENDJOB
```

Program LIMITS must be set to program execution memory required plus 4K. If the RERUN program (S31A) determines that insufficient memory has been allocated to the activity, it attempts to GEMORE the additional memory required. The additional 4K of memory allocated for the RERUN program is released before the call to GEROLL.

All required files must be allocated by the user. If the checkpoint file is on tape the user must allocate a permanent file (file code YY) to which the RERUN program copies the tape checkpoint. For example:

```
1      8        16
$      PRMFL    YY,W,R,cat/file descriptor
```

Activity Restart

The activity restart function is invoked by executing the S31A program. The function is controlled by the RSTRT directive. The directives must be placed in the temporary data file V* which must be allocated by the user.

The format of the RSTRT directive is:

```
RSTRT,FCOD/fc/
[,CKPTN/nn/]
[,FNAME/filename/]
```

where:

FCOD/fc/ is a required parameter that specifies the file code of the checkpoint file.

CKPTN/nn/ is an optional parameter that specifies the checkpoint number from which the activity is to be restarted. If this parameter is omitted, the activity is restarted from the last checkpoint that was taken.

FNAME/filename/ is an optional parameter that specifies the file name contained in the checkpoint tape header label. If this parameter is present it is compared with the file name from the tape header label. This option is applicable only to non-FNIT tapes.

The directives consist of keywords and values. Keywords are separated from their associated optional values by beginning and ending slashes (/). Keywords as well as multiple values within a keyword are separated by commas. A blank preceded by a slash terminated the directive.

Directives must begin in column 1.

The JCL required to invoke the activity restart function is:

```
1      8       16
$      IDENT
$      PROGRAM S31A
$      LIMITS  (see following explanation)
$      DATA    V*
RSTRT...
$      TAPE    ...
$      PRMFL   ...
       .
       .
       .
$      ENDJOB
```

Program LIMITS must be set to program execution memory required plus 4K. If the RSTRT program (S31A) determines that insufficient memory has been allocated to the activity, it attempts to GEMORE the additional memory required. The additional 4K of memory allocated for the RSTRT program is released before the call to GEROLL.

All required files must be allocated by the user. If the checkpoint file is on tape the user must allocate a permanent file (file code YY) to which the RSTRT program copies the tape checkpoint. For example:

```
1      8       16
$      PRFML   YY,W,R,cat/file descriptor
```

Print Function

The Print function provides a means of scanning a checkpoint file for checkpoint information. The Print function is invoked by the S31A program. The function is controlled by the PRINT directive.

The format of the PRINT directive is:

```
PRINT,FCOD/fc/
```

where:

FCOD/fc/ is a required parameter that specifies the file code of the checkpoint file.

The directives consist of keywords and values. Keywords are separated from their associated optional values by beginning and ending slashes (/). Keywords as well as multiple values within a keyword are separated by commas. A blank preceded by a slash terminates the directive.

Directives must begin in column 1.

The JCL required to invoke the Print function is:

```
1      8       16
$      IDENT
$      PROGRAM S31A
$      LIMITS  ,4K
$      DATA    V*
PRINT...
$      PRMFL   fc,R,cat/file descriptor
$      ENDJOB
```

Volume Size Function

The volume size function is used in conjunction with the replace defective volume function. The volume size function is controlled by directives on the temporary data file W*.

The format of the volume size directive is:

fc,Bnnnnn[,L]

where:

fc is a required parameter that specifies the file code of the file on which the volume size is to be limited.

Bnnnnn is a required parameter that specifies the number of blocks (physical records) to be written on each volume of the file. nnnnn is limited to 5 digits.

L is an optional parameter that specifies that the external label exit is to be used for this file. This function is described in the following subsection.

Directives begin in column 1. Parameters are separated by commas (,). A blank terminates the directive. Only one file code specific directive is permitted on a line.

If the directive file code is T*, the content of the directive string from column 3 through 80 is printed on the console typewriter.

DEDICATED TAPE

The user JCL may not specify an alternate tape unit in the $ TAPE control cards if operator tape dedication is in use for the pertinent file code. For example:

```
1       8       16
$       TAPE9   fc,LUD,,12345,1,MYFILE,,DEN16
```

APPENDIX A

PROGRAMMING AIDS

MAGNETIC TAPE DENSITIES

The UFAS tape density settings are applicable to the Microprogrammed Peripheral Controller (MPC) 500 and 600 tape subsystems, ASA subsystems, and mixed subsystems (a combination of the two types of subsystems). Density settings of 6250 bpi require an MPC 0610.

Densities can be set via the $ TAPE control card or by default. UFAS does not allow the user to set density internally.

The MPC subsystems and the ASA subsystems use the densities specified at system startup time or on the $ TAPE card option DENn. When specified, the $ TAPE control card setting always prevails. For detailed information about the densities pertinent to the various subsystems, see Tables A-1, A-2, and A-3.

Table A-1. MPC Magnetic Tape Subsystems

SYSTEM STARTUP SPECIFIED DENSITIES		$ TAPE DENSITY SPECIFIED/ALLOCATED					
9-TRACK TAPE							
HIGH	LOW	DEN16	DEN8	DEN5	DEN2	NONE	DEN9
6250	N/A		REQUESTED DENSITY WILL BE ALLOCATED			6250	UNCHANGED
1600	N/A		REQUESTED DENSITY WILL BE ALLOCATED			1600	UNCHANGED
$\boxed{800}$	N/A		REQUESTED DENSITY WILL BE ALLOCATED			800	UNCHANGED
556	N/A		REQUESTED DENSITY WILL BE ALLOCATED			556	UNCHANGED
200	N/A		REQUESTED DENSITY WILL BE ALLOCATED			200	UNCHANGED
7-TRACK TAPE							
N/A	$\boxed{800}$	Illegal	REQUESTED DENSITY WILL BE ALLOCATED			800	UNCHANGED
N/A	556	Illegal	REQUESTED DENSITY WILL BE ALLOCATED			556	UNCHANGED

Table A-1 (cont) MPC Magnetic Tape Subsystems

SYSTEM STARTUP SPECIFIED DENSITIES		$ TAPE DENSITY SPECIFIED/ALLOCATED			
N/A	200	Illegal	REQUESTED DENSITY WILL BE ALLOCATED	200	UNCHANGED

NOTE: If system startup densities are not specified on the $ INFO card, the default indicated in the outlined block is assumed.

Table A-2. ASA Tape Subsystems

SYSTEM STARTUP SPECIFIED DENSITIES		$ TAPE DENSITY SPECIFIED/ALLOCATED					
HIGH	LOW	DEN16	DEN8	DEN5	DEN2	NONE	DEN9
6250[6]	6250	ILLEGAL					
1600[6]	1600	ILLEGAL					
1600[6]	800	ILLEGAL					
1600[6]	556	ILLEGAL					
1600[6]	200	ILLEGAL					
800	800	ILLEGAL					
800	556[5]	–[6]	800	556	–[1]	800	UNCHANGED
800	200[2] [7]	–[6]	800	–[1]	200	800	UNCHANGED
556	556[4]	–[6]	–[1]	556	–[1]	556	UNCHANGED
556	200[3]	–[6]	–[1]	556	200	556	UNCHANGED
200	200	ILLEGAL					

NOTES:

1. When the user specifies on the $ TAPE control card a density not listed in Tables A-1, A-2, and A-3, the program aborts.
2. Density setting: 800 bpi high and 200 bpi low
3. Density setting: 556 bpi high and 200 bpi low
4. Density setting: 556 bpi high and 556 bpi low
5. Density setting: 800 bpi high and 556 bpi low

} ASA Tape Subsystem

6. Illegal
7. If system startup densities are not specified on the $ INFO card, the default indicated in the outlined block is assumed.

Table A-3. Mixed Subsystems

SYSTEM STARTUP SPECIFIED DENSITIES		$ TAPE DENSITY SPECIFIED/ALLOCATED					
HIGH	LOW	DEN16	DEN8	DEN5	DEN2	NONE	DEN9
6250	6250			ILLEGAL			UNCHANGED
1600	1600			ILLEGAL			UNCHANGED
1600	800[2]	1600[6]	800/800	556[1]	200/200	1600/800	UNCHANGED
1600	556[3]	1600[6]	800[1]	556/556	200/200	1600/800	UNCHANGED
1600	200[2]	1600[6]	800/800	556[1]	200/200	1600/200	UNCHANGED
800	800			ILLEGAL			UNCHANGED
800	556[5]	1600[6]	800/800	556/556	200[1]	800/800	UNCHANGED
800	200[2] [7]	1600[6]	800/800	556[1]	200/200	800/800	UNCHANGED
556	556[4]	1600[6]	800[1]	556/556	200[1]	556/556	UNCHANGED
556	200[3]	1600[6]	800[1]	556/556	200/200	556/556	UNCHANGED
200	200			ILLEGAL			UNCHANGED

NOTES:

1. When the user specifies on the $ TAPE control card a density not listed in Tables A-1, A-2, and A-3, the program aborts.
2. Density setting: 800 bpi high and 200 bpi low
3. Density setting: 556 bpi high and 200 bpi low
4. Density setting: 556 bpi high and 556 bpi low
5. Density setting: 800 bpi high and 556 bpi low

   ASA Tape Subsystem
6. Illegal
7. If system startup densities are not specified on the $ INFO card, the default indicated in the outlined block is assumed.

CHECKPOINT RECORDS ON DATA TAPES

Checkpoint records delimited by end-of-file 00 on data tapes are bypassed on UFF sequential, H2000, and GFRC files, except on data tapes that contain these checkpoints recorded at 1600 bits per inch on an MPC600. The hardware issues an EOF mark, but transmits a standard EOF character (17 or 23) to the software. If tapes with checkpoint records are to be read on an MPC600, the tapes must be copied in advance to eliminate the checkpoint records.

Files with checkpoints are delimited by standard EOF marks because an MPC600 does not allow non-standard EOF marks. Consequently, the user must provide a separate checkpoint file if checkpoints are required. Checkpoints must never be included in data tapes on UFF sequential, H2000, and GFRC files.

CONTROL CARDS

Tape or mass storage control cards must be supplied by the user at execution time to specify the devices to be selected for each file. Mass storage devices for ISP and UFF files must be allocated as random (see Control Cards Reference Manual).

Buffer Pooling (UFF Relative, Indexed, And Integrated Only)

In order to change buffer space assignment and to specify buffer pooling, the user must provide, when the program is ready for execution, a $ DATA .U file and control cards having the following format:

```
      1       8
1 -   FILE    FC/XX/,PLID/YYYYYY/,NBUF/NNNN/,BFSZ/MMMMM/,LOVI/III/,GOVI/GGG/
```

X - 2-byte file code of the file to which card applies.

Y - 1 to 6 alphanumeric characters for name of pool; all file cards that refer to the same pool must have the same name in this field.

N - 1 to 4 numeric characters to indicate the number of buffers in the pool.

M - 1 to 5 numeric characters to specify the number of characters in the largest control interval of all the files in the pool.

I - 1 to 3 numeric characters, ranging from 1 to 510, which will override LOVINC in the FIB for UFF indexed files in build mode. This parameter represents a ratio of data control intervals to overflow control intervals that will be allowed in the file during build mode.

G - 1 to 3 numeric characters, ranging from 1 to 510, which will override GOVINC in the FIB for indexed files in build mode. This parameter represents a ratio of data control intervals to overflow control intervals that will be allowed in the file during build mode.

Although the parameters are optional some are interdependent and cannot be used separately. The following formats are valid:

```
      1       8
2 -   FILE    FC/XX/,PLID/YYYYYY/,NBUF/NNNN/,BFSZ/MMMMM/

3 -   FILE    PLID/YYYYYY/,NBUF/NNNN/,BFSZ/MMMMM/

4 -   FILE    PLID/IDS/,NBUF/NNNN/,BFSZ/MMMMM/

5 -   FILE    FC/XX/,NBUF/NNNN/,BFSZ/MMMMM/

6 -   FILE    FC/XX/,PLID/YYYYYY/

7 -   FILE    FC/XX/,PPS/ASCII/

8 -   FILE    FC/XX/,TPMARK/NO/
```

Format 2 - Assign this file code to the pool denoted by PLID.

Format 3 - Assign all UFF files (except I-D-S/II files) to the pool denoted by PLID.

Format 4 - Assign all I-D-S/II files for the same pool.

Format 5 - Assign the number of buffers and the buffer size specified to this file.

Format 6 - Assign this file to the pool denoted by PLID. If the pool does not exist, FIB information is used to build file buffers.

Format 7 - Assign the output to a print-file created in the Page Printing System (PPS) format.

Format 8 - Assign this file to an unlabeled EBCDIC tape with no leading tape mark.

Formats 2, 5, and 6 must have unique file codes. Formats 6 and 7 can be used in addition to format 4. Formats 3 and 4 can be used only once. Format 5 is used to override the number and size of buffers specified in the program.

Space Assignment For Internal Structures

UFAS dynamically acquires space for all necessary internal structures. These internal structures comprise the following tables and buffers: File Control Table (FACT), Device Control Table (DCT), Buffer Control Table (BCT), File Attribute Block (FAB), and label buffers when needed. No direct intervention is required on the part of the user.

When additional memory is required, the unused space in the program area (remaining from the loading process) is used. If the unused space is insufficient, more is requested from the operating system. The user can reduce system overhead by increasing the memory required to run the program with a $ LIMITS card (see Control Cards Reference Manual). The increase can be estimated as the sum of the number of words required for all control intervals of all the files to be opened at the same time.

Execution Time Parameters

The $ DATA .U file can be used with a control card with format 4 to override FIB parameters for control interval size (BFSZ), the number of buffers (NBUF), the load overflow increment (LOVI) and the general overflow increment (GOVI). When attempting to override the FIB parameters for a tape file, the user may request a maximum of two buffers.

Example:

```
1      8
_____________________________________________
FILE   FC/AA/,NBUF/5/,BFSZ/2048/
FILE   FC/BB/,LOVI/10/,GOVI/20/
NOTE:  The first format requires the presence of both the NBUF and BFSZ
       parameters.
```

Defaults

If neither the control interval size nor the number of buffers is specified in the FIB or the $ DATA .U file, UFAS will provide buffer space for each file large enough to contain 4096 words. The specific defaults for control interval sizes and buffers are listed by file format. If the defaults do not meet the file requirements, the user must specify the sizes.

- Control Interval Size Defaults

UFF (mass storage)	512 words
GFRC (mass storage)	320 words
ISP	320 words
UFF (tape)	4096 words
ANS	4096 words
IBM	4096 words
GFRC (tape)	4096 words
H2000	4096 words

- Number of Buffer`Defaults

$$\text{Number of Buffers} = \frac{4096}{\text{Control Interval Size}}$$

- Both CI Size and Number of Buffer Defaults

UFF (mass storage)	8 - 512-word buffers
GFRC and ISP (mass storage)	12 - 320-word buffers
All tape formats	1 - 4096-word buffer

FORMULAS

Control Interval Size

The following formulas are intended to assist the user in determining the control interval size to be supplied to the FIB when records have fixed length. Note that CI and R are expressed in bytes or characters, depending on whether the external code set specified in the FIB is a 6-bit or 8-bit plus parity code set.

CI - Control interval size in bytes or characters

N - Number of fixed-length records per block

R - Fixed-length record size in bytes or characters

UFF Sequential and Relative

CI = N(R+4)+8 (external code set must be 9-bit data only)

UFF INDEXED

CI = N(R+10)+22 (external code set must be 9-bit data)

American National Standard (with BSN)

CI = N(R)+6 (external code set must be 9-bit data only)

American National Standard (without BSN) and IBM

CI = N(R) (external code set must be 9-bit data only)

GFRC

CI = N(R) (external code set may be 6- or 9-bit data)

H2000

CI = N(R) (external code set must be 6-bit data only)

ISP

CI = N(R+12)+6 (external code set must be 6-bit data only)

Index File Size

The file size required for the prime keys and alternate keys is calculated as follows:

1. Prime Key

Independent variables used:

NREC - Number of records in the file
BLOCK - Number of records per control interval (average)
CISZ - Control interval size in words
KEYSZ - Prime key size in words

$$A = \frac{(NREC\ /\ BLOCK)}{(CISZ-3)/(KEYSZ+1)}$$

$$B = \frac{A}{(CISZ-3)/(KEYSZ+1)}$$

$$C = \frac{B}{(CISZ-3)/(KEYSZ+1)}$$

$$D = \frac{C}{(CISZ-3)/(KEYSZ+1)}$$

$$\underset{\text{(Links)}}{SIZE} = (A + B + C + D) \times \frac{CISZ\ \text{(words)}}{3840}$$

2. Alternate Key

Independent variables used in addition to those listed for prime key:

UREC - Number of different key values in alternate key file
DREC - Number of records containing duplicate key values

If all keys are unique,

$$A = \frac{NREC}{(CISZ-5)/(KEYSZ+1)}$$

If duplicate keys are present,

$$A = \frac{UREC}{(CISZ-5)/(KEYSZ+1)} + \frac{DREC}{(CISZ-5)}$$

$$\underset{\text{(Links)}}{SIZE} = (A + B + C + D) \times \frac{CISZ\ \text{(words)}}{3840}$$

NOTES: 1. If additional space is required, UFAS will provide the space via a MME GEMORE.

2. The alternate key calculation must be performed for each alternate key. For Indexed files with alternate keys, the space calculated for prime key must be added to the space calculated for each alternate key.

UFF INDEXED ALTERNATE KEY SORT

UFAS sorts the alternate key values to construct alternate key indexes when the file is built or when the I-D-S/II Key File Utility routine is run (see the Sort/Merge Program manual, order number DD09, for discussion of Sort procedures).

UFAS allocates a 1-link file for Sort I/O. If the user does not allocate at least one collation file (S1), UFAS obtains a 1-link file via a MME GEMORE. If the user has not allocated Sort space, UFAS makes sure that at least 16K of memory is available for the Sort.

The file allocated for Sort I/O is a UFF sequential mass storage file with a CI size of 2048 bytes. File space is enlarged in 1-link increments, as needed. Two buffers are specified.

If the number of alternate key values is large, performance of the alternate key index built may be improved by increasing the number of buffers and/or enlarging the size of the Sort file, and/or increasing memory size for Sort.

If the I-D-S/II Key File Utility routine is not being used, the user can allocate sufficient Sort Collation file space. Memory can be allocated with a $ USE card as follows:

```
1      8      16
$      USE    .SMA/1/,.SMB/n/,.SMC/1/
```

where: n = At least 16K words

UFAS provides an additional means for specifying Sort I/O file size, number of buffers, and Sort memory size at execution.

The user controls the size and growth of the Sort I/O file via either a $ PRMFL or a $ FILE card. UFAS increases file size in increments of 5 percent of the initial file size.

The user-allocated file code, number of buffers, and/or size of Sort memory can be specified to UFAS via a $ DATA .U file control card as follows:

```
1      8
_____________________________________________
KEYSRT IFC/aa/,SFC/bb/,SNBUF/cccc/,SMEM/dd/
```

where:

aa - File code of the indexed index file to which the Sort parameters apply

bb - File code of the user-allocated Sort I/O file (optional).

cccc - Number of buffers to allocate to the Sort I/O file (optional)

dd - Number of 1K-word units of memory to reserve for Sort (optional). This field has no effect if there is user allocated Sort memory space. This option is the only means of reserving memory at program execution time, and must be used if a memory size change is required when executing the I-D-S/II Key File Utility routine.

If more than one index file will be built in an activity, the SMEM parameter (if desired) must be included on the KEYSRT card pertaining to the index file that will be built first.

The parameter FC/**/ can be used with the I-D-S/II Key File Utility routine, and with any user program in which there is only one alternate key indexed file open at a time. The specified Sort parameters then apply to each alternate key file sort. The user allocates only one Sort input and output file, which is re-used as each alternate key indexed file is opened and closed.

APPENDIX B

FILE INFORMATION BLOCK

The file information block (FIB) provides the user with a means of communicating the specific characteristics of a file to the Unified File Access System. All file information between the user and UFAS is channeled through the file information block. For each file of a user program, a file information block must be coded in the user program to define the characteristics of the file and the required processing options. The file information block may be created by the General Loader at load time if so requested via a control card.

In this system a file information block is composed of 23 words (0 through 22) and each word has one or more assigned fields, delimited by their respective bits, for specific designations (see Figure B-1). If a field is not applicable to a particular situation, it can be either ignored or assigned a value of zero. Fields that are set in a conflicting manner cause an exception condition.

Numeric literal values in the FIB are binary, unless otherwise indicated. All pointers are set within bits 0-17 of the reserved word or bits 0-23 for words 7 through 10. Values assigned to characters, such as number of characters or character offset, are based on the internal code set (bits 18-35 of word 3).

Word Number	Bits 0 2 5 8 11 14 17 20 23 26 29 32 35								
0	Number of Buffers Requested	File Format	File Code						
+1	Record size	Control Interval Size							
+2	Organ-ization	AM[1]	Media Code	a	b	c	d e f g h i j k l m n o p (see Fields in Table B-1)	q	r s
+3	External Code Set	Internal Code Set							
+4	Number of Record Types	File Sequence Number	Checkpoint Count Maximum Control Interval Size for all Areas						
+5	Percent Fill	Index-File File Code							
+6	Key Pointer Descriptor								
	Relative Key Value Pointer	Relative Key Type							
+7	Process Area Pointer								
	Word Pointer	CN[2]	MBZ[3]	Reserved					
+8	File Status Code Pointer								
	Word Pointer	CN[2]	MBZ[3]	Reserved					
+9	File Name Pointer								
	Word Pointer	CN[2]	MBZ[3]	Reserved					
+10	Retention Period Descriptor								
	Word Pointer	CN[2]	TN[4]	S[5]	SF[6]	N[7]			

[1] Access Mode

[2] Character number indicating starting position within word.

[3] Must be zero

[4] Code defining type of numeric characters

[5] Sign and decimal type

[6] Scaling factor

[7] Number of characters or bits in data string

For detailed definition of symbols see Extended Instruction Set in Macro Assembler Program manual.

Figure B-1. File Information Block Format

Word Number	Bits 0 2 5 8 11 14 17	20 23 26 29 32 35
+11	File Name Error Procedure Pointer	
+12	Pointer to Processing Mode Error Procedure Table	
+13	Input Transliteration Table Pointer	
+14	Output Transliteration Table Pointer	
+15	File Code List Pointer	
+16	Label Exit Table Pointer	
+17	Record Size Word Pointer	
+18	Storage for File Access Control Table (FACT) Address	
+19	Reserved	
+20	User.Label Process area (ANS,UFF,IBM Files Only)	
	Word Pointer	CN[2] | MBZ[3] | Reserved
	Format Type	Blocking Factor
+21	Banner Character	
+22	Pad Character	

NOTE: Words 20, 21, and 22 are used with H2000. Word 20 is also used with UFF., ANS and files with User Label Processing.

Figure B-1 (cont). File Information Block Format

Table B-1. File Information Block Map

Word Number	Field Symbol	Bits	Contents
0	DFRBUF	0-11	Number of buffers requested (binary count). Default for tape files: ● DFRBUF = 1 Default for mass storage files: ● If DFPSZE = 0 DFRBUF = 8 for UFF files DFRBUF = 12 for ISP and GFRC files ● If DFPSZE ≠ 0 DFRBUF = $\frac{N}{DFPSZE}$ where: N = 16128 9-bit characters (4032 words) for UFF files = 24570 6-bit characters (4095 words) for ISP files
	DFIFF	12-17	File format indicator 0 Non-standard (including unlabeled) 1 GFRC format 2 American National Standard format 3 IBM format 4 UFF format (default value) 5 H2000 format 6 ISP coexistence format 7 I-D-S/I coexistence format
	DFFCDE	18-35	File code (two ASCII bytes)
1	DFRSZE	0-17	Maximum record size (binary count of number of characters). This is the actual record size for the fixed-length record or maximum record size for variable-length records.
	DFPSZE	18-35	Control interval size (binary count of the number of characters). Default for tape files: ● 24570 6-bit characters (4095 words) for GFRC and H2000 files ● 16380 9-bit characters (4095 words) for ANSI and IBM files Default for mass storage files: ● 2048 9-bit characters (512 words) for UFF files ● 1920 6-bit characters (320 words) for ISP files

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
2	DFORG	0-5	File organization 0 Sequential (default) 1 Relative 2 Indexed 3 Integrated
	DFACCE	6-8	File access mode 0 Sequential (default, except for ISP) 1 Random 2 Dynamic (default only for ISP)
	DFMCDE	9-14	File media code in octal (GFRC files only) 0 Media code does not apply 1 Binary card image 2 BCD card image 3 BCD print line image 6 GFRC format ASCII 7 ASCII print line image 10 Time sharing system information record 11 BCD print line image 12 ASCII card image 15 ASCII print line image NOTE: If media code is zero and the file is assigned to SYSOUT, the media code is set to 3 or 7 depending on the external code set. If media code is not zero, a process area is required (bit 23 of word 2) and GFRC format is assumed (see bits 16, 17, and 30). Media codes 11 and 15 indicate that a user defined report code was inserted in the first two-character positions of the logical record. Appropriate ignore characters are entered in these character positions before output is released to a printer.
	DFSZRI	15 (Field a)	Record size error indicator 0 Error condition is ignored. 1 UFAS returns to user's error procedure or aborts if no error procedure is specified (see substatus code 95).
	DFBSNI	16 (Field b)	Block serial number indicator (GFRC and American National Standard Files only) 0 Block serial numbers used (required if media not zero) 1 Not used
	DFFVLR	17 (Field c)	FLR/VLR indicator (sequential files only, except H2000) 0 Variable-length records (VLR) (required if media code not zero) 1 Fixed-length records

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
	DFOPTF	18 (Field d)	Optional file indicator (sequential single files opened for input only) 0 Not an optional file 1 Optional file
	DFNSEQ	18	Sequence check bypass during Indexed file build 0 Perform sequence check 1 Bypass sequence check
	DFMULT	19 (Field e)	Multifile indicator (applies only to sequential tape files, but not to H2000 or to extend processing mode) 0 Not multifile 1 Multifile
	DFSUBS	19	Indexed file subfile indicator 0 Subfiles do not exist 1 Subfiles exist
	DFCCOM	20	DUAL Indexed file Commercial Collate key sequence indicator 0 No Commercial Collate sequence 1 Commercial Collate sequence
	DFBNRI	21 (Field g)	Bannered file indicator (H2000 files only) 0 File is not bannered 1 File is bannered by word 21
	DFTTI	22 (Field h)	Transliteration table indicator (does not apply to UFF indexed and ISP files) 0 No transliteration is required 1 Apply transliteration tables supplied by words 13 and 14
	DFPAI	23 (Field i)	Process area indicator (optional only for sequential files) 0 Process area supplied by word 7 1 Process area is not supplied (in-place processing is required and is supplied by word 7)
	DFFSCI	24 (Field j)	File status code indicator 0 File status code field is not supplied 1 File status code field is supplied by word 8
	DFFEPI	25 (Field k)	File name error procedure indicator 0 Error procedure is not supplied 1 Error procedure is supplied by word ll

DFPEPI	26 (Field l)	Processing mode error procedure indicator 0 Error procedure table is not supplied. 1 Error procedure table is supplied by word 12.
DFULEI	27 (Field m)	User label exit table indicator (applies only to GFRC) 0 User label exit table is not supplied. 1 User label exit table is supplied by word 16.

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
	DFFCLI	28 (Field n)	File code list indicator (applies only to UFF and ISP) 0 File code list is not supplied. 1 File code list is supplied by word 15.
	DFRSWI	29 (Field o)	Record size word pointer indicator 0 Pointer not supplied. 1 Pointer supplied by word 17.
	DFPRI	30 (Field p)	Partitioned record indicator (not applicable to H2000 files) 0 File does not contain partitioned records. (required if media code is not zero). 1 File contains partitioned records.
	DFLBLI	31 (Field p)	Label indicator (does not apply to H2000, UFF, ANSI, and ISP files) 0 File contains labels. 1 File does not contain labels.
	DFLBLF	31-32	Label type indicator (H2000 files only) 0 80-character label (default) 1 120-character label 2 Unlabeled
	DFMODE	32 (Field r)	Recording mode (GFRC and DUAL Indexed files) 0 Binary for GFRC, ASCII for DUAL Indexed (odd parity) 1 BCD (odd parity, except for 7-track tapes)
	DFPIC	33-35 (Field s)	Processor Identifier -- indicates what processor is making the current access. 0 Native 1 OM-IV 2 XITR
3	DFETCD	0-17	External code set 0 ASCII (default) 1 EBCDIC 2 GBCD (GFRC BCD) 3 HBCD (Honeywell BCD) 4 JIS (Japanese Industrial Set) 5 36-bit binary (for USS Sequential only)
	DFITCD	18-35	Internal code set 0 ASCII (default) 1 EBCDIC 2 GBCD 3 HBCD 4 JIS 5 36-bit binary (for UFF Sequential only)

NOTE: These fields define the character size of the data as it exists on the external media and as it will exist in memory.

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
4		0-8	Reserved
	DFFSEQ	9-17	File sequence number is a binary count of files preceding the desired file (sequential files only, except H2000). If zero, multifile bit (word 2, bit 19) is set, and no rewind was specified at open time, the call is positioned at the beginning of the next file. (Default is zero.) File positioning indicator (H2000 files only) 0 Tape is left in current position (default) 1 For positioning to beginning of the first reel If open with no rewind is specified for an input tape and the file sequence number equals zero, it is assumed that the tape is positioned just in front of a valid H2000 header label. If an end-of-file condition is encountered on input processing, the tape is positioned as follows: • Odd parity labels 1EOF Tape position 1ERI or next 1HDR • Even parity labels 1EOF TM Tape position 1ERI or next HDR If succeeded by a close with no rewind, the tape is positioned in front of the next header label if one exists. If close with no rewind is invoked before reaching an end-of-file condition, the current tape position will be undefined.
	DFCHEK	18-35	Incremental record count (binary) in which checkpoint dumps are taken. UFAS maintains a count of records accessed (read and written). Whenever the count equals the number specified in this field, a checkpoint dump is taken on (C.) file code. If this field is zero, no checkpoints are taken.
5	DFPFIL	0-17	Control interval percent fill is number of characters of each control interval that is to be filled with data at time file is built (indexed files opened for output only).
	DFIFCD	18-35	File code of the data file index is two ASCII bytes. (Applies only to UFF indexed and ISP coexistence files.)

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
6			Relative files
		0-17	For bits 18-35=0
			Pointer to 36-bit binary relative key value.
	DFKDES		For bits 18-35≠0
			Pointer to an Extended Instruction Set (EIS) descriptor.
		18-35	Internal relative key type
			0 36-bit binary 1 Numeric ASCII field (not floating point) 2 GFRC numeric packed decimal field (not floating point) 3 16-bit binary 4 32-bit binary
			NOTE: If user provides a pointer to a 36-bit binary relative key value in word 6, UFAS uses this value. However, if the type field indicates a different value for the key, bits 0-17 of word 6 must point to one of the following EIS descriptors:
			1 9-bit numeric (not floating point) 2 4-bit numeric (not floating point) 3 9-bit alphanumeric indicating two bytes 4 9-bit alphanumeric indicating four bytes
			This descriptor is used as part of a code group that converts the key into a 36-bit binary key.
			ISP files
		0-17	Word pointer
	DFKDES		Pointer to the prime key description list.
		18-35	Zero
			Key description list format:
			• Byte offset within record of beginning of key field for bits 0-17.
			• Byte length of key for bits 18-35.

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
			<u>Indexed files</u>
	DFKDES	0-17	Word pointer
			Pointer is set to a key description list.
	DFGNIN	18-26	FIB macro name - GOVINC - whose value ranges from 0 to 510; ratio of data control intervals to general overflow control intervals, allocated during build mode.
	DFLCIN	27-35	FIB macro name - LOVINC - whose value ranges from 0 to 510; ratio of data control intervals to local overflow control intervals allocated during build mode.
			Key description list format:
			● If bit 0 equals 1, duplicate keys are allowed (alternate keys only).
			● Byte offset within record of beginning of key field for bits 1-17.
			● Byte length of key for bits 18-35.
			NOTE: Each key must have one entry. A word of zeros terminates the list. The first entry represents the prime key. The second entry represents the first alternate key (additional entries represent other alternate keys).
7			Process area
		0-17	Word pointer
	DFPRSS	18-20	Starting byte position within word
		21-35	Must be zero
			NOTE: This word contains the pointer to the process area if bit 23 of word 2 is zero. If bit 23 of word 2 is one, this word contains the pointer to the record area.
8			File status code
	DFSTAT	0-17	Word pointer
		18-20	Starting byte position within word
		21-23	Must be zero
		24-35	Reserved
			See Exception Conditions
			This field size is always two ASCII bytes.

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
9	DFFILN		File name (does not apply to ISP files)
		0-17	Word pointer
		18-20	Starting byte position within word
		21-23	Must be zero
		24-35	Reserved
			This field size for file name (in ASCII) is as follows: ● Extended GFRC - 12 bytes ● American National Standard and UFF - 17 bytes ● IBM - 17 bytes ● H2000 - 10 bytes
			NOTE: IBM file names in ASCII will be converted to EBCDIC by UFAS.
10	DFRETP		Retention period descriptor (does not apply to ISP files)
		0-17	Word pointer
		18-20	Starting byte position within word
		21	Code that defines type of numeric characters specified: 0 9-bit 1 4-bit
		22-23	Must be greater than zero
		24-29	Must be zero
		30-35	Must be less than or equal to 3
			NOTE: No retention is assumed if this field is equal to zero. Permanent retention is assumed if the field represents the largest possible value, that is, 999.
11	DFUERR	0-17	Pointer is set to user's error processing procedure for file name.
		18-35	Reserved

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
12	DFPERR	0-17	Pointer is set to user's error processing procedure table according to processing mode as follows: Word 0 - Pointer set for output mode 1 - Pointer set for input mode 2 - Pointer set for input/output mode 3 - Pointer set for extend mode These pointers are contained in bits 0-17 of each word.
		18-35	Reserved
13	DFRTBL	0-17	Sequential files - pointer set to input transliteration table
		18-35	Reserved
14	DFWTBL	0-17	Sequential files - pointer set to the output transliteration table
		18-35	Reserved
15	DFFCPT	0-35	Pointer is set to a list of file codes (for mass storage files only). Each file code in the list represents one unit of a multi-volume file and causes device verification to take place. The list must not contain the primary (first volume) file code given in word 0. All volumes must be on the same type of device and online. The format of the list is as follows: Word 0 0-35 - Number of entries in binary equals n. 1-n 0-35 - Device code is two ASCII bytes, right justified. NOTE: Maximum number of additional data file codes is seven for ISP files.
16	DFLEXT	0-17	Label exit table pointer (applies to GFRC files only). Pointer is set to the user's label exit table. The table is composed of user exit addresses in bits 0-17. These addresses for GFRC processing are: ● Pre-header ● Post-header ● Pre-trailer ● Post-trailer
		18-35	Reserved

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
17	DFRSW	0-17	Pointer to record size word For Get Next and Get macros, UFAS returns the smallest record size (actual record size or FIB record size to the user in binary). For a Put macro or rewrite function, the user may supply the record size to override the FIB (maximum) record size if VLR or the VIB does not contain the record size (VIB does not apply to ISP files).
		18-35	Reserved
18	DFPFCT	0-17	This word is to be used by UFAS.
		18-35	Reserved
19	DFRCW	0-35	Reserved
20	DFTYPE	0-17	File format type code in octal, right justified (H2000 files only) 0 1.AF - Unblocked fixed-length records 1 1.AV - Unblocked variable-length records 2 1.B - Blocked fixed-length records 40 4.1F - Unblocked fixed-length records 41 4.1V - Unblocked variable-length records 42 4.2 - Blocked fixed-length records
	DFBLK	18-35	Blocking factor Format types 1.B and 4.2 indicate the number of records per block (default 191).
			User label process area (ANS,UFF, and IBM only)
	DFULAR	0-17	Word pointer.
		18-20	Starting byte position within word.
		21-35	Must be zero.
21			Banner character (H2000 files only)
		0-5	Input If a bannered file (bit 21 of word 2) is specified and the user exercises the option of selecting a one banner-character, only those records in a block with the chosen banner are released to the user. If the user specified a bannered file but does not select a banner character by setting this field to zero, UFAS assumes that all records associated with a banner character in the 40-57 octal range are legitimate data records.

Table B-1 (cont). File Information Block Map

Word Number	Field Symbol	Bits	Contents
	DFBANR		Output
			The user may elect to have UFAS build bannered files with one 6-bit banner character at the physical beginning of each tape block.
			If this option is selected, the user may provide a one banner-character, which is placed in each tape block, except blocks containing label records that always have the banner character 1. The banner character provided must be in the 40-57 octal range.
		6-35	Reserved
22			Pad character (H2000 files only)
		0-5	Input
			If the user selects to have UFAS single out and remove padding records, an explicit pad character must be provided, which can be any 6-bit configuration except all zeros. Any record containing the pad character in all characters of the record is considered a padding record.
			If this option is not elected, the record is considered a standard data record and is released to the user. (Note that the preceding action as specified for bannered files takes place before this test for padding records.) If a pad character is not provided, no check is made for padding records.
	DFPAD		Output
			The user cannot command UFAS to write a padding record. Such action takes place automatically when it is necessary to complete an exact number of records per block.
			The user may specify, however, a pad character which will be used in each padding record. If the user does not specify a pad character by setting this field to zero, a default value is selected as follows:
			● 77 (octal) - odd parity data ● 11 (octal) - even parity data
		6-35	Reserved

APPENDIX C

PROGRAMMING EXAMPLES

This appendix contains examples of source programs written in COBOL-74 and GMAP. The examples contain within the programs a description of the purpose of each program.

Figures C-1 and C-2 are examples of a deck setup and a COBOL-74 program to build a GFRC sequential file. Figures C-3 through C-6 contain examples of GMAP programs to build GFRC and UFF relative files, to process an index file, and to access ISP files. Figure C-7 is a sort example of a UFF sequential file.

```
$ SNUMB
$ IDENT
$ USERID
$ OPTION
$ CBL74
$ PRMFL
$ FILE
$ FILE
$ LIMITS

  COBOL-74 Program

$ EXECUTE
$ SYSOUT
$ TAPE
$ DATA
$ ENDJOB
***EOF
```

Figure C-1. GFRC Sequential File Deck Setup for COBOL-74 Program

```
1     8        16
IDENTIFICATION DIVISION.
PROGRAM-ID.  MISC09.
AUTHOR.         HONEYWELL.
DATE-WRITTEN.  02/04/72.
DATE-COMPILED.   04/11/75.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  HIS-SERIES-60.
OBJECT-COMPUTER.  HIS-SERIES-60.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN CR-CARD-READER
    ORGANIZATION GFRC SEQUENTIAL SSF.
    SELECT VLR-FILE ASSIGN VR
    ORGANIZATION GFRC SEQUENTIAL.
    SELECT POUT-FILE ASSIGN LP-PRINTER
    ORGANIZATION GFRC SEQUENTIAL SSF.
I-O-CONTROL.
DATA DIVISION.
FILE SECTION.
FD  CARD-FILE  LABEL RECORD STANDARD CODE-SET IS GBCD.
01  CR-REC.
    02  CNTIN     PIC X.
    02  CR-DATA   PIC X(20).
    02  FILLER   PIC XXX.
    02  FILLER   PIC  X(56).
FD VLR-FILE
                         BLOCK CONTAINS 816 CHARACTERS
                          LABEL RECORDS ARE STANDARD
                VALUE OF FILE-ID IS "VLR-FILE"
                               DATA RECORD IS VLR-REC.
01  VLR-REC.
    02  RLW-1  PIC  9(6).
    02  VLR-DATA  PIC X(200).
FD  POUT-FILE  LABEL  RECORD  STANDARD CODE-SET IS GBCD
                        DATA RECORD IS LP-REC.
01  LP-REC           PIC X(120).
WORKING-STORAGE SECTION.
77  P-CTR            PIC 99      VALUE 0  USAGE IS COMPUTATIONAL.
77  L-CTR            PIC 99      VALUE 60 USAGE IS COMP.
77  RLW-2  PIC  9(6)  VALUE 1.
77  RLW-3  PIC  9(6)  VALUE  1.
77  RLW-4  PIC  9(6)  VALUE  10.
77  RLW-5  PIC  9(6).
77  W-CNTIN          PIC X.
77  D-CTR            PIC 99.
77  WBLC-CTR         PIC 99      VALUE 1 COMPUTATIONAL.
77  C2-C1  USAGE COMP-1.
77  C1-D             PIC 9999.
77  DT-CTR           PIC 99   COMP.
```

Figure C-2. COBOL-74 Program to Build a GFRC Sequential File

```
1      8       16
01  VLR-RSV.
    02  RLW-6  PIC  9(6)  VALUE  ZERO.
    02  VLR-TEL    PIC X(20) USAGE IS DISPLAY
                  OCCURS 10 TIMES INDEXED BY VTX.
01  LABEL-AREA.
    02  BTL-ID     PIC X(8)  VALUE "VLR-FILE".
    02  YDDD       PIC X(4)  USAGE IS DISPLAY.
    02  ONNN       PIC X(4)  VALUE "0001".
01  TIME-AREA.
    02  FILLER     PIC X(3)  USAGE IS DISPLAY.
    02  HH         PIC X(2)  USAGE IS DISPLAY.
    02  MM         PIC X(2)  USAGE IS DISPLAY.
    02  TT         PIC X     USAGE IS DISPLAY.
01  HEAD-1.
    02  FILLER     PIC X(30) VALUE SPACE.
    02  FILLER     PIC X(53) VALUE "***  VLR FILE AND USAGE PHRA
    "SE TEST   ***      TIME ".
    02  L-HH       PIC XX.
    02  FILLER     PIC X     VALUE ":".
    02  L-MM       PIC XX.
    02  FILLER     PIC X     VALUE ":".
    02  L-TT       PIC X.
    02  FILLER     PIC X(9)  VALUE "    PAGE ".
    02  L-PAGE     PIC ZZ9.
01  HEAD-2.
    02  FILLER     08C X(68) VALUE "            CONTINUATION
    "      DATA                        COUNTER"
01  DTL-1.
    02  FILLER     PIC X(25) VALUE SPACE.
    02  L-CNT      PIC X.
    02  FILLER     PIC X(9)  VALUE SPACE.
    02  L-DATA     PIC X(20).
    02  FILLER     PIC X(11) VALUE SPACE.
    02  LD-CTR     PIC Z9.
01  DTL-2.
    02  FILLER     PIC X(8)  VALUE "    NO. ".
    02  BLC-CTR    PIC 99.
    02  FILLER     PIC X(15) VALUE "    RLW-SIZE = ".
    02  RLW-SZ     PIC Z9.
    02  FILLER     PIC X(23) VALUE "      DATA AREA SIZE = ".
    02  DT-SZ      PIC ZZ9.
    02  FILLER     PIC X(15) VALUE " CHARACTERS.   ".
01  DTL-3.
    02  FILLER     PIC X(22) VALUE "                  DATA".
    02  DT-NO      PIC Z9.
    02  FILLER     PIC X(9)  VALUE SPACE.
    02  DT-LP      PIC X(20).
    02  FILLER     PIC X(15) VALUE SPACE.
PROCEDURE DIVISION.
PROC SECTION.
```

Figure C-2 (cont). COBOL-74 Program to Build a GFRC Sequential File

```
1      8       16
START1.
    MOVE "2050" TO YDDD.
    OPEN INPUT CARD-FILE
         OUTPUT VLR-FILE, POUT-FILE.
    READ CARD-FILE AT END GO TO NEXT-SENTENCE.
NEXT-SENTENCE.
    MOVE CNTIN TO W-CNTIN MOVE CR-DATA TO VLR-TBL (1).
    MOVE 1 TO D-CTR   MOVE 1 TO RLW-6 SET VTX TO 2.
    MOVE CNTIN TO L-CNT MOVE CR-DATA TO L-DATA.
    MOVE D-CTR TO LD-CTR PERFORM POUT-RTN.
READ-CR.
    READ CARD-FILE AT END GO TO C-EOF.
    IF W-CNTIN IS NOT EQUAL TO CNTIN GO TO VLR-OUT.
    IF D-CTR IS GREATER THAN 9 GO TO VLR-OUT.
PO. MOVE CR-DATA TO VLR-TBL (VTX) SET VTX UP BY 1.
    ADD 1 TO D-CTR.
    MOVE CNTIN TO L-CNT MOVE CR-DATA TO L-DATA.
    MOVE D-CTR TO LD-CTR PERFORM POUT-RTN.
    GO TO READ-CR.
VLR-OUT.
    COMPUTE D-CTR = D-CTR * 5 + 1.
    MOVE D-CTR TO C2-C1.
    MOVE C2-C1 TO RLW-6.
                  1
** 1 5-150 SIGN OF SENDING ITEM WILL NOT BE MOVED TO THIS ITEM

    WRITE VLR-REC FROM VLR-RSV
    MOVE 0 TO D-CTR    SET VTX TO 1.
    MOVE 0 TO RLW-6.   MOVE CNTIN TO W-CNTIN.
    GO TO PO.
POUT-RTN SECTION.
NEX.
    IF L-CTR IS LESS THAN 60 GO TO P-GO.
    MOVE "00000000" TO TIME-AREA.
    MOVE HH TO L-HH    MOVE MM TO L-MM
    MOVE TT TO L-TT.   ADD 1 TO P-CTR.
    MOVE P-CTR TO _-PAGE.
    WRITE LP-REC FROM HEAD-1 AFTER PAGE.
    MOVE SPACE TO .P-REC.
    WRITE LP-REC FROM HEAD-2 AFTER 2.
    MOVE 4 TO L-CTR.
P-GO.
    WRITE LP-REC FROM DTL-1 AFTER 1.
    ADD 1 TO L-CTR.
P.EXIT.
    EXIT.
EOF SECTION.
C.EOF.
    IF RLW-6 IS EQUAL TO 0 GO TO C-CLOSE.
    WRITE VLR-REC FROM VLR-RSV.
```

Figure C-2 (cont). COBOL-74 Program to Build a GFRC Sequential File

```
1      8       16
C-CLOSE.
    CLOSE CARD-FILE VLR-FILE.
    OPEN INPUT VLR-FILE.
    MOVE 60 TO L-CTR.
READ-VLR.
    READ VLR-FILE AT END GO TO VLR-EOF.
    MOVE VLR-REC TO VLR-RSV.
    IF RLW-1 IS EQUAL TO RLW-6 NEXT SENTENCE
    ELSE MOVE "IF RLW = RLW RESULT IS INCORRECT" TO LP-REC
    WRITE LP-REC BEFORE 2.
    MOVE WBLC-CTR TO BLC-CTR ADD 1 TO WBLC-CTR.
    MOVE RLW-6 TO RLW-5   MOVE RLW-5 TO C2-C1.
                                        1
** 1 5-156 POSSIBLE LEFT TRUNCATION

    MOVE C2-C1 TO C1-D    MOVE C1-D TO RLW-SZ.
                  1                    2
** 1-5-156 POSSIBLE LEFT TRUNCATION
** 1 5-150 SIGN OF SENDING ITEM WILL NOT BE MOVED TO HIS ITEM
** 2 5-156 POSSIBLE LEFT TRUNCATION

    COMPUTE C1-D = ( C1-D - 1 ) / 5.
    WRITE LP-REC FROM C1-D BEFORE 2.
    MULTIPLY C1-D BY 20 GIVING DT-SZ.
    MOVE DTL-2 TO DTL-1 PERFORM POUT-RTN.
    MOVE 1 TO DT-CTR  SET VTX TO 1.
RO. MOVE DT-CTR TO DT-NO  MOVE VLR-TBL (VTX) TO DT-LP.
    MOVE DTL-3 TO DTL-1 PERFORM POUT-RTN.
    IF VTX = C1-D 30 TO READ-VLR.
    ADD 1 TO DT-CTR  SET VTX UP BY 1 GO TO RO.
VLR-EOF.
    CLOSE VLR-FILE POUT-FILE.
    STOP RUN.
END COBOL.
```

Figure C-2 (cont). COBOL-74 Program to Build a GFRC Sequential File

```
1      8       16
$      IDENT   ACCOUNT #, NAME
$      GMAP    NDECK
$      LIMITS  ,28K
$      SELECT  UFAS/IMCV/MACRO
       LBL     DSRO,H6000H5.000
       TTL     H6000H5.000     6000 SERIAL VLR FILE BUILD     760130DSRO
       CPR     1976
       TTLDAT
       PMC     ON
       EDITP   ON
       LODM    .DMAC
       TTLS    6000 SERIAL VARIABLE-LENGTH RECORD FILE BUILD PROGRAM
*
*              PURPOSE=      TO BUILD A 6000 SERIAL VARIABLE-LENGTH RECORD
*                            FILE, THE RECORDS OF WHICH CONTAIN THEIR
*                            ORDINAL NUMBERS IN THE FIRST FOUR CHARACTERS

*              CALLS=        FIBMAC    (DECLARATIVE MACRO)
*                            .DOPEN
*                            .DPUT
*                            .DCLOS
*
*              CALLED BY=    ACTIVITY INITIATION ROUTINE
*
*
*
*
*
*
       EJECT
       SYMDEF  .DDSRO,.DBLD

       SYMREF  .DVLPO,.DPRNT,U.ATOG

*
*
*      DECLARE FILES
*
*
FIB    FIBMAC  FC,GFRC,[NBUF,2,RSZ,100,CISZ,1280,PROAR,WAREA1,0,
       ETC     FSCODE,STAT1,0,IFLR,0]
*
FIBOUT FIBMAC  P*,GFRC,[NBUF,2,RSZ,102,CISZ,320,PROAR,WAREA2,0,
       ETC     FSCODE,STAT2,0,IFLR,0,MEDCOD,3,EXTCOD,GBCD,OUTTRN,U.ATOG]
```

Figure C-3. GMAP Program to Build a GFRC File

```
1      8       16
________________________________________________________________
*
*
.DDSRO NULL
.DBLD  NULL
*
*
*
*      INITIALIZE SYSOUT SWITCH
*
       STZ     SWITCH,$
*
*      OPEN FILES
*
       DOPEN   FIBLS,LIST,(FIBA,FIB,SYM,.DVLPO,MODE,OT)
*
*      CHECK DOPEN STATUS CODE
*
       LDX     .XR1,STAT1,$
       CMPX    .XR1,GOOD,$
       TNZ     ABORT4,$

       DOPEN   FIBLS,LOUT,(FIBA,FIBOUT,SYM,.DPRNT,MODE,OT)

       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABORT5,$

*
*
*      INITIALIZE BINARY RECORD COUNT
*
       STZ     BRCCNT,$
*
BACK   NULL
*
*      ADD 1 TO BINARY RECORD COUNT
*
       AOS     BRCCNT,$
*
*      CALL ROUTINE TO CONVERT RECORD COUNT FROM BINARY INTO
*      ASCII CHARACTERS
*
       TSX     .XR3,.DTCNV
*
*      PLUG ASCII RECORD COUNT INTO RECORD AREA
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8       16
*
       LDA     ARCCNT,$
       STA     WAREA1,$
*
       CMPA    =0040061060060          ARCCNT=100? (KLUDGY SYSTEM!!!!)
       TZE     EOF,$
*
*
*      PUT RECORD INTO FILE
*
       DLPUT FIB,VIB,EOF,(RSZ,100)
*
*      CHECK STATUS CODE
*

       LDX     .XR1,STAT1,$
       CMPX    .XR1,GOOD,$
       TNZ     ABORT1,$
*
*      WRITE MESSAGE ONTO SYSOUT
*
       LDQ     SWITCH,$
       TZE     BACK,$
*
       DLPUT FIBOUT,VIBOUT,ABORT2,(RSZ,102,RPCOD,63,SLCOD,0)
*
*      CHECK STATUS CODE
*
       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABORT3,$
       TRA     BACK,$
*
EOF    NULL
*
*      NORMAL TERMINATION UPON END OF FILE
*
*
*      CLOSES FILES
*
       DCLOS   FIBLS,LIST,(FIBA,FIB,REW,RWFL)
*
*      CHECK DCLOS STATUS CODE
*
       LDX     .XR1,STAT1,$
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8       16
       CMPX    .XR1,GOOD,$
       TNZ     ABORT6,$
*
*
       DCLOS   FIBLS,LOUT,(FIBA,FIBOUT,REW,NRFL)
*
*      CHECK DCLOS STATUS CODE
*
       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABORT7,$
*
       LDQ     =3H00K,DL
       MME     GEFINI
*
*
ABORT1 NULL
*
*      BAD PUT ON THE WORKING FILE
*
       LDQ     =3H0A1,DL
       MME     GEBORT
*
ABORT2 NULL
*
*      ALTERNATE RETURN TAKEN UPON PUT ON SYSOUT
*
       LDQ     =3H0A2,DL
       MME     GEBORT
*
ABORT3 NULL
*
*      BAD PUT ON SYSOUT
*
       LDQ     =3H0A3,DL
       MME     GEBORT
*
ABORT4 NULL
*
*      BAD OPEN ON THE WORKING FILE
*
       LDQ     =3H0A4,DL
       MME     GEBORT
*
ABORT5 NULL
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8       16
*
*      BAD OPEN ON SYSOUT
*
       LDQ     =3H0A5,DL
       MME     GEBORT
*
ABORT6 NULL
*
*      BAD CLOSE ON THE WORKING FILE
*
       LDQ     =3H0A6,DL
       MME     GEBORT
*
ABORT7 NULL
*
*      BAD CLOSE ON SYSOUT
*
       LDQ     =3H0A7,DL
       MME     GEBORT
*
*
*      DECLARATION OF LOCAL VARIABLES
*
SWITCH BSS     1
BRCCNT BSS     1
ARCCNT BSS     1
WAREA1 NULL
WAREA2 ASCII   10,
       ASCII   10,
       ASCII   10,
LIST   BSS     3
LOUT   BSS     3
VIB    BSS     2
VIBOUT BSS     2
STAT1  BSS     1
STAT2  BSS     1
*
GOOD   ASCII   1,0000
*
*
*      DECLARATION OF LOCAL CONSTANTS
*
.XR1   EQU     1
.XR2   EQU     2
.XR3   EQU     3
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8       16
*


.DTCNV NULL
*
*      ROUTINE TO CONVERT A BINARY NUMBER STORED IN CELL BRCCNT
*      INTO A 4-DIGIT ASCII CHARACTERS STORED INTO ARCCNT
*
*      CLEAR OUTPUT CELL ARCCNT
*
       STZ     ARCCNT,$                CLEAR ARCCNT
*
*      INITIALIZE DIGIT COUNT TO 1
*
       EAX     .XR2,1
*
*      LOAD INPUT INTO Q REGISTER
*
       LDQ     BRCCNT,$
*
*      BINARY-TO-ASCII CONVERSION
*
BCK    NULL
*
       DIV     TEN,$                   Q = Q/10; A = REMAINDER(Q/10)
       EAX     .XR1,0,AL
       TNZ     NXT,$
       QLR     0
       TNZ     NXT,$
       EAX     .XR1,10               IF Q=0 AND A=D THEN STORE BLANK
NXT    LDA     DIGITS,.XR1           LOAD DIGIT WORD
       ANA     =0777000,DU             ISOLATE DIGIT CHARACTER
       ORA     ARCCNT,$                STORE DIGIT CHARACTER
       CMPX    .XR2,4,DU             IS IT THE END?
       TZE     OUT,$                   YES.
       ARL     9                       PREPARE SPACE FOR NEXT DIGIT
       STA     ARCCNT,$
       ADLX    .XR2,1,DU
       TRA     BCK,$


OUT    NULL
*
*      NORMAL TERMINATION
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8      16
*
       STA    ARCCNT,$
       TRA    0,.XR3                   RETURN TO CALLER
*
*      DECLARATION OF LOCAL VARIABLES
*
DIGITS ASCII  1,0
       ASCII  1,1
       ASCII  1,2
       ASCII  1,3
       ASCII  1,4
       ASCII  1,5
       ASCII  1,6
       ASCII  1,7
       ASCII  1,8
       ASCII  1,9
       ASCII  1,
*
TEN    DEC    10

        END
```

Figure C-3 (cont). GMAP Program to Build a GFRC File

```
1      8       16
$      COMMENT REWRITE EACH RECORD OF AN ISP FILE
$      COMMENT
$      COMMENT
$      GMAP    NDECK
$      LIMITS  ,32K
       LODM    .DMAC            GET THE UFAS MACROS
       SYMREF  .DISPS           ISP DIRECTORY FOR SEQUENTIAL ONLY
       SYMDEF  START
START  NULL
OPN    DOPEN   FIBLS,AA,(FIBA,BB,SYM,.DISPS,MODE,IO)
GXT    NULL
       DLGXT   BB,,EOF     GET A RECORD
*
*              MODIFY THE RECORD
*
RWR    DLRWR   BB,,IVK     REWRITE THE RECORD
       TRA     GXT
IVK    NULL
*      KEY IN PROCESS AREA DOESNT MATCH THE CURRENT RECORD
       LDQ     =3HONG,DL
       MME     GEBORT
EOF    NULL
CLS    DCLOS   FIBLS,EE,(FIBA,BB)
       MME     GEFINI
AA     BSS     3
BB     FIBMAC  DD,ISP,[PROAR,ADDR,0,NBUF,4,
*      RECORD AND CI (PAGE)SIZE TAKEN FROM UTILIZATION RECORD
       ETC     FCLST,FCS,      MULTIPLE DATA FILE FILE-CODES
       ETC     RSZWRD,SZ,KEYPTR,SPEC],II
SPEC   ZERO                    KEY OFFSET AND LENGTH TAKEN FROM
       ZERO                       UTILIZATION RECORD
FCS    OCT     2
       VFD     18/,A18/DE
       VFD     18/,A18/DF
EE     BSS     2
SZ     DEC     0               UFAS PUTS RECORD SIZE HERE
*                              ON GET NEXT
ADDR   BSS     20
       END
$      EXECUTE DUMP
$      PRMFL   II,R,R,DODSON/INDEX
$      PRMFL   DD,W,R,DODSON/DATA
$      PRMFL   DE,W,R,DODSON/DATA2
$      PRMFL   DF,W,R,DODSON/DATA3
$      COMMENT
```

Figure C-4. GMAP Program to Access an ISP File

```
1     8       16
$     COMMENT
$     COMMENT ADD RECORDS TO AN ISP FILE
$     COMMENT
$     COMMENT
$     GMAP    NDECK
$     LIMITS  ,32K
      LODM    .DMAC
      SYMREF  .DISP           ISP DIRECTORY - ALL FUNCTIONS
      SYMDEF  START
START NULL
OPN   DOPEN   FIBLS,AA,(FIBA,BB,SYM,.DISP,MODE,IO)
*
*
*             CREATE A RECORD
*             AFTER ALL RECORDS ADDED, GO TO EOF
*
PUT   NULL
      DLPUT   BB,,IVK     ADD THE RECORD TO THE FILE
      TRA     PUT
IVK   NULL
*     A RECORD ALREADY EXISTS WITH THIS KEY
      LDQ     =3HONG,DL
      MME     GEBORT
EOF   NULL
CLS   DCLOS   FIBLS,EE,(FIBA,BB)
      MME     GEFINI
AA    BSS     3
BB    FIBMAC  DD,ISP,[PROAR,ADDR,0,NBUF,4,
      ETC     ACCMOD,RAN,RSZWRD,SZ,KEYPTR,SPEC],II
*     RECORD AND CI (PAGE) SIZES TAKEN
*     FROM UTILIZATION RECORD
SPEC  ZERO    10,11           MUST MATCH KEY OFFSET, LENGTH IN
      ZERO                    UTILIZATION RECORD
EE    BSS     2
ADDR  BSS     4
SZ    OCT     0               GIVE SIZE HERE ON PUT IF RECORD
*                             NOT MAXIMUM SIZE
      BSS     15
      END
$     EXECUTE DUMP
$     PRMFL   II,R,R,DODSON/INDEX
$     PRMFL   DD,W,R,DODSON/DATA
$     COMMENT
$     COMMENT
$     COMMENT DELETE RECORDS FROM AN ISP FILE
$     COMMENT
```

Figure C-4 (cont). GMAP Program to Access an ISP File

```
1      8       16
$      COMMENT
$      GMAP    NDECK
$      LIMITS  ,32K
       LODM    .DMAC           GET UFAS MACROS
       SYMREF  .DISP           ISP FULL DIRECTORY
       SYMDEF  START
START  NULL
OPN    DOPEN   FIBLS,AA,(FIBA,BB,SYM,.DISP,MODE,IO)
*
*              PLACE KEY OF RECORD TO DELETE IN
*      KEY OFFSET AND LENGTH POSITION IN PROCESS AREA
*      WHEN NO MORE TO DELETE, GO TO EOF
*
DEL    NULL
       DLDEL   BB,,IVK         GET THE RECORD AND DELETE IT
       TRA     DEL
IVK    NULL
*      RECORD DOESNT EXIST OR IS ALREADY DELETED
       LDQ     =3HONG,DL
       MME     GEBORT
EOF    NULL
CLS    DCLOS   FIBLS,EE,(FIBA,BB)
       MME     GEFINI
AA     BSS     3
BB     FIBMAC  DD,ISP,[PROAR,ADDR,0,NBUF,4,
       ETC     FCLST,FCS,      MULTIPLE DATA FILE FILE-CODES
       ETC     ACCMOD,RAN,KEYPTR,SPEC],II
*      RECORD AND CI (PAGE) SIZES TAKEN
*      FROM UTILIZATION RECORD
SPEC   ZERO                    KEY OFFSET AND LENGTH TAKEN FROM
       ZERO                       UTILIZATION RECORD
FCS    OCT     2
       VFD     18/,A18/DE
       VFD     18/,A18/DF
EE     BSS     2
ADDR   BSS     20
       END
$      EXECUTE DUMP
$      PRMFL   II,R,R,DODSON/INDEX
$      PRMFL   DD,W,R,DODSON/DATA
$      PRMFL   DE,W,R,DODSON/DATA2
$      PRMFL   DF,W,R,DODSON/DATA3
```

Figure C-4 (cont). GMAP Program to Access an ISP File

```
1      8       16
$      IDENT   account #, NAME
$      GMAP    NDECK
$      LIMITS  ,28K
$      SELECT  UFAS/IMCV/MACRO
       LBL     DRX0,H6000H5.000
       TTL     H6000H5.000     UFF RELATIVE FILE BUILD        760130DRX0
       CPR     1976
       TTLDAT
       PMC     ON
       EDITP   ON
       LODM    .DMAC
       TTLS    UFF RELATIVE FILE BUILD PROGRAM
*
*              PURPOSE=      TO BUILD A UFF RELATIVE
*                            FILE, THE RECORDS OF WHICH CONTAIN THEIR
*                            ORDINAL NUMBERS IN THE FIRST FOUR CHARACTERS
*
*              CALLS=        FIBMAC    (DECLARATIVE MACRO)
*                            .DOPEN
*                            .DPUT
*                            .DCLOS
*
*              CALLED BY=    ACTIVITY INITIATION ROUTINE
*
*
*
*
*
*
       EJECT
       SYMDEF  .DDRX0,.DBLD

       SYMREF  .DRELS,.DPRNT,U.ATOG

*
*
*      DECLARE  FILES
*
*
FIB    FIBMAC  FC,UFF,[NBUF,2,RSZ,100,CISZ,320,PROAR,WAREA1,0,
       ETC     FSCODE,STAT1,0,ORG,REL]
*
FIBOUT FIBMAC  P*,GFRC,[NBUF,2,RSZ,102,CISZ,320,PROAR,WAREA2,0,
       ETC     FSCODE,STAT2,0,IFLR,0,MEDCOD,3,EXTCOD,GBCD,OUTTRN,U.ATOG]
```

Figure C-5. GMAP Programs to Build and Position UFF Relative Files

```
1      8      16
*
*
.DDRXO NULL
.DBLD  NULL
*
*
*      INITIALIZE SYSOUT SWITCH
*
       STZ    SWITCH,$
*
*
*      OPEN FILES
*
       DOPEN  FIBLS,LIST,(FIBA,FIB,SYM,.DRELS,MODE,OT)
*
*      CHECK DOPEN STATUS CODE
*
       LDX    .XR1,STAT1,$
       CMPX   .XR1,GOOD,$
       TNZ    ABORT4,$

       DOPEN  FIBLS,LOUT,(FIBA,FIBOUT,SYM,.DPRNT,MODE,OT)

       LDX    .XR2,STAT2,$
       CMPX   .XR2,GOOD,$
       TNZ    ABORT5,$

*
*
*      INITIALIZE BINARY RECORD COUNT
*
       STZ    BRCCNT,$
*
BACK   NULL
*
*      ADD 1 TO BINARY RECORD COUNT
*
       AOS    BRCCNT,$
*
*      CALL ROUTINE TO CONVERT RECORD COUNT FROM BINARY INTO
*      ASCII CHARACTERS
*
       TSX    .XR3,.DTCNV
*
*      PLUG ASCII RECORD COUNT INTO RECORD AREA
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*
       LDA     ARCCNT,$
       STA     WAREA1,$
*
       CMPA    =0040061060060         ARCCNT=100? (KLUDGY SYSTEM!!!!)
       TZE     EOF,$
*
*
*      PUT RECORD INTO FILE
*
       DLPUT FIB,VIB,EOF,(RSZ,100)
*
*      CHECK STATUS CODE
*

       LDX     .XR1,STAT1,$
       CMPX    .XR1,GOOD,$
       TNZ     ABORT1,$
*
*      WRITE MESSAGE ONTO SYSOUT
*
       LDQ     SWITCH,$
       TZE     BACK,$
*
       DLPUT FIBOUT,VIBOUT,ABORT2,(RSZ,102,RPCOD,63,SLCOD,0)
*
*      CHECK STATUS CODE
*
       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABORT3,$
       TRA     BACK,$
*
EOF    NULL
*
*      NORMAL TERMINATION UPON END OF FILE
*
*
*      CLOSES FILES
*
       DCLOS   FIBLS,LIST,(FIBA,FIB,REW,RWFL)
*
*      CHECK DCLOS STATUS CODE
*
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
______________________________________________________________
       LDX     .XR1,STAT1,$
       CMPX    .XR1,GOOD,$
       TNZ     ABORT6,$
*
*
       DCLOS   FIBLS,LOUT,(FIBA,FIBOUT,REW,NRFL)
*
*      CHECK DCLOS STATUS CODE
*
       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABORT7,$
*
       LDQ     =3HOOK,DL
       MME     GEFINI
*
*
ABORT1 NULL
*
*      BAD PUT ON THE WORKING FILE
*
       LDQ     =3H0A1,DL
       MME     GEBORT
*
ABORT2 NULL
*
*      ALTERNATE RETURN TAKEN UPON PUT ON SYSOUT
*
       LDQ     =3H0A2,DL
       MME     GEBORT
*
ABORT3 NULL
*
*      BAD PUT ON SYSOUT
*
       LDQ     =3H0A3,DL
       MME     GEBORT
*
ABORT4 NULL
*
*      BAD OPEN ON THE WORKING FILE
*
       LDQ     =3H0A4,DL
       MME     GEBORT
*
```

Figure C-5 (cont). GMAP Programs to Build and Positoon UFF Relative Files

```
1      8       16
ABORT5 NULL
*
*      BAD OPEN ON SYSOUT
*
       LDQ     =3H0A5,DL
       MME     GEBORT
*
ABORT6 NULL
*
*      BAD CLOSE ON THE WORKING FILE
*
       LDQ     =3H0A6,DL
       MME     GEBORT
*
ABORT7 NULL
*
*      BAD CLOSE ON SYSOUT
*
       LDQ     =3H0A7,DL
       MME     GEBORT
*
*
*      DECLARATION OF LOCAL VARIABLES
*
SWITCH BSS     1
BRCCNT BSS     1
ARCCNT BSS     1
WAREA1 NULL
WAREA2 ASCII   10,
       ASCII   10,
       ASCII   10,
LIST   BSS     3
LOUT   BSS     3
VIB    BSS     2
VIBOUT BSS     2
STAT1  BSS     1
STAT2  BSS     1
*
GOOD   ASCII   1,0000
*
*
+      DECLARATION OF LOCAL CONSTANTS
*
.XR1   EQU     1
.XR2   EQU     2
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
.XR3   EQU     3
*


.DTCNV NULL
*
*      ROUTINE TO CONVERT A BINARY NUMBER STORED IN CELL BRCCNT
*      INTO A 4-DIGIT ASCII CHARACTERS STORED INTO ARCCNT
*
*      CLEAR OUTPUT CELL ARCCNT
*
       STZ     ARCCNT,$                CLEAR ARCCNT
*
*      INITIALIZE DIGIT COUNT TO 1
*
       EAX     .XR2,1
*
*      LOAD INPUT INTO Q REGISTER
*
       LDQ     BRCCNT,$
*
*      BINARY-TO-ASCII CONVERSION
*
BCK    NULL
*
       DIV     TEN,$                   Q = Q/10;  A = REMAINDER(Q/10)
       EAX     .XR1,0,AL
       TNZ     NXT,$
       QLR     0
       TNZ     NXT,$
       EAX     .XR1,10               IF Q=0 AND A=0 THEN STORE BLANK
NXT    LDA     DIGITS,.XR1           LOAD DIGIT WORD
       ANA     =0777000,DU             ISOLATE DIGIT CHARACTER
       ORA     ARCCNT,$                STORE DIGIT CHARACTER
       CMPX    .XR2,4,DU             IS IT THE END?
       TZE     OUT,$                   YES.
       ARL     9                       PREPARE SPACE FOR NEXT DIGIT
       STA     ARCCNT,$
       ADLX    .XR2,1,DU
       TRA     BCK,$

OUT    NULL
*
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*      NORMAL  TERMINATION
*
       STA     ARCCNT,$
       TRA     0,.XR3                RETURN TO CALLER
*
*      DECLARATION OF LOCAL VARIABLES
*
DIGITS ASCII   1.0
       ASCII   1,1
       ASCII   1,2
       ASCII   1,3
       ASCII   1,4
       ASCII   1,5
       ASCII   1,6
       ASCII   1,7
       ASCII   1,8
       ASCII   1,9
       ASCII   1,
*
TEN    DEC     10

       END
$      SELECTA UFAS/IMCV/MAC
$      EXECUTE DUMP
$      LIMITS  ,10K
$      SELECT  UFAS/IMCV/CONTROL
$      FILE    FC,R1S,R
$      GMAP    NDECK
$      LIMITS  ,28K
$      SELECT  UFAS/IMCV/MACRO
       LBL     DRX1,H6000H5.000
       TTL     H6000H5.000     UFF RELATIVE FILE POSIT. TEST 760130DRX1
       CPR     1976
       TTLDAT
       PMC     ON
       EDITP   ON
       LODM    .DMAC
       TTLS    UFF RELATIVE FILE POSITIONING
*              TEST PROGRAM
*
*              PURPOSE=      TO EXERCISE THE FILE POSITIONING FUNCTIONS
*                            REWIND, SPACE FORWARD AND BACKSPACE ON AN
*                            UFF RELATIVE FILE
*
*              CALLS=        FIBMAC    (DECLARATIVE MACRO)
*                            DOPEN
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*                            DLGXT
*                            DLPOS
*                            DCLOS
*
*              CALLED BY=    ACTIVITY INITIATION ROUTINE
*
*
*
*

       EJECT

       SYMDEF  .DDRX1,.DTFP

       SYMREF  .DRELS,.DPRNT,U.ATOG

*
*
*
*      DECLARE FILES
*
FIB    FIBMAC  FC,UFF,[NBUF,2,RSZ,100,CISZ,320,PROAR,WAREA1,0,
       ETC     FSCODE,STAT1,0,ORG,REL]


FIBOUT FIBMAC  P*,GFRC,[NBUF,2,RSZ,102,CISZ,320,PROAR,WAREA2,0,
       ETC     FSCODE,STAT2,0,IFLR,0,MEDCOD,3,EXTCOD,GBCD,OUTTRN,U.ATOG]


.DDRX1 NULL
.DTFP  NULL

*
*      INITIALIZE SYSOUT SWITCH
*
       STZ     SWITCH,$
*
*
*      INITIALIZE PHASE AND STEP STATUSES
*
       LDA     =6HPHAS01
       STA     PHASST,$                        INITIALIZE PHASE STATUS

       LDA     =6HSTP0.0
       STA     STEPST,$                        INITIALIZE STEP STATUS
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8        16
*
*      OPEN FILES
*
       DOPEN    FIBLS,LIST,(FIBA,FIB,SYM,.DRELS,MODE,TN)
*
*      CHECK DOPEN STATUS CODE
*
       LDX      .XR1,STAT1,$
       CMPX     .XR1,GOOD,$
       TNZ      ABOR1,$

       DOPEN    FIBLS,LOUT,(FIBA,FIBOUT,SYM,.DPRNT,MODE,OT)
*
*      CHECK DOPEN STATUS CODE
*
       LDX      .XR1,STAT2,$
       CMPX     .XR1,GOOD,$
       TNZ      ABOR2,$
*
*      SET STEP STATUS
*
       LDA      =6HSTP1.0
       STA      STEPST,$

BACK1  NULL
*
*      INITIALIZE LOCAL VARIABLES
*
       STZ      GETCNT,$
       STZ      BCKOUT,$
       STZ      FOROUT,$
       STZ      FOROUT,$
       STZ      STAT1,$
       STZ      STAT2,$
*
*
*
*      PERFORM N LOGICAL GETS, N BEING EQUAL TO MAXCNT
*
BACK2  NULL
*
*      GET A LOGICAL RECORD
*
       DLGXT    FIB,VIB,EOF,(RSZ,100)
*
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*      CHECK DLGXT STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR3,$
*
*      WRITE THE LOGICAL RECORD JUST READ ONTO SYSOUT
*
       LDQ     SWITCH,$
       TZE     SKIP1,$
*
       DLPUT   FIBOUT,VIBOUT,ABOR4,(RSZ,102,RPCOD,63,SLCOD,0)
*
*      CHECK DLPUT STATUS CODE
*
       LDX     .XR3,STAT2,$
       CMPX    .XR3,GOOD,$
       TNZ     ABOR5,$
*
SKIP1  NULL
*
*
*      ADD 1 TO THE COUNT OF GETS
*
       AOS     GETCNT,$
*
*
*      CHECK WHETHER OR NOT THE LIMIT OF THE NUMBER OF GETS HAS BEEN
*      REACHED
*
       LDA     GETCNT,$
       CMPA    MAXCNT,$
       TNZ     BACK2,$
*
*
*
*      SET STEP STATUS
*
       LDA     =6HSTP2.Q
       STA     STEPST,$
*
*      BACKSPACE BY N LOGICAL RECORDS, N BEING EQUAL TO BCKOUT
*
       DLPOS   FIB,VIB,ABOR6,(BACKI,BCKNB,OUTNBI,BCKOUT)
*
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*      CHECK DLPOS STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR7,$
*
*

*      SET STEP STATUS
*
       LDA     =6HSTP3.0
       STA     STEPST,$
*
*      PERFORM ONE GET
*
       DLGXT   FIB,VIB,EOF,(RSZ,100)
*
*      CHECK DLGXT STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR3,$
*
       LDA     WAREA1,$
       CMPA    RECHD1,$
       TNZ     ABOR14,$
*
*
*      WRITE THE LOGICAL RECORD JUST READ ONTO SYSOUT
*
       LDQ     SWITCH,$
       TZE     SKIP2,$
*
       DLPUT   FIBOUT,VIBOUT,ABOR4,(RSZ,102,RPCOD,63,SLCOD,0)
*
*      CHECK DLPUT STATUS CODE
*
       LDX     .XR3,STAT2,$
       CMPX    .XR3,GOOD,$
       TNZ     ABOR5,$
*
SKIP2  NULL
*
*
*      SET STEP STATUS
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
________________________________________________________________
*
       LDA     =6HSTP4.0
       STA     STEPST,$
*
*      FORWARD SPACE BY N=7 LOGICAL RECORDS
*
       DLPOS   FIB,VIB,ABOR7,(FORW,7,OUTNBI,FOROUT)
*
*      CHECK DLPOS STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR8,$
*
*      SET STEP STATUS
*
       LDA     =6HSTP5.0
       STA     STEPST,$
*
*      PERFORM THE FINAL GET
*
       DLGXT   FIB,VIB,EOF,(RSZ,100)
*
*      CHECK DLGXT STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR3,$
*
       LDA     WAREA1,$
       CMPA    RECHD2,$
       TNZ     ABOR15,$
*
*
*      WRITE THE LOGICAL RECORD JUST READ ONTO SYSOUT
*
       LDQ     SWITCH,$
       TZE     SKIP3,$
*
       DLPUT   FIBOUT,VIBOUT,ABOR4,(RSZ,102,RPCOD,63,SLCOD,0)
*
*      CHECK DLPUT STATUS CODE
*
       LDX     .XR3,STAT2,$
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
       CMPX    .XR3,GOOD,$
       TNZ     ABOR5,$
*
SKIP3  NULL
*
*
*      SET STEP STATUS
*
       LDA     =6HSTP6.0
       STA     STEPST,$
*
*      CHECK THE VALUES OF BCKOUT AND FOROUT
*
       LDA     BCKOUT,$

       CMPA    BCKNB,$                BCKOUT = BCKNB?
*      TNZ     ABOR9,$               NO.
*
*
       LDA     FOROUT,$
       CMPA    =7,DL                     FOROUT = 7?
       TNZ     ABOR10,$               NO.
*
*      CHECK WHETHER OR NOT IT IS THE FIRST TIME (PHASE 1)
*
       LDA     PHASST,$
       CMPA    =6HPHAS01
       TNZ     RETRN,$
*
*      REWIND FILE
*
       DLPOS   FIB,VIB,ABOR11,(BEGIN,FILE)
*
*      CHECK DLPOS STATUS CODE
*
       LDX     .XR2,STAT1,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR12,$
*
*
*      SET PHRASE STATUS
*
       LDA     =6HPHAS02              BEGINNING OF PHASE 2
       STA     PHASST,$
       TRA     BACK1,$
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
*
EOF    LDQ     =3H0A0,DL
       MME     GEBORT

ABOR1  LDQ     =3H0A1,DL
       MME     GEBORT

ABOR2  LDQ     =3H0A2,DL
       MME     GEBORT

ABOR3  LDQ     =3H0A3,DL
       MME     GEBORT

ABOR4  LDQ     =3H0A4.DL
       MME     GEBORT

ABOR5  LDQ     =3H0A5,DL
       MME     GEBORT

ABOR6  LDQ     =3H0A6,DL
       MME     GEBORT

ABOR7  LDQ     =3H0A7,DL
       MME     GEBORT

ABOR8  LDQ     =3H0A8,DL
       MME     GEBORT

ABOR9  LDQ     =3H0A9,DL
       MME     GEBORT

ABOR10 LDQ     =3HA10,DL
       MME     GEBORT

ABOR11 LDQ     =3HA11,DL
       MME     GEBORT

ABOR12 LDQ     =3HA12,DL
       MME     GEBORT

ABOR13 LDQ     =3HA13,DL
       MME     GEBORT

ABOR14 LDQ     =3HA14,DL
       MME     GEBORT
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
________________________________________________________
ABOR15 LDQ     =3HA15,DL
       MME     GEBORT

ABOR16 LDQ     =3HA16,DL
       MME     GEBORT


RETRN  NULL
*
*      CLOSE FILES
*
       DCLOS   FIBLS,LIST,(FIBA,FIB,REW,RWFL)
*
*      CHECK DCLOS STATUS CODE
*
       LDX     .XR1,STAT1,$
       CMPX     .XR1,GOOD,$
       TNZ     ABOR11,$

       DCLOS   FIBLS,LOUT,(FIBA,FIBOUT,REW,NRFL)
*
*      CHECK DCLOS STATUS CODE
*
       LDX     .XR2,STAT2,$
       CMPX    .XR2,GOOD,$
       TNZ     ABOR16,$
*
*
       LDQ     =3HOOK,DL
       MME     GEFINI
*
*      DECLARATIONS OF LOCAL VARIABLES
*
SWITCH BSS     1
PHASST BSS     1
STEPST BSS     1
LIST   BSS     3
LOUT   BSS     3
VIB    BSS     3
VIBOUT BSS     3
GETCNT BSS     1
MAXCNT BSS     10
BCKNB  DEC     6
BCKOUT DEC     0
FOROUT DEC     0
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
_______________________________________________________________

STAT1  BSS     1
STAT2  BSS     1
WAREA1 NULL
WAREA2 ASCII   10,
       ASCII   10,
       ASCII   10,
*
GOOD   ASCII   1,0000
RECHD1 ASCII   1,   4
RECHD2 ASCII   1,  11
*
*
*      DECLARATION OF LOCAL CONSTANTS
*
.XR1   EQU     1
.XR2   EQU     2
.XR3   EQU     3
*

       END
$      SELECTA UFAS/IMCV/MAC
$      EXECUTE DUMP
$      SELECT  UFAS/IMCV/CONTROL
$      LIMITS  ,10K
$      FILE    FC,R1S,R
$      ENDJOB
```

Figure C-5 (cont). GMAP Programs to Build and Position UFF Relative Files

```
1      8       16
$      GMAP    NDECK
$      LIMITS  ,30K
       LODM    .DMAC
       SYMDEF  START
       SYMREF  .DIDYN          SYMREF THE DYNAMIC INDEX ROUTINES.
*
START  NULL
       DOPEN   FIBLS,FBLS1,(FIBA,FIBAD1,SYM,.DIDYN,MODE,IO)
*
*              POSITION ON KEY (KEY = 33).
       DLPOS   FIBAD1,VIBAD1,NOKEY,(FCT,EQ)
*
*              RANDOMLY INSERT RECORD (KEY = 42).
       MLR
       ADSC9   RECRD1,0,28
       ADSC9   RECRD,0,28
       DLPUT   FIBAD1,VIBAD1,BADKEY,(RSZ,28)
*
*              RETRIEVE NEXT SEQUENTIAL RECORD AFTER RECORD WHOSE
*                 KEY IS EQUAL TO "33".
       DLGXT   FIBAD1,,FILEND
*
*              CHANGE DATA OF RECORD AND REWRITE.
       MLR
       ADSC9   NEWDAT,0,16
       ADSC9   DATAD,0,16
       DLRWR   FIBAD1,VIBAD1,BADRIT,(RSZ,28)
*
*              RANDOMLY RETRIEVE RECORD (KEY = 9).
       LDA     9,DL
       STA     KEYAD
       DLGET   FIBAD1,,BADKEY,(RSZ,28)
*
*              DELETE RECORD (KEY = 9).
       DLDEL   FIBAD1,,BADKEY
*
       DCLOS   FIBLS,FBLS1,(FIBA,FIBAD1)
       MME     GEFINI
*
*
*
*              THE PROCESS AREA
*
RECRD  UASCI   2,PRIME KEY-
KEYAD  DEC     33
```

Figure C-6. GMAP Program to Process an Indexed File

```
1      8       16
_____________________________________________________________

DATAD  UASCI   4,RECORD DATA AREA
*
*
RECRD1 UASCI   2,PRIME KEY-
       DEC    42
       UASCI   4,PARTICULAR DATA
*
NEWDAT UASCI   4,CHANGED DATA
*
VIBAD1 BSS     2               VIB
*
FBLS1  BSS     3               FIBLIST
*
KYPRAD NULL                    PRIME KEY DESCRIPTION FOLLOWED BY ZEROS.
       ZERO    8,4
       ZERO    0
*
NOKEY  NULL                       ERROR ROUTINES WHOSE ADDRESSES ARE
BADRIT NULL                    SPECIFIED AT EACH IO CALL.  EACH
BADKEY NULL                    ROUTINE SHOULD EITHER ABORT OR RETURN
FILEND NULL                    TO UFAS VIA THE "UFSRET" MACRO.
*
ERRTN  NULL
       MME     GEBORT
       UFSRET  ERROR
*
*              FIB
*
FIBAD1 FIBMAC  10,UFF,[NBUF,4,CISZ,250,RSZ,28,FSCODE,F10,0,
       ETC     USERER,ERRTN,ORG,IND,ACCMOD,DYN,PROAR,RECRD,0,
       ETC     PCTFIL,200,KEYPTR,KYPRAD],11
*
F10    BSS     1               BITS 0-18, 2-BYTE STATUS CODE.
*
       END
$      EXECUTE DUMP
$      PRMFL   10,W,R,UFAS/EGOLF/DATA1
$      PRMFL   11,W,R,UFAS/EGOLF/INDEX1
$      ENDJOB
```

Figure C-6 (cont). GMAP Program to Process an Indexed File

```
1      8      16
-----------------------------------------------------------
       LODM   .DMAC                  (get UFAS macros from library)
.SM4   SET    0
.SM68  SET    0     }                (indicates UFAS SORT)
.SM73  SET    1
       600SM
       SORT   INOUT,,84              (variable-length records)
       FIELD  (A6,A6,A12)
       SEQ    (A1,A2,A3)
       SYMREF .DSEQ                  (load UFAS sub-package) one per FIB
       ZERO   .DSEQ                   required at FIB-2 - Output Symref
       ZERO   .DSEQ                   required at FIB-1 - Input Symref

                                               record size in bytes
INOUT  FIBMAC SA,UFF,[RSZ,84,CISZ,888,PROAR,REC,0] process area required
                                               SA is standard input
                                               file

REC    BSS    21                      (space for largest record)
```

NOTE: If a fixed-length record SORT is desired the following additional changes are required:

```
       LODM   .DMAC
.SM4   SET    0
.SM68  SET    0
.SM73  SET    1
       600SM
       SORT   INOUT             (change)
       FIELD  (A6,A6,A12)
       SEQ    (A1,A2,A3)
       SYMREF .DSEQ
       ZERO   .DSEQ
       ZERO   .DSEQ


INOUT  FIBMAC SA,UFF,[IFLR,1,RSZ,84,CISZ,888,PROAR,REC,0] (change)


REC    BSS    21                (space for largest record)
```

Figure C-7. Free-Standing SORT of a UFF Sequential File

APPENDIX D

GLOSSARY

Access Mode

Manner in which records are to be operated upon within a file.

Alternate Key

A subordinate key to the prime key whose contents serve to identify a record in an indexed file (see Prime Key).

ASCII

American Standard Code for Information Interchange (ASCII) is the 9-bit data code used to write American National Standard tapes (four bytes per 36-bit word).

Banner

Set of characters used to identify specific blocks in H2000 files.

Block

Physical unit of data normally composed of one or more logical records. On random access media, block is equivalent to control interval. On magnetic tape the physical block size is not fixed.

Buffer

Area of main memory for storing control intervals (physical blocks) of data. Size of buffer is determined by the control interval size and buffer control words.

Build File

To write into a file for the first time. UFAS label check routines ascertain that data file space is void and establish required labels. File attribute information is placed in the subfile label to permit control and validation of subsequent activities using this file.

Byte

Nine-bit data character (ASCII or EBCDIC).

Compile Time

Time at which a source program is translated by a compiler to an object program.

Control Interval

Smallest logical unit of data records addressable by a file access mode. On permanent storage the control interval is composed of one or more physical records.

Current Record Pointer

Conceptual entity that is used in the selection of the next record (a logical position indicator).

Data File

Collection of items (data records) that are logically related.

Device Control Table

Table containing information pertaining to a device, such as file code, device type, device address, and file name.

Dynamic Access

Storage or retrieval of a logical record in either a sequential mode or nonsequential (random) mode during the entire span of an Open function.

End-of-File Condition

Condition that occurs during execution of a Get Next function when no next logical record exists.

Extend Mode

Equivalent to opening in the output processing mode, except that the file is positioned, when opened, immediately after the last logical record of the file.

File

Collection of records.

File Format

Specific file structure that defines a specific block or control interval structure and label structure.

File Organization

Permanent logical file type established at the time the file is built.

File Space

Total mass storage space assigned to a user file by the GCOS Operating System. When applied to magnetic tape, the term means the entire available recording surface.

FLR

Fixed-length record has a data item format with a fixed number of characters.

Format

Specific arrangement of a set of data.

H2000

Series 2000 Operating System.

Indexed File

File with indexed organization.

Indexed Organization

Permanent logical file type in which each record is identified by the value of one or more keys within the record.

Input File

File that is opened in the input mode.

Input/Output Mode

State of a file after execution of an OPEN statement and before execution of a CLOSE statement.

Integrated Organization

Permanent logical file type in which each record occurrence is identified by a record type code. N set pointers in each record describe the relationships among the records.

Invalid Key Condition

Condition that occurs when a specific value of the key associated with an indexed or relative file is determined to be invalid.

ISP

Indexed Sequential Processor that provides capabilities for creating and processing indexed sequential files.

Key

Data item that serves to identify the location of a record, or data items that serve to identify the ordering of data. A key is composed of a key value and a pointer.

Key of Reference

Key, prime or alternate, currently being used to access records within a file.

Magnetic Tape

A storage medium in which information can be stored and accessed in sequential order.

Mass Storage File

Collection of records that is assigned to a mass storage medium, and can be accessed randomly.

Multi-Unit File

File contained on more than one unit of external media.

Non-Unit File

File for which the term unit has no particular meaning.

Object Time

Time at which an object program is executed.

Open Mode

State of a file after execution of an OPEN statement and before execution of a CLOSE statement.

Optional File

Sequential file opened for input that may or may not be present on a given execution activity. Present means defined to the GCOS Operating System.

Output File

File that is opened in either the output mode or the extend mode.

Output Mode

State of a file after execution of an OPEN output or extend statement and before the execution of a CLOSE statement.

Page

Fixed size unit of record storage, also designated as control interval.

Padded Record

A record filled with assigned characters in H2000 files.

Partitioned Record

Variable-length logical record that is larger than a block size being used. The logical record is broken up or partitioned over two or more physical records. Partitioned records are used in processing GFRC sequential files.

Prime Key

Key whose contents uniquely identify a record within a file.

Process Area

Storage area reserved for user processing of the logical record.

Processing Mode

State of a file after execution of an open function.

Random Access

Access mode in which a specified value of a key in the object program identifies the logical record that is obtained, deleted from, or placed in a relative or indexed file.

Record

Most inclusive data item. Synonymous with logical record.

Record Area

Storage area allocated for the purpose of processing the record described in a record description entry in the File Section.

Reel

Magnetic tape unit.

Relative File

File with relative organization.

Relative Key

Key whose contents identify a logical record in a relative file.

Relative Organization

Permanent logical file type in which each record is uniquely identified by an integer value greater than zero that specifies the record's ordinal position in the file.

Removable Mass Storage

Mass storage unit that allows logical or physical swapping of files.

Run Unit

Set of one or more object programs that function as a unit at object time to provide problem solutions.

Sequential Access

Access mode in which logical records are retrieved or entered in a file in a consecutive predecessor-to-successor logical record sequence determined by the order of the records in a file.

Sequential File

File organization in which a record is identified by a predecessor-successor relationship established when the record is entered in the file.

Single Unit File

File entirely contained in one unit.

Spanned Record

Logical record contained in two or more blocks.

Tape

See Magnetic Tape.

Unit

Logical module of external storage.

Utilization Record

First record of the data file and indexed file of an ISP file. This record contains the file characteristics and has always 62 words.

VLR

Variable-length record has a data item format that does not specify the size of each individual record, but allows the length of the record to vary up to a maximum size.

APPENDIX E

INDEXED FILE OVERFLOW

INDEXED DATA FILE

The records of an indexed file are originally placed in the data file according to ascending prime key order. An index is built to describe the key ranges of each data control interval in the file. This index is not changed during the life of the file. As inserts are made in the data file, an attempt is made to put the record in the appropriate control interval according to its prime key value.

If the record does not fit in the desired control interval, either the records must be shifted or a chain must be established to a space where the record will fit. These records that are inserted after file load time and that will not fit in the control interval described in the index are referred to as overflow records. The following discussion describes several methods used by UFAS to place and retrieve these records efficiently.

METHODS

Percent Fill

Space may be left on every data control interval at load time for anticipated additional records. The amount of space to be used at load time in each control interval can be specified as an amount less than the total control interval size by using the PCTFIL parameter of the FIB macro. The extra space will then be available for additional records. This method is adequate if records are to be added to the file in a random order unknown at load time, because it precludes the need for additional input/output activity if the added records fit in the extra space allocated by percent fill. However, it is not an efficient method if many records are to be added to a particular area of the file and omitted from other areas.

General Overflow

When an indexed data file is initially loaded, any unused space remaining after loading is completed is made available to hold records added during subsequent file updates. If an attempt is made to add a record and it does not fit in the desired control interval, because either no percent fill was specified or the percent fill space available was used by another added record, the record will be placed in the unused space in an overflow control interval. Additional added records will use this same overflow control interval until the control interval is full. The next control interval then becomes the overflow control interval and overflow records are put into this control interval until it is full. This method makes optimum use of disk space. The file need only be as large as the load, because space will be acquired (up to the user defined file limit) by UFAS as required. This method has several disadvantages. Over a period of time, overflow records from a single control interval may become scattered throughout the general overflow region and it may be necessary to effect additional input/output activity to retrieve one of these records. Also, under concurrent access when one record is added to a general overflow control interval, no other user can add a record to general overflow until the first user has taken a checkpoint. Thus, many users can be held waiting in a concurrent access mode for the overflow control interval to become free. This is specially noticeable in a TP environment. It should be noted, however, that percent fill and general overflow may be used in conjunction with each other.

Inventoried Overflow

Space may be made available for record overflow in many areas of the data file under control of an inventory created by UFAS. Two types or levels of overflow are available to the user:

- The first level, called local overflow, is made up of groups of overflow control intervals distributed at equally spaced intervals throughout the file. The intervals are specified in the FIB macro by the parameter LOVINC. Each group of local overflow control intervals accepts records only from the data control interval adjacent to its group. This overcomes the disadvantages mentioned under general overflow. That is, overflow records for a particular region are placed together in the same overflow control interval, thus reducing the input/output activity required to access these records; and, many users may add overflow records concurrently without being held up because the overflow control intervals are distributed.

- The second level, called inventoried general overflow, is used when no local overflow is specified or when a group of local overflow control intervals becomes full. The general overflow, in this instance, is inventoried to permit several control intervals to accept overflow records concurrently.

An attempt is made to place overflow records in the inventoried general overflow region according to a ratio-type hash of key value, in order to maintain specific areas in the inventoried general overflow, but the whole general overflow region is available for overflow records from anywhere in the file. The advantage of the second level of overflow is the efficient use of file space when many records are added to only one region of the file. The inventoried general overflow is specified in the FIB macro by the parameter GOVINC.

The inventoried overflow method is more economic and enables files to operate longer efficiently between reloading activities than the percent fill or general overflow method. The inventoried overflow method is specifically recommended if files are to be updated concurrently.

The $ DATA .U file may also be used to provide the local or general inventoried overflow parameters. (See Appendix A and FIB parameter defaults in Section V.) No overflow can be specified by setting the particular parameter to 511.

ALTERNATE KEY INDEX

Every time a record is added to the data file an entry is placed in an alternate key index structure for each alternate key of the record. The index entries are maintained in sort order, thus the index control intervals for alternate keys may overflow. To allow for this overflow, space is provided in the alternate key fine level index control intervals as specified in the FIB macro by the percent fill parameter. Also, local overflow is available for the regions of each index. If local overflow is specified for the data file, the same ratio of control intervals is used for the index file. If no local overflow is specified, but general overflow is specified for the data file, the same ration of control intervals is used for the index file. If neither local overflow nor general overflow are specified, the default is one local overflow control interval for every 15 index control intervals containing index entries and all unused index file control intervals are used for general overflow. To specify no local overflow, LOVINC and GOVINC (or $ DATA .U) must be set greater than 510, or no value must be specified for these parameters and for percent fill.

General overflow will be available for all alternate keys to use, once a local overflow region has been filled or if no local overflow exists. The general overflow in the alternate key structure is not inventoried.

INDEX

2000/60
  2000/60 Magnetic Tape Interchange Facility 3-1

ACCESS
  ACCESS MODES 1-2
  FILE PROTECTION AND CONCURRENT ACCESS CONTROL 1-61

ACCESSIBILITY
  ACCESSIBILITY 1-25, 1-28

ACTIVITY
  Activity Restart 12-1

ADJUST
  Adjust Density 12-1, 12-2

ALLOCATING
  ALLOCATING A PRINTER 11-1

ALTERNATE
  ALTERNATE KEY INDEX E-3

AMERICAN
  UFF And American National Standard Tape File Format 1-20

ANS
  UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1
  USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4

ASSIGNMENT
  Space Assignment A-5

ATTRIBUTES
  PHYSICAL ATTRIBUTES 1-20, 1-33, 1-47, 1-52
  PHYSICAL ATTRIBUTES (Tape Files) 1-35

BATCH
  UFAS Directories For DM-IV (Batch Or Timesharing) 7-3

BEGINNING
  Beginning of Tape Label (BTL) 1-39, 1-55

BLOCK
  BLOCK COUNT 1-28, 1-30, 1-42, 1-57

BLOCK (cont)
  BLOCK LENGTH 1-29, 1-31
  FILE INFORMATION BLOCK MACRO 2-3, 3-2, 4-3, 5-3, 6-3, 11-3
  PPS Block control Word 1-37.1

BTL
  Beginning of Tape Label (BTL) 1-39, 1-55

BUFFER
  BUFFER OFFSET LENGTH 1-29, 1-31
  BUFFER POOLING 1-62, A-4

BUTTON
  Printer Button Interface 11-2

CAPABILITY
  DEDICATED PRINTER CAPABILITY 11-1

CHECKPOINT
  Checkpoint Files And Program Macro - DCKPT 2-41, 4-36, 5-34, 6-23
  Checkpoint Files Macro - DCKPF 2-40, 4-35, 5-33, 6-23
  Checkpoint Record Tracking 12-2
  CHECKPOINT RECORDS ON DATA TAPES A-3

CLOSE
  Close Macro - DCLOS 2-44, 3-23, 4-40, 5-36, 6-28, 11-9

COBOL
  COBOL 74 program C-1

COBOL-74
  COBOL-74 FIL-STATS 11-2

CODE
  major status code 10-1
  substatus code 10-1
  SYSTEM CODE 1-28, 1-30.1

CODES
  EXCEPTION CONDITIONS AND STATUS CODES 10-1
  media codes 11-1

CONCATENATE
  concatenate UFAS sequential files 8-1

CONCATENATION
  FILE CONCATENATION 8-3

CONCURRENT
  FILE PROTECTION AND CONCURRENT ACCESS CONTROL 1-61

CONDITIONS
  EXCEPTION CONDITIONS AND STATUS CODES 10-1

CONSOLE
  CONSOLE MESSAGES 10-18

CORRESPONDING
  CORRESPONDING HDR1 FIELDS 1-30

COUNT
  BLOCK COUNT 1-28, 1-30, 1-42, 1-57
  RECORD COUNT 1-57

CREATION
  CREATION DATE 1-28, 1-40, 1-56

DATA
  CHECKPOINT RECORDS ON DATA TAPES A-3
  DATA FORMAT 1-35, 1-48, 1-53, 1-20.1
  DATA REGION 1-7
  INDEXED DATA FILE E-1

DATE
  CREATION DATE 1-28, 1-40, 1-56
  EXPIRATION DATE 1-28
  RETENTION DATE 1-41

DCKPF
  Checkpoint Files Macro - DCKPF 2-40, 4-35, 5-33, 6-23

DCKPT
  Checkpoint Files And Program Macro - DCKPT 2-41, 4-36, 5-34, 6-23

DCLOS
  Close Macro - DCLOS 2-44, 3-23, 4-40, 5-36, 6-28, 11-9

DEDICATED
  DEDICATED PRINTER CAPABILITY 11-1
  Dedicated Tape 12-2, 12-8

DEFECTIVE
  Defective Volume Replacement 12-1

DELETE
  Delete Macro - DLDEL 4-20, 5-19, 6-17

DENSITIES
  MAGNETIC TAPE DENSITIES A-1

DENSITY
  Adjust Density 12-1, 12-2
  TAPE DENSITY 1-41

DINFO
  Information Macro - DINFO 2-38, 3-21, 4-32, 5-30, 6-24

DIRECTORIES
  UFAS Directories For DM-IV (Batch Or Timesharing) 7-3
  UFAS Directories For DM-IV-TP 7-3

DIRECTORY
  directory names 2-17, 3-10, 4-12, 5-11, 6-9
  Directory Selection 7-2

DLDEL
  Delete Macro - DLDEL 4-20, 5-19, 6-17

DLGET
  Get Macro - DLGET 4-29, 5-26, 6-21

DLGXT
  Get Next Macro - DLGXT 2-29, 3-17, 4-27, 5-25, 6-19

DLPOS
  Position Macro - DLPOS 2-31, 4-22, 5-20, 6-18

DLPUT
  Put Macro - DLPUT 2-24, 3-15, 4-16, 5-16, 6-14, 11-7

DLRLB
  Read User Labels - DLRLB 2-34

DLRWR
  Rewrite Macro - DLRWR 2-27, 4-18, 5-17, 6-15

DLWLB
  Write User Labels - DLWLB 2-35

DM-IV
  UFAS Directories For DM-IV (Batch Or Timesharing) 7-3

DM-IV-TP
  UFAS Directories For DM-IV-TP 7-3

DOPEN
  Open Macro - DOPEN 2-16, 3-8, 4-10, 5-10, 6-8, 11-6

DROLF
  Rollback Files Macro - DROLF 2-42, 4-37, 5-35, 6-27

DROLP
  Rollback Files And Program Macro - DROLP 2-43, 4-38, 5-35, 6-28

DWRAP
  Wrap-up Macro - DWRAP 2-43, 3-20, 4-39, 5-36, 6-28

DYNAMIC
  Dynamic 1-3

END-OF-FILE/REEL
  End-of-File/Reel Label 1-57
  End-of-File/Reel Label (EOF/EOR) 1-56

END-OF-FILE/VOLUME
  First End-of-File/Volume Label (EOF1/EOV1) 1-30
  Second End-of-File/Volume Label (EOF2/EOV2) 1-31

END-OF-REEL/FILE
  End-of-Reel/File Label (EOR/EOF) 1-42

EOF/EOR
  End-of-File/Reel Label (EOF/EOR) 1-56

EOF1/EOV1
  First End-of-File/Volume Label (EOF1/EOV1) 1-30

EOF2/EOV2
  Second End-of-File/Volume Label (EOF2/EOV2) 1-31

EOR/EOF
  End-of-Reel/File Label (EOR/EOF) 1-42

ERRONEOUS
  REEL NUMBER ERRONEOUS 1-41

EXCEPTION
  EXCEPTION CONDITIONS AND STATUS CODES 10-1

EXITS
  Label Exits 1-43

EXPIRATION
  EXPIRATION DATE 1-28

EXTEND
  Extend 1-3
  Extend Processing 1-33

EXTERNAL
  User External Labels 12-1

FACILITY
  2000/60 Magnetic Tape Interchange Facility 3-1
  H2000/60 Magnetic Tape Interchange Facility 1-52

FIB
  FIB macro 2-3, 3-2, 4-3, 5-3, 6-3

FIELDS
  CORRESPONDING HDR1 FIELDS 1-30

FIL-STATS
  COBOL-74 FIL-STATS 11-2

FILE
  FILE CONCATENATION 8-3

FILE (cont)
  FILE FORMATS 1-5
  File Generation Group 8-1
  FILE GENERATIONS 8-2
  File Group Index 8-2
  FILE IDENTIFIER 1-27
  FILE INFORMATION BLOCK MACRO 2-3, 3-2, 4-3, 5-3, 6-3, 11-3
  FILE NAME 1-41, 1-55, 1-57
  File Name Identified Tape 12-1
  FILE NAME IDENTIFIED TAPE (FNIT) 12-2
  FILE ORGANIZATIONS 1-2
  FILE PROTECTION AND CONCURRENT ACCESS CONTROL 1-61
  FILE SECTION NUMBER 1-27
  FILE SEQUENCE NUMBER 1-27, 1-41
  FILE SERIAL NUMBER 1-40, 1-55
  FILE SET IDENTIFIER 1-27
  GFRC File Format 1-34
  H2000 Tape File Format 1-52
  IBM Tape File Format 1-46
  Index File Size A-8
  INDEXED DATA FILE E-1
  ISP File Format 1-18
  Single File Input Processing 1-45, 1-46, 1-32.1
  UFF And American National Standard Tape File Format 1-20
  UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1
  Unified File Formats 1-5
  UNLABELED GFRC FILE PROCESSING 1-45
  Unlabeled tape file 8-1
  UNLABELED UFF FILE PROCESSING 1-32
  UNLABELED UNLT FILE PROCESSING 1-46
  UNLT TAPE FILE FORMAT 1-45

FILE-HEADER
  First File-Header Label (HDR1) 1-27, 1-48
  Second File-Header Label (HDR2) 1-29, 1-49

FILE-HEADER/END-OF-FILE
  Second File-Header/End-of-File (HDR2/EOF2) 1-49

FILES
  Checkpoint Files And Program Macro - DCKPT 2-41, 4-36, 5-34, 6-23
  Checkpoint Files Macro - DCKPF 2-40, 4-35, 5-33, 6-23
  concatenate UFAS sequential files 8-1
  PHYSICAL ATTRIBUTES (Tape Files) 1-35
  Rollback Files And Program Macro - DROLP 2-43, 4-38, 5-35, 6-28
  Rollback Files Macro - DROLF 2-42, 4-37, 5-35, 6-27
  System Input And Output Files 1-33
  UNLABELED GFRC TAPE FILES 1-44
  Unlabeled Tape Files 1-58
  UNLABELED UFF TAPE FILES 1-32
  USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4

FILGP
  $ FILGP control 1-27
  $ FILGP control card 8-3

FILL
  Percent Fill E-1

FIRST
  First End-of-File/Volume Label (EOF1/EOV1) 1-30
  First File-Header Label (HDR1) 1-27, 1-48

FNIT
  FILE NAME IDENTIFIED TAPE (FNIT) 12-2
  FNIT 12-1

FORMAT
  Control Word Format 1-37.1
  DATA FORMAT 1-35, 1-48, 1-53, 1-20.1
  GENERAL FORMAT 1-20, 1-35, 1-47, 1-53
  GFRC File Format 1-34
  H2000 Tape File Format 1-52
  IBM Tape File Format 1-46
  Indexed Format 1-9
  Integrated Format 1-15
  ISP File Format 1-18
  PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1
  RECORD FORMAT 1-31
  Relative Format 1-9
  Sequential Format 1-8
  UFF And American National Standard Tape File Format 1-20
  UNLT TAPE FILE FORMAT 1-45

FORMATS
  FILE FORMATS 1-5
  LABEL FORMATS AND PROCESSING 1-22, 1-38, 1-48, 1-53
  Unified File Formats 1-5

FORMULAS
  FORMULAS A-7

FSCODE
  GMAP FSCODE 11-2

FUNCTIONAL
  FUNCTIONAL MACROS 2-15, 3-7, 4-9, 5-9, 6-7, 11-5

FUNCTIONS
  Processing Functions 1-58
  Rerun functions 12-1, 12-2

GENERAL
  GENERAL FORMAT 1-20, 1-35, 1-47, 1-53
  General Overflow E-2

GENERATION
  File Generation Group 8-1
  GENERATION NUMBER 1-27
  GENERATION VERSION NUMBER 1-28

GENERATIONS
  FILE GENERATIONS 8-2

GET
  Get Macro - DLGET 4-29, 5-26, 6-21
  Get Next Macro - DLGXT 2-29, 3-17, 4-27, 5-25, 6-19

GFRC
  GFRC File Format 1-34
  UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1
  UNLABELED GFRC FILE PROCESSING 1-45
  UNLABELED GFRC TAPE FILES 1-44

GMAP
  GMAP FSCODE 11-2

GROUP
  File Generation Group 8-1
  File Group Index 8-2

H2000
  H2000 Tape File Format 1-52

H2000/60
  H2000/60 Magnetic Tape Interchange Facility 1-52

HDR1
  CORRESPONDING HDR1 FIELDS 1-30
  First File-Header Label (HDR1) 1-27, 1-48

HDR2
  Second File-Header Label (HDR2) 1-29, 1-49

HDR2/EOF2
  Second File-Header/End-of-File (HDR2/EOF2) 1-49

HEADER
  HEADER SEQUENCE NUMBER 1-41

IBM
  IBM Tape File Format 1-46
  UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1
  USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4

IDENTIFICATION
  INSTALLATION IDENTIFICATION 1-40

IDENTIFIED
  File Name Identified Tape 12-1
  FILE NAME IDENTIFIED TAPE (FNIT) 12-2

IDENTIFIER
  FILE IDENTIFIER 1-27
  FILE SET IDENTIFIER 1-27
  LABEL IDENTIFIER 1-39, 1-42, 1-55, 1-56
  LABEL IDENTIFIER AND NUMBER 1-25, 1-27, 1-29, 1-30, 1-31
  OWNER IDENTIFIER 1-25

IDENTIFIER (cont)
UFAS IDENTIFIER 1-40, 1-43

INDEX
ALTERNATE KEY INDEX E-3
File Group Index 8-2
Index File Size A-8

INDEXED
Indexed 1-2
INDEXED DATA FILE E-1
Indexed Format 1-9

INFORMATION
FILE INFORMATION BLOCK MACRO 2-3, 3-2, 4-3, 5-3, 6-3, 11-3
Information Macro - DINFO 2-38, 3-21, 4-32, 5-30, 6-24

INPUT
Input 1-3
Multi-File Input Processing 1-32.1
Multi-Volume Input Processing 1-31.1
Multiple Input Processing 1-45
Single File Input Processing 1-45, 1-46, 1-32.1
System Input And Output Files 1-33

INPUT/OUTPUT
Input/Output 1-3

INSTALLATION
INSTALLATION IDENTIFICATION 1-40

INTEGRATED
Integrated 1-2
Integrated Format 1-15

INTERCHANGE
2000/60 Magnetic Tape Interchange Facility 3-1
H2000/60 Magnetic Tape Interchange Facility 1-52

INTERFACE
Printer Button Interface 11-2

INTERVAL
Control Interval Size A-7

INVENTORIED
Inventoried Overflow E-2

INVENTORY
INVENTORY REGION 1-17

ISP
ISP File Format 1-18

KEY
ALTERNATE KEY INDEX E-3

LABEL
Beginning of Tape Label (BTL) 1-39, 1-55
End-of-File/Reel Label 1-57

LABEL (cont)
End-of-File/Reel Label (EOF/EOR) 1-56
End-of-Reel/File Label (EOR/EOF) 1-42
First End-of-File/Volume Label (EOF1/EOV1) 1-30
First File-Header Label (HDR1) 1-27, 1-48
Label Exits 1-43
LABEL FORMATS AND PROCESSING 1-22, 1-38, 1-48, 1-53
LABEL IDENTIFIER 1-39, 1-42, 1-55, 1-56
LABEL IDENTIFIER AND NUMBER 1-25, 1-27, 1-29, 1-30, 1-31
LABEL PROCESSING MESSAGES 10-18
LABEL STANDARD VERSION 1-25
PARTIAL TAPE LABEL (PTL) 1-43
Second End-of-File/Volume Label (EOF2/EOV2) 1-31
Second File-Header Label (HDR2) 1-29, 1-49
USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4
Volume-Header Label (VOL1) 1-25, 1-48

LABELS
Read User Labels - DLRLB 2-34
User External Labels 12-1
Write User Labels - DLWLB 2-35

LENGTH
BLOCK LENGTH 1-29, 1-31
BUFFER OFFSET LENGTH 1-29, 1-31
RECORD LENGTH 1-29, 1-31

MACRO
Checkpoint Files And Program Macro - DCKPT 2-41, 4-36, 5-34, 6-23
Checkpoint Files Macro - DCKPF 2-40, 4-35, 5-33, 6-23
Close Macro - DCLOS 2-44, 3-23, 4-40, 5-36, 6-28, 11-9
Delete Macro - DLDEL 4-20, 5-19, 6-17
FIB macro 2-3, 3-2, 4-3, 5-3, 6-3
FILE INFORMATION BLOCK MACRO 2-3, 3-2, 4-3, 5-3, 6-3, 11-3
Get Macro - DLGET 4-29, 5-26, 6-21
Get Next Macro - DLGXT 2-29, 3-17, 4-27, 5-25, 6-19
Information Macro - DINFO 2-38, 3-21, 4-32, 5-30, 6-24
Open Macro - DOPEN 2-16, 3-8, 4-10, 5-10, 6-8, 11-6
Position Macro - DLPOS 2-31, 4-22, 5-20, 6-18
Put Macro - DLPUT 2-24, 3-15, 4-16, 5-16, 6-14, 11-7
Return Macro - UFSRET 2-37, 3-19, 4-31, 5-29, 6-22
Rewrite Macro - DLRWR 2-27, 4-18, 5-17, 6-15
Rollback Files And Program Macro - DROLP 2-43, 4-38, 5-35, 6-28

MACRO (cont)
  Rollback Files Macro - DROLF 2-42, 4-37, 5-35, 6-27
  Wrap-up Macro - DWRAP 2-43, 3-20, 4-39, 5-36, 6-28

MACROS
  FUNCTIONAL MACROS 2-15, 3-7, 4-9, 5-9, 6-7, 11-5

MAGNETIC
  2000/60 Magnetic Tape Interchange Facility 3-1
  H2000/60 Magnetic Tape Interchange Facility 1-52
  MAGNETIC TAPE DENSITIES A-1

MAJOR
  major status code 10-1

MEDIA
  media codes 11-1

MESSAGES
  CONSOLE MESSAGES 10-18
  LABEL PROCESSING MESSAGES 10-18

METHODS
  METHODS E-1

MODES
  ACCESS MODES 1-2
  PROCESSING MODES 1-3

MULTI-FILE
  Multi-File Input Processing 1-32.1

MULTI-VOLUME
  Multi-Volume Input Processing 1-31.1
  Multi-Volume Output Processing 1-31.1

MULTIPLE
  Multiple Input Processing 1-45

NAME
  FILE NAME 1-41, 1-55, 1-57
  File Name Identified Tape 12-1
  FILE NAME IDENTIFIED TAPE (FNIT) 12-2
  Routine Package SYMREF Name 2-17, 3-10, 4-12, 5-11, 6-9

NAMES
  directory names 2-17, 3-10, 4-12, 5-11, 6-9

NATIONAL
  UFF And American National Standard Tape File Format 1-20

NEXT
  Get Next Macro - DLGXT 2-29, 3-17, 4-27, 5-25, 6-19
  NEXT VOLUME SERIAL NUMBER 1-25, 1-30.1

NUMBER
  FILE SECTION NUMBER 1-27
  FILE SEQUENCE NUMBER 1-27, 1-41
  FILE SERIAL NUMBER 1-40, 1-55
  GENERATION NUMBER 1-27
  GENERATION VERSION NUMBER 1-28
  HEADER SEQUENCE NUMBER 1-41
  LABEL IDENTIFIER AND NUMBER 1-25, 1-27, 1-29, 1-30, 1-31
  NEXT VOLUME SERIAL NUMBER 1-25, 1-30.1
  REEL NUMBER ERRONEOUS 1-41
  REEL SEQUENCE NUMBER 1-40, 1-55
  REEL SERIAL NUMBER 1-43
  TAPE REEL SERIAL NUMBER 1-40, 1-55
  VOLUME SERIAL NUMBER 1-25

OFFSET
  BUFFER OFFSET LENGTH 1-29, 1-31

OPEN
  Open Macro - DOPEN 2-16, 3-8, 4-10, 5-10, 6-8, 11-6

ORGANIZATION
  UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1

ORGANIZATIONS
  FILE ORGANIZATIONS 1-2

OUTPUT
  Multi-Volume Output Processing 1-31.1
  Output 1-3
  Output Processing 1-33, 1-45
  System Input And Output Files 1-33

OVERFLOW
  General Overflow E-2
  Inventoried Overflow E-2

OWNER
  OWNER IDENTIFIER 1-25

PACKAGE
  Routine Package SYMREF Name 2-17, 3-10, 4-12, 5-11, 6-9

PAGE
  PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1

PARTIAL
  PARTIAL TAPE LABEL (PTL) 1-43

PARTITIONED
  Partitioned Records 1-37

PERCENT
  Percent Fill E-1

PERIOD
  RETENTION PERIOD 1-41, 1-56

PHYSICAL
  PHYSICAL ATTRIBUTES 1-20, 1-33, 1-47, 1-52

PHYSICAL (cont)
PHYSICAL ATTRIBUTES (Tape Files) 1-35

POOLING
BUFFER POOLING 1-62, A-4

POSITION
Position Macro - DLPOS 2-31, 4-22, 5-20, 6-18

PPS
PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1
PPS Block control Word 1-37.1
PPS Record Control Word 1-37.2

PRINTER
ALLOCATING A PRINTER 11-1
DEDICATED PRINTER CAPABILITY 11-1
Printer Button Interface 11-2
PRINTER STATUS 11-2

PRINTING
PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1

PROCEDURES
PROCEDURES 11-2

PROCESSING
Extend Processing 1-33
LABEL FORMATS AND PROCESSING 1-22, 1-38, 1-48, 1-53
LABEL PROCESSING MESSAGES 10-18
Multi-File Input Processing 1-32.1
Multi-Volume Input Processing 1-31.1
Multi-Volume Output Processing 1-31.1
Multiple Input Processing 1-45
Output Processing 1-33, 1-45
PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1
Processing Functions 1-58
PROCESSING MODES 1-3
Single File Input Processing 1-45, 1-46, 1-32.1
UNLABELED GFRC FILE PROCESSING 1-45
UNLABELED UFF FILE PROCESSING 1-32
UNLABELED UNLT FILE PROCESSING 1-46
USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4

PROGRAM
Checkpoint Files And Program Macro - DCKPT 2-41, 4-36, 5-34, 6-23
COBOL 74 program C-1
Rollback Files And Program Macro - DROLP 2-43, 4-38, 5-35, 6-28

PROTECTION
FILE PROTECTION AND CONCURRENT ACCESS CONTROL 1-61

PTL
PARTIAL TAPE LABEL (PTL) 1-43

PUT
Put Macro - DLPUT 2-24, 3-15, 4-16, 5-16, 6-14, 11-7

RANDOM
Random 1-3

READ
Read User Labels - DLRLB 2-34

RECORD
Checkpoint Record Tracking 12-2
PPS Record Control Word 1-37.2
RECORD COUNT 1-57
RECORD FORMAT 1-31
RECORD LENGTH 1-29, 1-31

RECORDS
CHECKPOINT RECORDS ON DATA TAPES A-3
Partitioned Records 1-37
Variable-Length Records 1-36

REEL
REEL NUMBER ERRONEOUS 1-41
REEL SEQUENCE NUMBER 1-40, 1-55
REEL SERIAL NUMBER 1-43
TAPE REEL SERIAL NUMBER 1-40, 1-55

REGION
DATA REGION 1-7
INVENTORY REGION 1-17

RELATIVE
Relative 1-2
Relative Format 1-9

REPLACEMENT
Defective Volume Replacement 12-1

RERUN
RERUN AND TAPE UTILITIES 12-1
Rerun functions 12-1, 12-2

RESERVED
RESERVED 1-25, 1-28, 1-29, 1-31, 1-41, 1-56

RESTART
Activity Restart 12-1

RESTRICTIONS
RESTRICTIONS 8-1

RETENTION
RETENTION DATE 1-41
RETENTION PERIOD 1-41, 1-56

RETURN
Return Macro - UFSRET 2-37, 3-19, 4-31, 5-29, 6-22

REWRITE
Rewrite Macro - DLRWR 2-27, 4-18, 5-17, 6-15

ROLLBACK
- Rollback Files And Program Macro - DROLP 2-43, 4-38, 5-35, 6-28
- Rollback Files Macro - DROLF 2-42, 4-37, 5-35, 6-27

ROUTINE
- Routine Package SYMREF Name 2-17, 3-10, 4-12, 5-11, 6-9

SECOND
- Second End-of-File/Volume Label (EOF2/EOV2) 1-31
- Second File-Header Label (HDR2) 1-29, 1-49
- Second File-Header/End-of-File (HDR2/EOF2) 1-49

SECTION
- FILE SECTION NUMBER 1-27

SELECTION
- Directory Selection 7-2

SEQUENCE
- FILE SEQUENCE NUMBER 1-27, 1-41
- HEADER SEQUENCE NUMBER 1-41
- REEL SEQUENCE NUMBER 1-40, 1-55

SEQUENTIAL
- concatenate UFAS sequential files 8-1
- Sequential 1-2
- Sequential Format 1-8
- UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1

SERIAL
- FILE SERIAL NUMBER 1-40, 1-55
- NEXT VOLUME SERIAL NUMBER 1-25, 1-30.1
- REEL SERIAL NUMBER 1-43
- TAPE REEL SERIAL NUMBER 1-40, 1-55
- VOLUME SERIAL NUMBER 1-25

SET
- FILE SET IDENTIFIER 1-27

SINGLE
- Single File Input Processing 1-45, 1-46, 1-32.1

SIZE
- Control Interval Size A-7
- Index File Size A-8
- Volume Size 12-1

SPACE
- Space Assignment A-5

STANDARD
- LABEL STANDARD VERSION 1-25
- UFF And American National Standard Tape File Format 1-20

STATUS
- EXCEPTION CONDITIONS AND STATUS CODES 10-1

STATUS (cont)
- major status code 10-1
- PRINTER STATUS 11-2

SUBSTATUS
- substatus code 10-1

SYMREF
- Routine Package SYMREF Name 2-17, 3-10, 4-12, 5-11, 6-9

SYSTEM
- PAGE PRINTING SYSTEM (PPS) FORMAT AND PROCESSING 1-37.1
- SYSTEM CODE 1-28, 1-30.1
- System Input And Output Files 1-33

TAPE
- 2000/60 Magnetic Tape Interchange Facility 3-1
- Beginning of Tape Label (BTL) 1-39, 1-55
- Dedicated Tape 12-2, 12-8
- File Name Identified Tape 12-1
- FILE NAME IDENTIFIED TAPE (FNIT) 12-2
- H2000 Tape File Format 1-52
- H2000/60 Magnetic Tape Interchange Facility 1-52
- IBM Tape File Format 1-46
- MAGNETIC TAPE DENSITIES A-1
- PARTIAL TAPE LABEL (PTL) 1-43
- PHYSICAL ATTRIBUTES (Tape Files) 1-35
- RERUN AND TAPE UTILITIES 12-1
- TAPE DENSITY 1-41
- TAPE REEL SERIAL NUMBER 1-40, 1-55
- UFF And American National Standard Tape File Format 1-20
- UNLABELED GFRC TAPE FILES 1-44
- Unlabeled tape file 8-1
- Unlabeled Tape Files 1-58
- UNLABELED UFF TAPE FILES 1-32
- UNLT TAPE FILE FORMAT 1-45

TAPES
- CHECKPOINT RECORDS ON DATA TAPES A-3

TIMESHARING
- UFAS Directories For DM-IV (Batch Or Timesharing) 7-3

TRACKING
- Checkpoint Record Tracking 12-2

UFAS
- concatenate UFAS sequential files 8-1
- UFAS Directories For DM-IV (Batch Or Timesharing) 7-3
- UFAS Directories For DM-IV-TP 7-3
- UFAS IDENTIFIER 1-40, 1-43

UFF
- UFF And American National Standard Tape File Format 1-20

UFF (cont)
UFF, ANS, GFRC, AND IBM SEQUENTIAL FILE ORGANIZATION 2-1
UNLABELED UFF FILE PROCESSING 1-32
UNLABELED UFF TAPE FILES 1-32
USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4

UFSRET
Return Macro - UFSRET 2-37, 3-19, 4-31, 5-29, 6-22

UNIFIED
Unified File Formats 1-5

UNLABELED
UNLABELED GFRC FILE PROCESSING 1-45
UNLABELED GFRC TAPE FILES 1-44
Unlabeled tape file 8-1
Unlabeled Tape Files 1-58
UNLABELED UFF FILE PROCESSING 1-32
UNLABELED UFF TAPE FILES 1-32
UNLABELED UNLT FILE PROCESSING 1-46

UNLT
UNLABELED UNLT FILE PROCESSING 1-46
UNLT TAPE FILE FORMAT 1-45

USER
Read User Labels - DLRLB 2-34
User External Labels 12-1
USER LABEL PROCESSING (For ANS, UFF And IBM Files) 1-4
USER WORDS 1-41, 1-42
Write User Labels - DLWLB 2-35

UTILITIES
RERUN AND TAPE UTILITIES 12-1

VARIABLE-LENGTH
Variable-Length Records 1-36

VERSION
GENERATION VERSION NUMBER 1-28
LABEL STANDARD VERSION 1-25

VOL1
Volume-Header Label (VOL1) 1-25, 1-48

VOLUME
Defective Volume Replacement 12-1
NEXT VOLUME SERIAL NUMBER 1-25, 1-30.1
VOLUME SERIAL NUMBER 1-25
Volume Size 12-1

VOLUME-HEADER
Volume-Header Label (VOL1) 1-25, 1-48

WORD
Control Word Format 1-37.1
PPS Block control Word 1-37.1
PPS Record Control Word 1-37.2

WORDS
USER WORDS 1-41, 1-42

WRAP-UP
Wrap-up Macro - DWRAP 2-43, 3-20, 4-39, 5-36, 6-28

WRITE
Write User Labels - DLWLB 2-35

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	SERIES 60 (LEVEL 66)/6000 UNIFIED FILE ACCESS SYSTEM (UFAS) ADDENDUM A	ORDER NO.	DC89-03A
		DATED	JULY 1980

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. □

FROM: NAME ______________________ DATE ______________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 66)/6000
UNIFIED FILE ACCESS SYSTEM (UFAS)

ORDER NO. DC89-03

DATED AUGUST 1979

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. □

FROM: NAME ______________________ DATE ______________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell