Burroughs

B 7000/B 6000 Series

I/O SUBSYSTEM

REFERENCE MANUAL

(RELATIVE TO MARK 3.2 RELEASE)

Copyright © 1981, Burroughs Corporation, Detroit, Michigan 48232

PRICED ITEM

Printed in U. S. America July 1981 5001779

Burroughs believes that the software described in this document is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, is accepted for any consequences arising out of use of this software material, including loss of profit, Indirect, special, or consequential damages. There are no warranties other than those expressly set forth in the PROGRAM PRODUCTS LICENSE AND SERVICE AGREEMENT.

The Customer should exercise care to assure that use of the software material will be in full compliance with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued from time to time to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded using the Remarks form at the back of the manual, or may be addressed directly to TIO West Documentation, Burroughs Corporation, P.O. Box 4040, El Monte, California 91734, U.S.A.

TABLE OF CONTENTS

INTRODUCTION 1- 1

SCOPE OF MANUAL 1- 1

BACKGROUND. 1- 1

OBJECTIVES OF THE INPUT-OUTPUT SUBSYSTEM. 1- 5

Transparent to User. 1- 5

Separate User From Peripheral Detail 1- 5

Device Independence. 1- 5

Device Generality. 1- 5

Conversation of Program and Program Skills 1- 5

Language-General Standard Conventions for System Issues. 1- 5

Automatic and Dynamic System Balancing 1- 6

Easy Tunability. 1- 6

Use of Defaults to Make Tuning Non-Essential 1- 6

Minimum Operator Dependence. 1- 6

SOME FUNDAMENTAL CONCEPTS 1- 7

GENERAL STATEMENTS CONCERNING FILE ATTRIBUTES 1- 9

Attributes Used for Identification 1- 10

ATTRIBUTES USED FOR FILE STRUCTURING. 1- 11

ATTRIBUTES GIVING STATUS OF PERMANENT FILES 1- 11

Diagnostic Attributes. 1- 11

ATTRIBUTES USED FOR ACCESS SECURITY 1- 12

ATTRIBUTES USED FOR TRANSLATION 1- 12

PHYSICAL FILES AND RELATED ATTRIBUTES. 2- 1

ATTRIBUTES BY KIND OF PERIPHERAL. 2- 1

CARD FILES. 2- 1

CONSOLE FILES 2- 2

DATA COMM FILES 2- 2

DISK FILES. 2- 4

LINE PRINTER FILES. 2- 6

MAGNETIC TAPE FILES 2- 7

PAPER TAPE FILES 2- 7
Reverse and Backspace Operations 2- 8
Special Hardware Features 2- 8
Input Operations 2- 8
Output Operations 2- 8
PORT FILES 2- 12
Port File Input/Output Operations 2- 12
FILE ATTRIBUTES 3- 1
INTRODUCTION 3- 1
DESCRIPTION OF FILE ATTRIBUTES 3- 2
APL 3- 11
AREAALLOCATED 3- 11
AREACLASS 3- 11
AREALENGTH 3- 12
AREAS 3- 13
AREASIZE 3- 13
ASSIGNTIME 3- 13
ATTERR 3- 14
ATTVALUE 3- 14
ATTYPE 3- 14
AVAILABLE 3- 14
BACKUPKIND 3- 15
BLOCK 3- 16
BLOCKSIZE 3- 16
BLOCKSTRUCTURE 3- 17
BUFFERS 3- 18
CARRIAGECONTROL 3- 18
CENSUS 3- 20
CHANGEDSUBFILE 3- 20
CHANGEEVENT 3- 20
CONTROLMESSAGE 3- 20
COPIES 3- 20
COPYNAME 3- 21

CREATIONDATE 3- 21
CRUNCHED 3- 21
CURRENTBLOCK 3- 22
CURRENTRECORD 3- 22
CYCLE 3- 22
CYLINDERMODE 3- 22
DATE 3- 23
DENSITY 3- 23
DEPENDENTSPECS 3- 23
DIRECTION 3- 24
DISPOSITION 3- 24
DUPLICATED 3- 25
ENABLEINPUT 3- 25
EOF 3- 25
ERRORTYPE 3- 25
EXCLUSIVE 3- 26
EXTMODE 3- 26
FAMILY 3- 27
FAMILYINDEX 3- 27
FAMILYNAME 3- 27
FAMILYSIZE 3- 28
FCM 3- 28
FILEKIND 3- 29
FILENAME 3- 30
FILEORGANIZATION 3- 31
FILESECTION 3- 31
FILESTATE 3- 31
FILETYPE 3- 33
FILEUSE 3- 35
FLEXIBLE 3- 35
FORMID 3- 36
FORMMESSAGE 3- 36
FRAMESIZE 3- 36

GENERATION 3- 37
HOSTNAME 3- 37
IAD. 3- 37
IMAGEFILEID. 3- 37
IMAGEFILEMAP 3- 37
INPUTEVENT 3- 38
INPUTTABLE 3- 38
INTERCHANGE. 3- 38
INTMODE. 3- 39
INTNAME. 3- 39
IOCLOCKS 3- 40
IOINERROR. 3- 40
KIND 3- 40
LABEL. 3- 41
LABELTYPE. 3- 41
LASTRECORD 3- 41
LASTSTATION. 3- 42
LASTSUBFILE. 3- 42
LINENUM. 3- 42
MAXCENSUS. 3- 43
MAXRECSIZE 3- 44
MAXSUBFILES. 3- 45
MINRECSIZE 3- 45
MYNAME 3- 45
MYUSE. 3- 46
NEWFILE. 3- 47
NORESOURCEWAIT 3- 47
OPEN 3- 48
OPTIONAL 3- 48
OUTPUTEVENT. 3- 48
OUTPUTTABLE. 3- 48
PACKNAME 3- 49
PAGE 3- 49

PAGESIZE 3- 49
PARITY 3- 50
POPULATION 3- 50
PRESENT. 3- 50
PROTECTION 3- 51
RECEPTIONS 3- 51
RECORD 3- 52
RECORDINERROR. 3- 52
RECOVERYFILEID 3- 52
RECOVERYINDEX. 3- 52
REEL 3- 52
REMOTEID 3- 52
RESIDENT 3- 53
ROWADDRESS 3- 53
ROWSINUSE. 3- 53
RSERROR. 3- 54
RSEXTENDIBLE 3- 54
RSRESTART. 3- 54
RSRESULT 3- 55
RSUNITNO 3- 55
SAVEFACTOR 3- 55
SCREEN 3- 56
SCREENSIZE 3- 56
SECURITYGUARD. 3- 56
SECURITYTYPE 3- 57
SECURITYUSE. 3- 57
SENSITIVEDATA. 3- 58
SERIALNO 3- 58
SINGLEPACK 3- 59
SINGLEUNIT 3- 59
SIZEMODE 3- 60
SIZEOFFSET 3- 60
SIZE2. . . . . 3- 60

SORTFILEID 3- 60
SORTTABLECODE. 3- 61
SPEED. 3- 61
STATE. 3- 61
STATIONCOUNT 3- 63
STATIONLIST. 3- 63
STATIONNAME. 3- 64
STATIONSDENIED 3- 64
SUBFILEERROR 3- 64
TANKING. 3- 65
TAPEREELRECORD 3- 65
TIMELIMIT. 3- 66
TITLE. 3- 66
TRAINID. 3- 67
TRANSLATE. 3- 68
TRANSLATING. 3- 69
TRANSMISSIONNO 3- 69
TRANSMISSIONO. 3- 69
TRANSMISSIONS. 3- 69
UCRID. 3- 69
UNITNO 3- 70
UNITS. 3- 70
UPDATEFILE 3- 70
USECATALOG 3- 71
USEDATE. 3- 71
VERSION. 3- 71
WIDTH. 3- 71
YOURNAME 3- 72
YOURUSERCODE 3- 72
DIRECT I/O BUFFER ATTRIBUTES. 3- 73
IOADDRESS. 3- 73
IOCANCEL 3- 73
IOCHARACTERS 3- 73

IOCOMPLETE 3- 74
IOCW 3- 74
IOEOF 3- 74
IOERRORTYPE 3- 74
IOMASK 3- 75
IOPENDING 3- 75
IORECORDNUM 3- 75
IORESULT 3- 75
IOTIME 3- 76
IOWORDS 3- 76
SPECIAL TOPICS 4- 1
FILE ASSIGNMENT 4- 1
New File vs. Permanent File 4- 1
Selecting a Peripheral for a New File 4- 3
Finding a Permanent File 4- 4
Interactions of the Myuse and Security Attributes 4- 4
Restrictions Imposed by Peripheral Association 4- 4
Port Subfile Assignment 4- 5
Circumstances Requiring Operator Intervention 4- 7
SOFTWARE TRANSLATION 4- 7
DUPLICATED FILES 4- 10
Directory Structure 4- 11
LIBRARY/MAINTENANCE Facilities 4- 11
Logical I/O 4- 12
BUFFER HANDLING 4- 12
PHYSICAL WRITE 4- 12
PHYSICAL READ 4- 12
RESULT DESCRIPTOR(S) 4- 12
File Restoration/Reconstruction 4- 14
IRRECOVERABLE READ PARITY ERROR 4- 14
IRRECOVERABLE FAILURES WORSE THAN READ PARITY 4- 14
DISK FILE DATA RECOVERY 4- 16
Introduction 4- 16

Disk File Subsystem Failure Modes 4- 16
SIMPLE (RECOVERY) READ PARITY ERRORS 4- 17
HEAD ELECTRONICS FAILURES 4- 17
HEAD CRASHES 4- 17
STORAGE UNIT FAILURES (STORAGE UNIT NOT READY) 4- 17
OTHER FAILURES 4- 17
Data File Recovery Units 4- 18
Dumps and Audit Trails 4- 18
Normal I/O Failure Detection 4- 19
DETECTION THAT AN ERROR HAS OCCURRED 4- 19
DETERMINATION OF WHICH RECORD IS IN ERROR 4- 20
DETERMINATION OF WHICH ERROR CONDITION EXISTS 4- 20
DETERMINATION OF WHICH ROW IS IN ERROR 4- 20
EMPLOYMENT OF SOFTWARE RESULT DESCRIPTOR 4- 21
Direct I/O Failure Detection 4- 21
DETERMINATION OF WHICH RECORD IS IN ERROR 4- 21
DETERMINATION OF WHICH ERROR CONDITION EXISTS 4- 21
DETERMINATION OF WHICH ROW IS IN ERROR 4- 22
Row Reconstruction 4- 22
Data Recovery Techniques 4- 23
DISK FILE AND SYSTEM-ACCESS SECURITY 4- 25
Introduction 4- 25
Usercodes and Passwords 4- 25
ASSIGNMENT OF USERCODES 4- 25
CONTROL OF PASSWORDS 4- 26
ACCESSCODES AND PASSWORDS 4- 26
System Access 4- 26
JOBS 4- 26
DATA COMM 4- 27
PROGRAMMATIC 4- 27
ZIP 4- 27
File-Naming Conventions 4- 27
FILENAME 4- 27

FAMILYNAME. 4- 28
File Access: Security Attributes. 4- 29
SECURITYGUARD ATTRIBUTE 4- 29
SECURITYTYPE ATTRIBUTE. 4- 29
SECURITYUSE ATTRIBUTE 4- 30
Attribute Interactions 4- 30
Attribute Specifications 4- 30
Library Maintenance. 4- 31
Privileged Users 4- 31
Security Violations. 4- 31
FORMAT OF EXTERNAL FILE NAMES. . . . . A- 1
FORMAT OF LIBRARY TAPES. . . . . B- 1
FORMAT OF PACK LABEL . . . . . C- 1
FORMAT OF INTERCHANGE DIRECTORY RECORD . . . . . D- 1
FORMAT OF INTERCHANGE FILE HEADER. . . . . E- 1
NAME CONVENTIONS FOR DISK PACKS. . . . . F- 1
STANDARD TAPE LABEL FORMATS. . . . . G- 1

INPUT-OUTPUT SUBSYSTEM INTRODUCTION

SCOPE OF MANUAL

This manual describes the Input-Output Subsystem of the B 7000/B 6000 Information Processing System. The Input-Output Subsystem includes those parts of the Master Control Program (MCP) and other system facilities that are primarily concerned with physical and logical input-output operations. In particular, the manual deals with information viewed as files, and also with the control of those files by means of file attributes.

The manual serves both as an introduction to system policy and conventions dealing with input-output, and also as a reference to attributes and formats that are occasionally needed in controlling and tuning I/O operations. Therefore, parts of the manual are useful to newcomers approaching the system for the first time, and other parts are useful to experienced systems programmers who are tuning the system for some special use.

The B 7000/B 6000 is a sophisticated system, but sophistication is not demanded of its users. A major design objective was ease of use, and there was a great effort to make the system provide efficient operation where possible, while making minimum demands on the user. This is particularly so in input-output operations. The sophistication is internal to the system, that is, in the logic of its integrated hardware and software. The user interface is simple.

A few assumptions are made about the background of the reader of the expository parts of this manual. These assumptions include introductory knowledge of at least one programming language, minimal experience in using it, and a casual knowledge of the major issues involved in the effective and efficient use of a computer.

The information in the manual is divided into four major areas: Introduction to System I/O Conventions, Descriptions of Physical Files and Related Attributes, Detailed Descriptions of the File Attributes, and descriptions of Selected Special Topics. In addition, formats of various files and labels are presented in the Appendix. All of the material is user-oriented; there is no internal description of MCP procedures. The newcomer and most users can safely restrict themselves to the introductory section.

BACKGROUND

This section briefly reviews some salient factors in the evolution of Input-Output Subsystems. This review provides the historical context in which the Input-Output Subsystem is meaningful, and in terms of which it can best be understood.

From the beginning of computer usage, the control of input-output devices presented a particularly difficult problem. First there was the machine-language problem of addressing and communicating with a group of strange and varying devices. For purposes of this manual, this is termed the complexity problem. Then there was the problem, termed the assignment problem, of making the right connection between the program and the particular file of data the program intended to deal with. There was also the need to synchronize the peripheral device and its transfers of data with the operations the program was performing on that data. This was a problem of synchronizing concurrent processes, but its most prominent feature was (and remains) that the I/O data transfer operation is usually very much slower than the internal data manipulation operations on the transferred data. Both designers and users

searched for ways to compensate for that mismatch of speed. In this manual this will be termed the efficiency problem. The aforementioned are not all the problems associated with I/O, but they are prominent ones.

The complexity problem was difficult, even with simple systems. As time went on, the size and variety of peripheral devices attached to computers became so complex, that the user had to have help to make the system at all manageable. Without higher-level languages, system conventions, and automatic system operations, the size of a manageable job would be severely limited.

These problems were first encountered in simple systems of cards, tapes and printers. Magnetic tape was the first sophisticated peripheral, and sequential file maintenance was the prototype file-oriented operation. Original solutions were first conceived in that content. Original language constructs (especially in COBOL) still carry some of the serial tape flavor.

The assignment problem involved the operator. In the most primitive systems, it required at least that the operator select the right physical file, e.g., the correct reel of tape, and mount it on the tape unit addressed by the program. The operator's actions also became a bottleneck. Sometimes it took longer to set up a task (with the system idle and waiting) than it did to run the task.

The idea then evolved of taking the name or other identifier of the file such as was written on the physical label affixed to the outside of the reel, and also recording it magnetically as the first record on the tape. At first this was used as a check to ensure the program that the desired file had in fact been assigned to it. On the B 5000 and later systems, the label was used to make the assignment automatic; that is, the operator merely mounted the reel on an available tape unit. This could be done before the reel was needed, and while unrelated programs were running. The MCP would read the label and identify the file. When it had a request for that file it would automatically make the assignment; that is, establish the connection between the requested physical file and the program requesting it. This provided a large measure of "device independence," that is, the programmer could think in terms of a logical file, and need have no concern about addressing a particular physical unit. The physical connection is made at run time by the MCP.

Other information besides file names came to be stored in labels and other markers, and more system aid dealing with them was made available to programmers, e.g., automatic detection of the end of a file on reading, or automatic bridging to an additional reel if the end of one reel was reached while writing a file. The accumulation of such system aids made peripheral programming tolerable. Gradually, an advanced state of device independence was reached in which the programmer could largely ignore a peripheral, except for those aspects that he chose to see. The problems of complexity were gradually removed from the user and resolved by the system.

Facing the mismatch between peripherals and processors, both system designers and application programmers sought solutions to the efficiency problem. One of the most basic and popular solutions is blocking of records. If a peripheral operation takes so long compared with the processing of the transferred records, why not transfer several at once? One of the difficulties was the additional complexity a program needed to unpack the records, one by one, after the block was read, or pack them, one by one, into blocks as they were written. This complexity also migrated into the system, thus aiding the application programmer, who merely had to invoke blocking of the size he wanted.

Another efficiency device is buffering. In this technique, instead of using only one memory area as the destination (in the example of a read) for a peripheral transfer, the system uses two or more, so that processing or unpacking operations can be underway in one while another peripheral transfer is taking place into another. Once again, the programming complexity of

maintaining and rotating the buffers and calling for the physical I/O operations when needed is assumed by the system. The programmer is free to deal with his file one logical record at a time, unmindful of the complexity of the system.

Peripherals rapidly evolved in size, variety, and complexity. One of the most important additions was the system store. System stores using disks or drums have been an integral part of all Burroughs multiprogramming systems since the B5000. Here many files, used and perhaps shared by many programs, were stored on a set of surfaces whose storage capacity they shared. In this environment the assignment of a stored file to a program was no longer a simple matter of causing the program to address the correct physical unit. Many files shared the same storage facility, and it was neither practical nor desirable to make file assignments without system intervention and aid. It became a system task to efficiently allocate space on a group of disks, considered as a system resource. It became a system responsibility to make associations between requesting programs and files stored on that system resource. So B 5000 and later systems maintained a system-wide disk directory of all files stored on this system resource.

The disk is also a heavily used part of a hierarchical memory. Historically, hierarchical memories have been most visible to application programmers who had to explicitly manage overlay transfers between main memory and backup stores. In the B 7000/B 6000 systems, these transfers are automatic system functions and are transparent to the user. They are an inherent part of the deeply imbedded virtual memory, and are not considered as part of the Input-Output Subsystem visible to the user.

Disk packs also evolved as popular media for file storage. Packs are used as a shared system resource similar to fixed, head-per-track disks. Because they are removable, however, they can also be used for somewhat more exclusive file storage.

Using magnetic tape, at any instant there is only one block that is convenient to read, the next one to come under the head. Tape programs were therefore heavily biased toward a sequential pattern of access, and both files and programs were organized with this in mind. With head-per-track disk it is possible to select any one of a very large number of heads, so any bit of data is accessible within one rotation. This device invites random patterns of access, and allows files and programs using it to be more flexibly structured. Disk packs have a smaller set of heads, which must be moved to select one of many cylinders, or sets of tracks. This creates pressure to organize files and programs to remain within a cylinder once it has been selected; apart from this fact, packs also allow random patterns of access. Various file structuring and accessing techniques evolved to reconcile the modes of storage and their limitations with the evolving needs of applications. At the same time, the increased ability to share files required some protection, and techniques of file security were developed. Access and security facilities also tended to filter down into operating systems and become system functions.

To meet the most demanding needs for file sharing and complex or novel file structures, data base management systems have evolved. The implementation of the Data Management System has been closely integrated with the MCP, and can be considered as an extension of the facilities of the Input-Output Subsystem for users with special needs. It is, however, beyond the scope of this manual.

As more and more critical data and operations were entrusted to computer systems, dependability of systems became more important. The integrating of stored data and the ability to recover gracefully when a failure occurred became especially important. These issues have strong impact on an Input-Output Subsystem.

The advent of data communication networks connected to computer systems over data transmission lines introduced radically new capabilities and modes of use. It also vastly increased the size, variety, and complexity of the I/O networks connected to computers, and the potential problems of I/O programmers. This presented a challenge to operating systems once again. This time the challenge was to separate the data communications programmer (already burdened with applicational problems) from the complexity of data communication networks, and also to gracefully reconcile data communications networks with other systems I/O. In the B 7000/B 6000, part of this task is accomplished in the Input-Output Subsystem.

OBJECTIVES OF THE INPUT-OUTPUT SUBSYSTEM

This section presents some important points of policy of the Input-Output Subsystem. The system can best be understood and used with these principles in mind. Some of these principles are general to the MCP and the system, but apply in particular to the Input-Output Subsystem.

Transparent to User

The Input-Output Subsystem is intended to be transparent to the user program. The user sees a simple, familiar, and convenient user interface. In most cases he need have no concern about the detailed internal operation of the Input-Output Subsystem; therefore, little of that detail is included in this manual.

Separate User From Peripheral Detail

Users may ignore the details in peripheral device control. The programmer is required to see only the factors of real significance. The user is able to deal with these factors using a high-level language.

Device Independence

The programmer need not be concerned with connections to peripheral units. Programs are not limited to a particular configuration or bound to any hardware address until run-time assignment is made.

Device Generality

To the extent that it is reasonable, the differences among the various kinds of peripherals are kept within the Input-Output Subsystem. The programmer can think in terms of a logical file, largely independent of the kind of peripheral used. The programmer or the system may change the assignment of that logical file from one kind of peripheral device to another with minimum impact on the program.

Conversation of Program and Program Skills

The Input-Output Subsystem is designed to accommodate high-level languages, popular data organization techniques, and current programming conventions. It minimizes conversion requirements for existing programs in high-level languages. It also permits changes to the system configuration without requiring changes in existing programs.

Language-General Standard Conventions for System Issues

The style and requirements of I/O statements for programming languages are observed. System-wide issues regarding efficiency and file-handling conventions in a multi-language, multi-programming environment are resolved in a common way for all programs.

Automatic and Dynamic System Balancing

Certain issues of I/O efficiency can only be resolved by the system, because only it has the necessary information. The system takes responsibility for such issues, thus reducing the burden on individual programs.

Easy Tunability

There are some areas in which a programmer who knows the characteristics of his particular program, or a site manager who knows the characteristics of his task environment, can tune better than the system can. The ability to tune in important areas, with minimum burden on the user is intended. The system is tuned by using simple parameters.

Use of Defaults to Make Tuning Non-Essential

The system is designed to operate efficiently with minimum tuning.

Minimum Operator Dependence

The system avoids reliance upon the operator. When operator communication is necessary, it is done in the simplest, most efficient manner possible.

SOME FUNDAMENTAL CONCEPTS

This section defines some terms and discusses some concepts that are important for understanding this manual and using the Input-Output Subsystem.

A file is a group of related records. Files are of central importance in the Input-Output Subsystem, for most of the communication between programs and the Input-Output Subsystem concerns files.

A file is defined in the program dealing with that file, or at least the file description is known to that program. A prototype example is a file description in the data division of a COBOL program.

To a conventional program a file represents a large collection of ordered data that exists apart from the program. The program needs to interact with parts of that data from time to time, and arranges with the Input-Output Subsystem to have those made available, or sent back to parts of external storage when ready. The Input-Output Subsystem moves the data into and out of user working areas in main memory, to which the program has access.

The unit of data moved into and out of the user working area is the record. The record can contain strings of characters, groups of binary words, or both. Though a program typically deals with subdivisions of a record, and has a detailed record description, that further breakdown is, with few exceptions, of no concern to the Input-Output Subsystem, which deals with complete records. It gives the program one record at a time, and takes from it one record at a time.

A file has some structure as seen by the program. The records may be all of the same length, or they may be of varying lengths. Length information must be declared or implied by the program and made available to the Input-Output Subsystem. If the length is variable, the Input-Output Subsystem must have means of detecting the actual length of each record, so that it can pass one record at a time, regardless of length. From the program's viewpoint, records are ordered in some way. In the simplest case this may be seen as following each other sequentially until the end of the file. In that case the program is typically interested only in accessing the next record in sequence.

The file, as it is stored on some recording medium, is called a physical file. A physical file may have some additional elements of structure. First, it may contain blocks. A block is a group of physically adjacent records which are packaged together so that they can be transferred to or from the physical file as a group. Further, the storage device may impose structure on the physical file; for example, data is transferred to and from disk in units of one or more segments or sectors of 30 words (180 EBCDIC characters). Also, because disk can contain many files concurrently, blocks are grouped into larger units called areas. The term physical file is sometimes used to refer to the physical device rather than the data stored on its recording media. The context usually makes clear which meaning is intended.

The file and its structure as seen by the program is called a logical file. A logical file is a file variable declared within an object program. It exists only within that program, and from the viewpoint of block-structured languages, only within the program block where it is declared or to blocks where it has been passed as a formal parameter. A logical file has no inherent properties until it is described by file attributes, or until it is associated with a physical file. A physical file inherits properties from the file attributes of the logical file which creates it. Multiple logical files can be associated with one physical file, and the attributes of those logical files need not be identical in all cases.

The Input-Output Subsystem acts as an intermediary between the logical file and the physical file associated with it. The Input-Output Subsystem allows the program to view the data in terms of an organization of logical records and to handle them one at a time. The Input-Output Subsystem sees the logical file on the program side and the physical file on the peripheral side, reconciling the two as necessary. It deals with the program, one logical record at a time, and executes I/O operations only when necessary.

A logical file exists in one of four states: open, closed with release, closed with retention, and open-unassigned. Before data can be transferred between a logical and physical file, the logical file must be open and the physical file must be assigned to the logical file. This can be accomplished by explicitly opening the logical file, or via the AVAILABLE attribute. If the OPTIONAL attribute is TRUE for a logical file, opening the file can leave it unassigned to a physical file. Opening a file explicitly does not cause data to be transferred between the logical and physical files, and the logical file can be closed without any I/O being performed upon the file. The logical file can be closed with retention, which leaves the physical file assigned, or closed with release which severs the connection between the logical and physical files.

To expedite and smooth the flow of I/O operations, the Input-Output Subsystem makes use of buffers. A buffer is an intermediate storage area, under control of the Input-Output Subsystem, used to store data in transit between the physical file and the user work area, while that data is under the jurisdiction of the Input-Output Subsystem. Two buffers are typically used, so that one can be dedicated to a peripheral transfer while the other is available for logical record operations or user work area transfers. The data transfer into or out of the file can be either word or character-oriented.

One task of the Input-Output Subsystem is to make the file assignment; that is, to establish a connection between the logical file of a requesting program and the corresponding physical file.

Where a new file is to be created, assignment includes finding and providing storage space or providing the address of a peripheral device of the requested or acceptable kind; for example, connecting an available printer.

Both the logical file internal to the program and the external physical file have file names. The external file name, if the physical file is stored on disk, is recorded in a directory; if stored on tape, is recorded in one of the tape labels; and if stored on cards, is punched on a control card preceding the data cards. The Input-Output Subsystem maintains the disk directory, and also reads and keeps track of all label information on physical files loaded by the operator. When it creates a new file, as on tape, it automatically writes the necessary labels, including the file name and other label information.

A physical file can be either temporary or permanent. A temporary file is one that is assigned only at the time of its original creation, and is of no further interest to any program. Examples are printer files and disk files that are used only as an intermediate step in a process. Temporary files are private to the program that creates them, have no visibiliy to the general system, and exist for the Input-Output Subsystem only while the logical files which created them remain assigned.

Newly created files are considered temporary unless the program requests them to be saved or protected. To make a file permanent, the Input-Output Subsystem enters it into the directory (if on head-per-track disk or disk pack) or suitably marks its label (if on tape). A permanent file is visible to all running programs and to the system operator. Access to a permament file can be limited by security facilities. A permanent file remains visible until deliberately purged or, if on removable media, until the operator removes it from the system.

GENERAL STATEMENTS CONCERNING FILE ATTRIBUTES

This section introduces the central concept of file attributes, and the way they are used for the overall management of files and input-output.

Each programming language provides some means of describing files and manipulating input-output operations. When the standard language facilities are not adequate, they have to be extended in implementations. That was often the case as peripheral and data communication equipment grew in variety.

Each programming language has its own style to which programmers using that language must become accustomed. The style of file descriptions and input-output statements in COBOL, FORTRAN, BASIC, ALGOL and PL/I are quite different.

In a multi-language, multi-program environment with applications of any complexity, files can no longer be viewed as the simple property of a single program. In systems oriented toward data communications, time-sharing, or data base management, files tend to become system components accessed by a number of programs. Those programs may be written in different languages. It is convenient to be able to manage files in a system-wide, global, language-general way. This was first done in the language of control cards, used for overall system control. Later it evolved into part of the Work Flow Language. The control of files and input-output is based on the use of file attributes.

This does not mean that the programmer writing in COBOL, for example, must learn a new language and rewrite his COBOL programs. The COBOL Compiler accepts programs in standard COBOL and makes the translation to file attributes. It does mean, however, that there is additional control and flexibility that the programmer can exert at run time in the language of the system.

The idea of attributes has been evolving for several years, particularly in Burroughs Extended ALGOL and in PL/I. File attributes are system control parameters used by the Input-Output Subsystem, containing all the information it needs when it connects a physical file to the logical file of a program and regulates file access.

File attributes include, for example, basic information such as file names and other identification, the kind of peripheral unit involved, record-size and block-size information, and whether the file is to be used for input, output, or both.

Various programming languages provide ways of stating the most basic attribute information. The programmer provides it in the conventional way. If he does not provide the information (where its omission is legal in the language used) the system provides the most reasonable defaults.

The Input-Output Subsystem allows dynamic file definition. This means that the precise nature of all attributes of a file need not be known at the time the program is written. Further, some attributes, even if defined in the program, can be changed.

To use this capability, the programmer must be familiar with the attributes available and also how to use them. An alphabetical reference list of file attributes is given later in this manual. Many of the attributes are not used by the typical programmer. An introduction to the classes and uses of attributes, and a brief explanation of the most commonly used attributes are included in this section.

Attribute values can be explicitly set in the various languages. This is done by equating an acceptable value to one of the attribute names. The syntax for doing this varies depending on the style of the individual language, and is given in the information manuals for the particular language.

File attributes can be changed at compile-time or program execution time by the use of file attribute assignment statements in the Work Flow Language. The specific syntax for the file attribute assignment statements is given in the B 5000/B 6000/B 7000 Series Work Flow Language Reference Manual.

Finally, some file attributes can be changed during program execution. While this can occur through Work Flow Language Statements, it is more typically done by statements in the running program that set file attributes.

Besides the file attributes used directly by the Input-Output Subsystem for file and input-output control, there is another class used for synchronization and intercommunication between running programs, or between running programs and the Input-Output Subsystem. These are maintained by the Input-Output Subsystem and typically are readable-only by user programs.

Some combinations of attributes are reasonable; others are not. The Input-Output Subsystem checks attributes for inconsistency. In general, it disallows combinations that are self-contradictory or that logically indicate errors, but otherwise it is permissive. Most attribute actions that are not clearly in error are permitted.

Attributes Used for Identification

The external name of the file is accessed through the use of the FILENAME and TITLE attributes. Every file also has an internal name, the name used to declare the file variable in the program (that also is an attribute, INTNAME, and can be changed). The Input-Output Subsystem uses INTNAME for FILENAME, TITLE, and INTNAME, when the FILENAME and TITLE are unspecified. The FILENAME and TITLE attributes are used to identify files on peripheral devices.

In a file maintenance situation, the name alone is not always enough, because updated versions of the file may be made and saved, using the same name. The system can keep track of the genealogy of an updated family of files, using the ANSI standard techniques of recording CYCLE and VERSION in the label or directory.

The CYCLE and VERSION attributes exist to help distinguish between different iterations of a file. Different iterations of a file that have the same CYCLE value are of the same genealogy. The best genealogy of a file is the iteration with the highest CYCLE value and the highest VERSION value within that CYCLE. The cataloging system defines generations as an absolute ordering among the genealogies of a file which takes into consideration CYCLE, VERSION, and the time of last update. CYCLE and VERSION are settable attributes, so the user can request or create a specific CYCLE and VERSION. When the file is on disk (head-per-track or pack), the name of the family (as distinguished from the name of the file) can also be given by assigning the name to the attribute FAMILYNAME.

One of the most important attributes is KIND, which represents a hardware device.

TEXT DELETED

The Input-Output Subsystem depends on the disk directory, or labels to identify permanent files. Tape, paper tape, and card files can be declared to be unlabeled files by setting an attribute LABEL=OMITTED or OMITTEDEOF. This lack of label records makes it impossible for the system to automatically associate

a physical file with the logical file. This means that the system operator must assign the physical file by system message dialog with the Input-Output Subsystem at the time the logical file is opened.

ATTRIBUTES USED FOR FILE STRUCTURING

The attributes MINRECSIZE, MAXRECSIZE, and BLOCKSIZE determine the lengths of the records and blocks of a file. The values for MINRECSIZE, MAXRECSIZE, and BLOCKSIZE are specified in units called frames. The attributes FRAMESIZE (or UNITS) and INTMODE determine the number of bits in each frame. Refer to the discussions of these attributes in Section 3 of this manual for more specific information.

Records can be either fixed length or variable length. If variable, the allowable limits of record size are given in two attributes, MINRECSIZE and MAXRECSIZE. For fixed length records both have the same value.

The attribute BLOCKSTRUCTURE (or FILETYPE) indicates which of several blocking techniques is to be used. For certain variable length blocking techniques, the attributes SIZEMODE, SIZEOFFSET, and SIZE2 are also needed.

An attribute, FILEKIND, allows the classification of several different kinds of files for programs that require information about the internal structure of the file and its records.

ATTRIBUTES GIVING STATUS OF PERMANENT FILES

The existence of a permanent file or the availability of a physical device may be a requirement before a logical file can be opened. Three attributes, AVAILABLE, RESIDENT, and PRESENT, allow user programs to inquire about the status of physical or permanent files in different ways, without requiring their presence where it is not needed.

Diagnostic Attributes

Not all combinations of attributes are valid; a number of attributes, for instance, are meaningful only for files stored on a certain KIND of peripheral. The Input-Output Subsystem makes consistency checks on attributes. Some of the information it gains can be useful to user programs, and is made available through read-only attributes. The ATTERR attribute indicates whether the most recent attribute action was in error. The values in the attributes ATTYPE and ATTVALUE help identify the problem.

The Input-Output Subsystem does extensive checking on physical I/O operations, and translates error information into a logical result descriptor that is meaningful to the user. This logical result descriptor is returned for each logical I/O operation. The STATE attribute also can be used to access this information. The Input-Output Subsystem relates the error information as closely as possible to a logical record and passes this information by means of the attribute RECORDINERROR. The system also makes other file information available through read-only attributes such as RECORD, which returns the current position or record number in the file.

ATTRIBUTES USED FOR ACCESS SECURITY

Three attributes are related to system file security. SECURITYTYPE and SECURITYUSE indicate the level of security needed (PRIVATE, PUBLIC, GUARDED, or CONTROLLED) and how the file may be used. SECURITYGUARD gives the name of the guardfile if GUARDED or CONTROLLED security is required. The security system is described in more detail later in this manual.

ATTRIBUTES USED FOR TRANSLATION

The Input-Output Subsystem is capable of providing automatic character translation when necessary. The attribute INTMODE indicates the mode of data assumed within the program while EXTMODE indicates the mode existing or desired in the external physical file. The system allows user selected software translation by specifying translation tables, with the attributes INPUTTABLE and OUTPUTTABLE. The user can control the scope of hardware and software translation by use of the attribute TRANSLATE and can detect when software translation is taking place by means of the Boolean attribute TRANSLATING.

PHYSICAL FILES AND RELATED ATTRIBUTES

ATTRIBUTES BY KIND OF PERIPHERAL

Many attributes apply only to particular kinds of peripherals, and are best understood in terms of the use of that kind of peripheral. This section briefly introduces the attributes related to each device type, and includes additional user information concerning the device type.

CARD FILES

The KIND values associated with cards and their meanings are as follows:

READER Card reader.

PUNCH Card punch.

Card files have a physical restriction that they must map into an 80 or 96 column punch card. The external mode (EXTMODE) of card files is restricted to BCL, EBCDIC, and BINARY (DATA can be used as a synonym on the label record for EBCDIC). BINARY card files have 20-word physical records, twelve bits per column. The card codes for EBCDIC and ASCII are the same, so the lack of ASCII as an external mode is in fact no restriction.

The 80 EBCDIC characters on a card do not fit exactly into 14 words; however, the 80 BCL characters can be stored completely in 10 words. Because of the physical nature of card files, there is an upper bound for the size of the BLOCKSIZE attribute. The values are 80, 10, 14, and 20, respectively, for character-oriented, word-oriented BCL, word- oriented EBCDIC, and BINARY files. Larger values for BLOCKSIZE will be reduced and a run-time attribute error will be generated when the file is opened.

Card files are usually labeled (LABEL equal to STANDARD) files. The format of the beginning label record is as follows:

<I> <mode> <title option>

where <I> is an invalid character punch; <mode> is either BCL, EBCDIC, DATA, or BINARY; and <title option> is the external file name.

When a card file is included within a job deck, the <title option> may be empty. The nameless card file will be assigned to the first logical reader file opened by the job.

The ending label record has the following format:

<I> END

or BEND punched graphically, for the BINARY end card.

Unlabeled card reader files can only be opened with the assistance of the system operator. Only Direct I/O punch files can be unlabeled.

To open an unlabeled card reader file, the operator must use the UL system input message to assign the physical unit to the logical file, before cards (without the beginning label record) are put into the card reader.

CONSOLE FILES

A logical file can be assigned to a supervisory console that is attached to a single line control by setting the KIND attribute to SPO, thus allowing terminal communications to an object program without using Data Comm. The external mode (EXTMODE) of a console file is EBCDIC. An output (MYUSE equal to OUT) console file is assigned to a scratch unit, preferably to the originating unit, if the program is entered from a supervisory console. An input or input-output (MYUSE equal to IN or IO) console file is assigned to a labeled unit. A supervisory console can be labeled by typing the following:

LABEL <file-name> <etx>

where <etx> is the ETX character represented by the value hexadecimal 03. When the console file is opened, all input to the file must be preceded by the (DELTA) character represented by hexadecimal 1D. An input record is all the characters between the <delta> and the <etx>. If the record is less than MAXRECSIZE, the record is padded on the right with the blank fill.

The end of the input file is indicated by typing "<delta> ? END <etx>". Upon receipt of this record, the console file will take end-of-file action. A supervisory console will again become scratch when a labeled file assigned to it is closed. The CL system input message can also be used to unlabel a supervisory console. Because of the nature of the input records, the most natural way to describe the logical file is with FRAMESIZE equal to 8, and BLOCKSTRUCTURE equal to EXTERNAL. This is a character-oriented, variable-length console file. A request to read MAXRECSIZE characters causes the system to transfer the next record into the record area (array), blank fill to MAXRECSIZE if necessary, and return in the logical I/O result descriptor's size field (47:20) the exact number of characters in the record. Likewise, on output, only the number of characters in the record (as described in the write statement's size parameter) will be transferred to the supervisory console.

DATA COMM FILES

The KIND value of REMOTE is associated with any remote device attached to the system by way of a Network Support Processor or Data Communication Processor.

An object program receiving data from one or more remote devices, and sending data to those devices, regards that data as a file, just as it would in communicating with local peripherals. The Input-Output Subsystem provides for Data Comm files essentially the same degree of media independence which can be achieved by an object program dealing with other peripheral files. Unlike the usual peripheral devices which are accessed through peripheral control units, remote devices are accessed through a special peripheral processor called the Network Support Processor (NSP) or the Data Communication Processor (DCP). However, the user need not be concerned with that difference.

Data Comm files are permanent files in the sense that they are described in the Network Definition and are assigned an external name (TITLE) in the Network Information File (NIF). The Network Definition Language (NDL) allows a Data Comm file to be a single station (with the possibility of the external name of the file being the name of the station) or a family of stations. A family is a list of stations with a family name which are grouped together in the FILE section of an NDL program. The STATIONLIST attribute makes it possible to dynamically alter the list by adding or subtracting stations.

A Data Comm station is controlled by a Message Control System (MCS) program; the MCS for each station is specified in NDL. The MCS is responsible for logging on the user, handling error situations, assigning the station to

object-program files, and performing other functions required or desired by the MCS designers. Examples of MCS programs supported by Burroughs are SYSTEM/DIAGNOSTICMCS, SYSTEM/RJE (for remote job entry), SYSTEM/APL, and SYSTEM/CANDE (for file editing and task control); other MCS programs may be implemented at user installations. When an object program opens a file involving a Data Comm station, the MCS may elect to participate in I/O or not. By participating, the MCS will process each input-output message, and may selectively route each message to or from particular programs, files in programs, or particular stations. The MCS and the object programs must agree on a protocol so that the MCS can perform message switching and the program can identify source/destination stations. Without MCS participation, the messages are routed directly between the Data Comm subsystem and a program by the Input-Output Subsystem.

When a Data Comm file is opened, a station list is created in addition to the File Information Block (FIB). The station list contains the information from NIF which is necessary to distinguish the different stations which have been included in the file.

The file attributes that are uniquely related to Data Comm files reference information stored and maintained in a station list. There are attributes which relate to Data Comm files as an entity, such as STATIONCOUNT, LASTSUBFILE and CENSUS, and attributes which relate to the characteristics of the specific stations in the file, such as WIDTH, DISPOSITION, SCREEN and ASSIGNTIME. Some of the more general file attributes, such as TITLE, POPULATION and PAGESIZE, have special meaning when referencing a Data Comm file.

When the station list is created, each station in the file is assigned a Relative Station Number (RSN) in the order in which they are described in NIF. The RSN is, in effect, an index into the station list. A program may perform reads and writes without regarding the RSN and the subsystem will direct the output to the terminal sending the preceding input. The LASTSUBFILE attribute or the RSN used as an index alters this relationship. The RSN is not unique to the station and the ordering of the station list can be changed by extensive use of the STATIONLIST attribute. If a station is subtracted from a file, its RSN becomes invalid and points to an empty area in the station list. A station which is added to a file may be assigned an RSN which belonged to a station which was subtracted from the file, which is to say that RSN's are reusable. It is also possible for a valid RSN to be larger than the value of the STATIONCOUNT attribute, after some stations have been subtracted from the file.

If a Data Comm file is closed with retention using one of the following ALGOL statements:

"CLOSE (FILEID,REWIND)" , "FILEID.OPEN := FALSE", or "CLOSE (FILEID,*)"

or the regular close in COBOL, and then reopened, each station in the file will have the same Relative Station Number that it has at the time the file was closed. Any changes in the file's STATIONLIST (additions or subtractions) will also have been retained. If a Data Comm file is closed with release (the normal ALGOL close) and reopened, all previous modifications to the file's STATIONLIST are ignored, and it reverts to the STATIONLIST defined in NIF.

An object program may open an unrestricted number of Data Comm files, and a station may be assigned to any number of Data Comm files. However, a station can be assigned concurrently to only one Data Comm file that is input-capable, without the station's controlling MCS participating in the I/O operations. The object program file's MYUSE value is considered when opening a Data Comm file. For a Data Comm file, MYUSE equal to OUT means output only, but MYUSE equal to IN or IO allows both input and output.

This means that for a Data Comm file that is opened with a write statement (where the source language permits this) which is to be used for both input and output, MYUSE must be set to IO prior to the first I/O action. This can be done in the program or by compile-time or run-time label equation.

An object program can write to a Data Comm file in two different ways. If the LASTSUBFILE attribute is set to a Relative Station Number, the output will be directed to that specific station in the file. If the value of the LASTSUBFILE attribute is zero (the default value), the output will be broadcast to every station in the file. If there is only one station in the file, the result is the same as for a write directed to that station. In ALGOL, the LASTSUBFILE attribute can be set using the [STATION <arithmetic expression>] form of the <record number or carriage control> part of the write statement.

An object program receives input from a Data Comm file in a first-in first-out order. After a read statement, the LASTSUBFILE attribute contains the RSN of the station from which the input came. A seek statement does no more than change the value of LASTSUBFILE.

When an object program opens a Data Comm file, a FILE OPEN DATACOM message for each station in the file is sent to the station's controlling MCS. The FILE OPEN message is a notification to the MCS that the logical file requests permission to communicate with the station. The MCS may respond in one of three ways: to allow, postpone, or deny assignment of the station to the file.

If the MCS allows assignment, the program may proceed to use the station and file, unless the Data Comm subsystem overrules: a denial is forced if the MCS did not elect participation and either this station has no line assignment, or this assignment would result in more than one concurrent input-capable file for the station.

If the MCS denies assignment, which it may also do subsequent to having allowed or postponed assignment, an end-of-file message is placed in the input queue if the file is input-capable. Further writes to the station will cause end-of-file action.

A multi-station Data Comm file can have end-of-file notification for each station in the file. The LASTSUBFILE attribute can be interrogated to discover which station is no longer assigned to the file after end-of-file action. The station's DISPOSITION attribute or the qualification field of the STATE attribute can be accessed to find the reason for the end-of-file.

DISK FILES

The KIND values associated with disk and their meanings are:

DISK non-removable storage, head-per-track disk.

PACK removable or nonremovable storage, movable head disk-pack.

A disk file can be defined as the backup medium for printer or punch; for example:

```
BACKUPKIND = DISK
BACKUPKIND = PACK
```

To the logical file, "native" mode disk pack and head-per-track disk look exactly alike. Throughout this document explicit distinctions between disk pack and head-per-track files will be made where the two differ; when disk alone is referenced, both kinds will be implied.

The smallest addressable area of a head-per-track disk unit is a 30-word segment. The smallest contiguous area addressable on a disk pack is a 30-word sector. The Input-Output Subsystem must create an initial set of addresses on an uninitialized disk pack, which determines the sector layout before the pack can be used. This function is performed by the IV system input message.

Logical files associated with disk are not restricted in their values for the BLOCKSIZE attribute (or if the file is unblocked, the MAXRECSIZE attribute), but for the most efficient use of disk space the value should be a multiple of segment/sector size. Unblocked, variable length disk files waste space but do allow random access to the records.

Another element of storage structure imposed by the disk media is the area. Disk files can be very large, and can grow dynamically. In many cases, it would be impossible or wasteful to allocate enough disk space in advance to hold the complete file. An area is the amount of disk space that is allocated at one time, as the file is being created. By use of the AREALENGTH (or AREASIZE) attribute, the user can specify the length of an area. If neither AREALENGTH nor AREASIZE is specified, disk space is allocated in areas which are approximately 1000 logical records in length.

The user has control over the maximum amount of disk space allocated to a file by use of the attribute AREAS. This specifies the maximum number of areas to be assigned. If it is not specified, a default value of 20 is used. AREAS can be either an absolute limit or a flexible limit. If the Boolean attribute FLEXIBLE is true, the limit can be exceeded.

Crunching is a disk space conserving technique that allows truncation of the last allocated area for files (such as code files) which are known to be non-extendable. It can be invoked either explicitly or by system convention for certain classes of file.

TEXT DELETED

One technique provided by the system to enhance the integrity of disk files is that of duplicated files. Duplicated files are further defined later in this manual.

Disk files are labeled files, that is, the external file name and other information are stored along with the file. A physical disk file becomes a permanent file, visible to other programs, when it is entered into the directory. Typically, this is done when the file is closed after creation by a statement in the program such as LOCK or CLOSE WITH CRUNCH. A file can also be a permanent file from the time it is originally opened by means of the attribute PROTECTION. This attribute can also cause file creation to be done in such a way that chances of recovering the whole file in the event of system failure during creation are increased.

Disk units are labeled devices that are organized into families with all the units having a common family name. The association of disk storage devices into families has only one restriction: the members of a family must be logically equivalent devices. Disk storage devices are considered logically equivalent when they have a common file and directory structure and where the segment (or sector) is of the same length and is addressed in the same manner.

When a disk unit is initialized, reconfigured, or labeled by the IV, RC, or LB system input messages; label records in the ANSI (USASI) volume and header tape label formats are written on the unit. The family name is a single identifier of up to 17 characters and it may be used as an identification check by means of the attribute FAMILYNAME. When a unit is mounted, or made ready, the label records are read and analyzed. If the disk drive is not "write enabled", the unit is marked as available for input only. If the unit has the same serial

number as another unit currently on-line, the new unit will be marked "DUP SN" and will not be made available for use.

A family's directory, containing file headers, and residing on a base unit is completely sufficient for all permanent SINGLEUNIT disk files which reside on that unit. For files spread over more than one member of the family, the file header contains references (the FAMILYINDEX) to the other family members that contain areas assigned to the file.

Space allocation for disk files is dependent upon the CYLINDERMODE (pack only) and the SINGLEUNIT attributes. When CYLINDERMODE is TRUE, areas are always allocated beginning at cylinder boundaries. For multi-unit files (SINGLEUNIT is FALSE), areas are allocated on a rotational basis throughout the family.

The logic used for initiating physical I/O operations to disk pack files is the same as used for all other types (KINDs). The seek statement does not cause arm movement of the disk pack drive unless the normal logic requires that a buffer be filled and a physical read initiated, preceded if necessary by the rewriting of the "top" buffer back into the file.

Files are organized upon interchange disk packs differently than upon native mode disk pack or head-per-track disk. The pack directory and file header layouts are designed for compatibility among various Burroughs systems. This mode may be selected on creation of pack files by use of the attribute INTERCHANGE; otherwise, the native format will be used. The interchange compatibility includes some restrictions on the files residing upon interchange packs. The external file name (TITLE) of a file residing on an interchange pack must be a single identifier, up to eight characters in length (if the TITLE is longer, it will be truncated). If the TITLE of the logical file consists of more than one identifier, then, as in the case of tape files, only the first and last identifiers are used. The first identifier (MFID) is used to identify the pack name of the interchange pack unless the FAMILYNAME attribute has also been initialized. In this case, the MFID is completely ignored.

LINE PRINTER FILES

The KIND value associated with line printer is PRINTER. The Input-Output Subsystem recognizes both EBCDIC and BCL drum printers and various models of train printers. The attribute EXTMODE can be used to specify a desired type of drum printer and the TRAINID attribute is used to select a train printer. The Input-Output Subsystem will accommodate to whichever type is assigned and take the necessary actions to print a file designed for a drum printer on a train printer, if that is required. The FORMID attribute allows a program to indicate to the system operator the need of special forms or other information related to the file.

Printer files can be declared as blocked files. The most extreme case of blocking (MAXRECSIZE equal to 1, BLOCKSIZE equal to 132, UNITS equal CHARACTERS) can be useful in printing graphs. When writing a blocked printer file, a write of zero length can be used to terminate the block.

The attributes PAGESIZE, LINENUM, and PAGE allow the description of a logical printer page, and the Input-Output Subsystem does the line and page counting.

Line printer carriage control is an integral part of the I/O statements of the different languages. The CARRIAGECONTROL attribute allows the use of two data types which include the carriage control in the first character of the record.

MAGNETIC TAPE FILES

The KIND values associated with magnetic tape and their meanings are:

TAPE Any tape.

TAPE7 Seven-track NRZ tape.

TAPE9 Nine-track NRZ tape.

TAPEPE Nine-track phase-encoded tape.

Tape can be defined as a backup medium for printer or punch file; for example:

BACKUPKIND = TAPE7

When creating a tape file, the user can select the recording density by setting the DENSITY attribute to BPI200, BPI556, BPI800, BPI1600, or BPI6250. The density must be one that is available on the type of tape unit selected.

Magnetic tape files can be either labeled or unlabeled, as indicated by the attribute LABEL. Both ANSI (USASI) standard tape labels and B5500/B3500 system labels are recognized by the Input-Output Subsystem. Only ANSI label records are written, when creating a labeled tape file. Only the first (MFID) and last (FID) identifiers in a logical file's TITLE are used in the external name of a physical tape file. ANSI standards require that labeled multi-file tapes have two identifiers in each file's external name and that the MFID be the same for all the files on the tape. Labeled tapes have a serial number which is initially assigned to the tape by the SN system input message, and can be later read by use of the SERIALNO attribute. The label information also contains the CYCLE and VERSION attributes which can be used as part of the file identification on file assignment. The attribute SAVEFACTOR can be used to save a tape file and to determine how long it will remain saved before being purged.

Where a file is split across reels, it is possible to identify each specific reel (each file section) by use of the attribute FILESECTION. The attribute TAPEREELRECORD gives the logical record number relative to the beginning of the current reel.

When a tape file, suitably structured for backward reading, is positioned at the end of the file; it can be opened backward by setting the DIRECTION attribute to REVERSE.

The minimum physical record size (logical block) for tape files is six words. Blocks are padded with zeroes if necessary.

PAPER TAPE FILES

The value of the KIND attribute for paper tape files is PAPERPUNCH or PAPERREADER.

Paper tape files have a form of reel switch similar to tape files. When the trailer label is sensed, an input reel switch is performed for paper tape reader files which the system operator can respond to by the System Input Messages UL, FR, or DS. When "Low tape" is sensed for a paper tape punch file, an output reel switch is performed and the program will continue when a scratch paper tape punch is made available.

Reverse and Backspace Operations

With a paper tape punch, there is no reverse positioning, just as there is none in a card punch. Several forms of paper tape readers are used with the B 7000/B 6000. Some paper tape readers allow reverse operations, but others do not.

Because undesirable side effects occur when a reverse operation is initiated to a paper tape reader not equipped with the reverse read option, the MCP is provided with a run-time option which limits non-Direct I/O access to the paper tape reader. If NORVRSPAPERTAPE is reset, reverse operations are allowed, and paper tape read parity retry is attempted when a parity error is sensed. When this option is set, reverse operations are not allowed with non-Direct I/O, and read parity retry is not attempted on paper tape.

When using Direct I/O, the user must avoid setting the IOCW buffer attribute bit 39 to 1 if the paper tape reader does not support reverse tape operation.

Special Hardware Features

Various models of paper tape equipment have special translate plugboards. These are enabled only if INTMODE is BCL and EXTMODE is BCL for I/O operations having odd parity. For further information, consult the Burroughs Paper Tape Reader and Paper Tape Punch Operator's Manual.

Several models have special wiring blocks for placement of channels on the tape. Plugboard jumper connections are shown in Figures 2-1 and 2-2. Use of various INTMODE and EXTMODE combinations on equipment without this wiring block results, in some cases, in track transposition.

Input Operations

The File Attribute Settings given in Table 2-1 show the various combinations of translation and parity handling for paper-tape reads while using the INTMODE and EXTMODE attributes. The ACTION column shows the result of the corresponding attribute settings for a given plugboard configuration. Figure 2-1 gives plugboard configurations for paper tape input.

Output Operations

INTMODE and EXTMODE are the most important attributes for paper-tape punch output. The combinations for the translation and parity handling for paper-tape punch output are described in Table 2-2, File Attribute Settings, preceding Figure 2-2 which gives paper tape punch plugboard configurations.

Table 2-1. File Attribute Settings - Input Operations

INTMODE	EXTMODE	PARITY	ACTION	PLUGBOARD
EBCDIC (DISPLAY)	EBCDIC (DISPLAY)	Ignored	Straight binary 8-bit to 8-bit read. No parity checks are made as 8 bits of information are transferred. The leader and trailer contain all 1's.	A
BCL (DISPLAY-1)	BCL (DISPLAY-1)	Odd (Recording Mode is STANDARD)	External BCL tape code is read into core as internal BCL. (See B 6700 Hardware Handbook, Form No. 5000276 for INTERNAL and EXTERNAL BCL tape codes.) Parity errors are sensed. The leader and trailer are comprised of all zeros. Stop codes and the optional reader translate plugboard are enabled.	B
EBCDIC (DISPLAY)	BCL (DISPLAY-1)	Ignored	External BCL tape code is read into core as EBCDIC format. Parity errors are sensed. The leader and trailer contain all 0's.	B
BCL (DISPLAY-1)	(DISPLAY-1)	Even (Recording Mode is NONSTANDARD)	Internal BCL tape code is read into core as Internal BCL; that is, a blind 6-bit binary read with parity checking enabled. Parity checking looks for even parity. The trailer contains all 0's. Stop codes and the optional reader translate plugboard are enabled.	C
EBCDIC (DISPLAY)	ASCII (ASCII)	Ignored	Two entirely different actions take place depending on the attribute TRANSLATE. If TRANSLATE at file open is set to value DEFAULTTRANS (0), then a 7-bit binary read is performed, with the 8-bit being taken as parity. This parity is odd. If TRANSLATE at file open is set to value FULLTRANS (1), then an 8-bit binary read is performed and software translates from 8-bit ASCII to EBCDIC. The leader and trailer contains all 1's.	A

MV1897

Table 2-2. File Attribute Settings - Output Operations

INTMODE	EXTMODE	PARITY	ACTION	PLUGBOARD
EBCDIC (DISPLAY)	EBCDIC (DISPLAY)	Ignored	Straight binary 8-bit to 8-bit punch. No parity is generated. The leader contains all 1's.	D
BCL (DISPLAY-1)	BCL (DISPLAY-1)	Odd (Recording Mode is STANDARD)	External BCL tape code is punched from an internal BCL core image. An odd parity is generated. Stop codes and the optional punch unit plugboard translate are enabled. The leader and trailer contain all 0's.	E
EBCDIC (DISPLAY)	BCL (DISPLAY-1)	Ignored	External BCL tape code is punched from EBCDIC core image. An odd parity is generated. Stop codes and the optional punch unit plugboard translate are enabled. The leader and trailer contain all 0's.	E
BCL (DISPLAY-1)	BCL (DISPLAY-1)	Even (Recording Mode is NONSTANDARD)	Internal BCL code image is punched from internal BCL core image, with an even parity being generated. The leader and trailer contain all 0's.	F
EBCDIC (DISPLAY)	ASCII (ASCII)	Ignored	Depending on the settings of the attribute TRANSLATE at file open, the following two entirely different things happen. If TRANSLATE is DEFAULT-TRANS (0), then 7 bits of core image are punched from the low-order 7 bits of each 8-bit byte, and an odd parity is generated in the eighth bit. If TRANSLATE is FULLTRANS (1), then the EBCDIC core image is first translated by means of the software EBCDICTOASCII table, and then the eight bits are punched using an 8-bit binary data transfer. The leader and trailer contains all 1's.	D

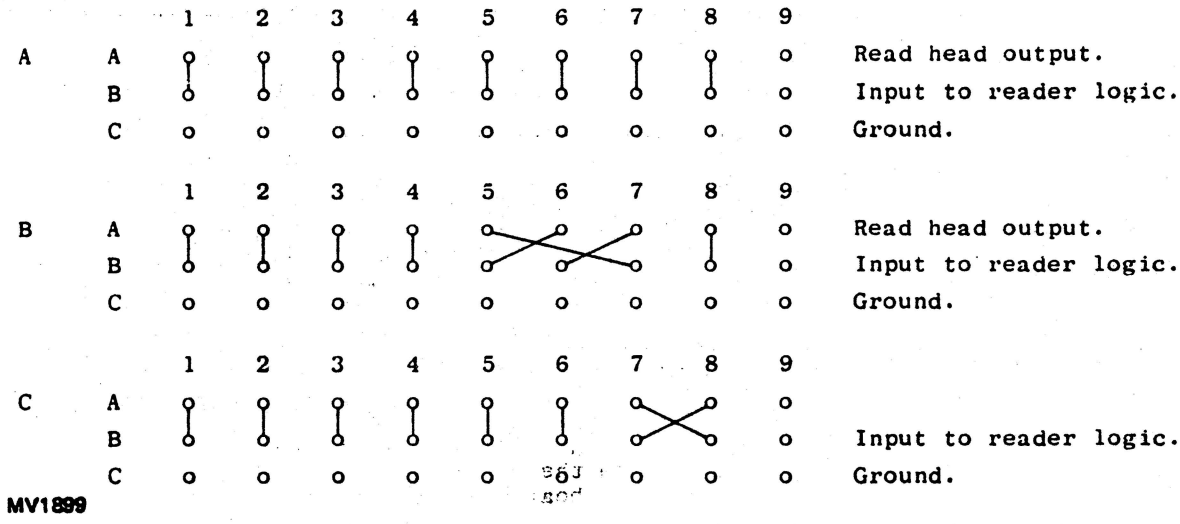


Figure 2-1. Plugboard Configurations for Paper Tape Read Operations

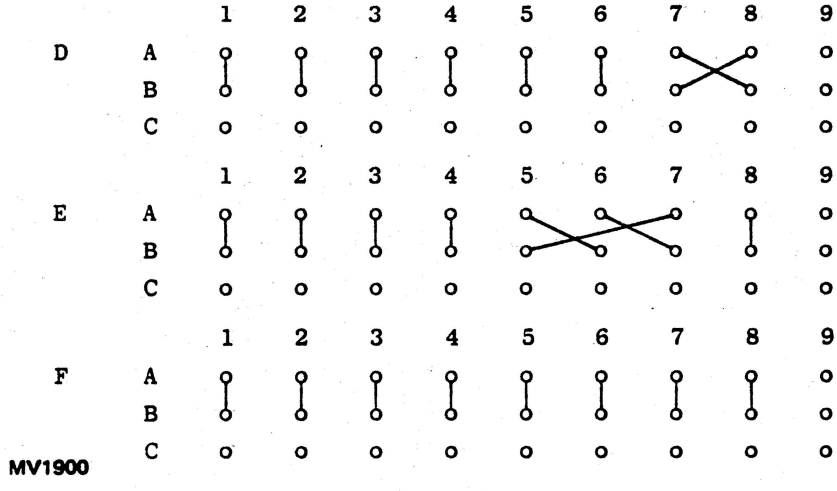


Figure 2-2. Plugboard Configurations for Paper Tape Punch Operations

PORT FILES

Port files differ from other KINDs of files in that no physical I/O device is associated with the logical file. Instead, all I/O operations occur between logical files declared in different programs, or in the same program, through one or more subfiles.

The value of the KIND attribute for port files is PORT. The attributes pertinent to port files fall into three groups:

a. Attributes that directly affect the port file as a whole but not the individual subfiles.

b. Attributes that directly affect both port files and subfiles.

c. Attributes that directly affect the individual subfiles but not the port file as a whole.

All attributes pertinent to port files are listed below. Each attribute is followed by a letter in parentheses indicating the above group in which it belongs.

BLOCKSTRUCTURE (a)	INPUTEVENT (b)	OUTPUTEVENT (c)
CENSUS (b)	INTNAME (a)	SECURITYTYPE (a)
CHANGEDSUBFILE (a)	LASTSUBFILE (a)	STATE (a)
CHANGEEVENT (b)	MAXCENSUS (c)	SUBFILEERROR (c)
CURRENTRECORD (c)	MAXRECSIZE (b)	TITLE (a)
FILESTATE (c)	MAXSUBFILES (a)	YOURNAME (c)
FRAMESIZE (a)	MYNAME (a)	YOURUSERCODE (c)

Refer to the individual discussions of these attributes in Section 3 of this manual for specific information.

Port File Input/Output Operations

Every port file has one or more subfiles on which I/O operations are performed. If no subfile index is specified, either the I/O operation is directed to the only subfile, if MAXSUBFILES equals 1, or an invalid subfile index error is returned, if MAXSUBFILES is greater than 1. If a subfile index of zero is specified for a read operation, the read (called a nonselective read) is performed on the subfile containing the next available message. If a subfile index of zero is specified for a write operation, the write (called a broadcast write) is performed on every subfile in the port. If an error occurs on any subfile during a broadcast write, the error is returned in the usual manner via the result descriptor and the STATE attribute.

I/O statements may contain a DONTWAIT specification. Normally, if a read operation is performed on a subfile with CENSUS equal to zero, the program is suspended until a message is available, and if a write operation is performed on a subfile with no buffers available, the program is suspended until a buffer is available. However, when DONTWAIT is specified for a read or write operation and it cannot be performed immediately, the I/O operation is terminated, and the program resumes execution. If an I/O operation is prematurely terminated because DONTWAIT was specified, the STATE attribute is set accordingly.

The quantity of data transferred during an I/O operation on a port subfile depends on some or all of the following:

a. The quantity specified in the I/O statement.

b. The MAXRECSIZE of the subfile.

c. The size of the user program's buffer.

d. The quantity of data actually received during a read operation on a port file with BLOCKSTRUCTURE equal to EXTERNAL.

- If the BLOCKSTRUCTURE of a port file equals EXTERNAL and the I/O operation is a read operation, the quantity of data transferred equals the minimum of the quantities represented by items a, b, c, and d above. (The quantity represented by item d may be ascertained programmatically by using the CURRENTRECORD attribute.) In all other instances, item d is not a factor, and the quantity of data transferred equals the minimum of the quantities represented by items a, b, and c. Read operations on port files with BLOCKSTRUCTURE equal to EXTERNAL do not supply blank fill; read operations on port files with BLOCKSTRUCTURE equal to FIXED supply blank fill on the right.

If the quantity of data specified in the I/O statement is greater than the MAXRECSIZE of the subfile or greater than the size of the user program's buffer, the data is truncated without an error being returned.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

FILE ATTRIBUTES

INTRODUCTION

The file attribute interface allows a user's program to interrogate or request modification of the Input-Output Subsystem information defining his file. The information is not accessible to any program without going through this special subsystem interface.

Each file attribute defines a characteristic of the file. Changing the state of a single attribute affects the file defined by that attribute and may cause the states of other attributes to be changed.

When the logical file is first touched in a program (either by setting or accessing any attribute, opening the file, or by any I/O statement involving the file), the attributes referenced in the file's declaration are set to their initial values. When label-equation via Work Flow Language (WFL) statements exist for the logical file at either compile time or execute time, the label-equated attributes are then set; first to the compiled-in label-equation and then to those presented at execute time. Label-equation matching utilizes the value of the INTNAME attribute. Finally, if the initial action on the file was to set an attribute, that attribute is set.

To understand when an attribute can be read or set, the different states of a logical file (open, closed, assigned, unassigned, with or without I/O) and how they can interrelate, must be made clear. Before data can be transferred between a logical and physical file, the logical file must be opened and the physical file must be assigned to the logical file. This can be accomplished by explicitly opening the logical file, or via the AVAILABLE attribute. If the OPTIONAL attribute is TRUE for a logical file, opening the file can leave it unassigned to a physical file. Opening a file explicitly does not cause data to be transferred between the logical and physical files, and the logical file can be closed without any I/O being performed upon the file. The logical file can be closed with retention, which leaves the physical file assigned, or closed with release.

When a physical file is assigned to the logical file, attributes which are not relevant to the physical file are ignored. If the file is opened and any logical attribute is found to be invalid although correctly set, the attribute's value will be changed by the Input-Output Subsystem and a nonfatal run-time error will be given. The assignment of a physical file to the logical file can cause the attributes of the logical file to be changed. This can be the result of operator intervention in file assignment, the conflict of contradictory attribute settings, the mismapping of logical and physical files, or the explicit setting of the DEPENDENTSPECS attribute to TRUE or the FILETYPE attribute to a value of 7 or 8. The values returned by attributes while a physical file is assigned to a logical file are values in use by the subsystem. The values may also be affected by system options, by certain task attributes, and by the FAMILY statement of WFL. Whenever there is a conflict between the logical and physical units used, the values returned by the attributes will be logical units.

Pointer valued attributes which may be reset can be done so by assigning to them a null identifier (".").

DESCRIPTION OF FILE ATTRIBUTES

This section provides descriptions of the file attributes. The file attributes (listed in Table 3-1) are discussed in alphabetical order. The name of the attribute precedes a summary of the attribute's characteristics. This is followed by a detailed discussion of the attribute.

The summary of the attribute's characteristics includes the following information: the physical unit types (KIND) to which this attribute applies (general, disk only, etc.), the ability of the attribute to be accessed (read) or set (write), the conditions under which the attribute may be accessed or set (unassigned, closed, assigned, open, anytime), the data type required when using the attribute (pointer, array, Boolean, integer, real, event), and the KINDs of files for which the attribute value is saved with the physical file (Disk, Tape).

Example

STATE

General, read only, assigned, real, not permanent

Can be expanded as follows:

The attribute STATE is a real (an entire word containing 48 bits of formatted information) valued attribute which is valid for all file KINDs. The attribute is read-only and valid only after the logical file has been assigned to a permanent file or peripheral unit. The value of the attribute is not saved with physical file.

Example

AREAS

Disk, read/write, anytime/closed, integer, Disk

Which can be expanded as follows:

The attribute AREAS is an integer-valued attribute which is only valid for head-per-track disk or disk pack (KIND equal to DISK or PACK) files. The attribute can be read at anytime and may be set only when the file is closed. The value of this attribute is saved with the physical disk file.

Tables 3-1 through 3-3 list the file attributes and some associated information. Table 3-1 lists the general type of file attribute, Table 3-2 lists the Direct I/O buffer attributes, and Table 3-3 lists the diskheader attributes.

The column headed Name is the symbolic name that the compilers recognize for this attribute. In Table 3-1, the column headed Kind indicates the type of files for which the attributes are valid. These include general, Data Comm, port, subfile, tape, printer, and disk (disk pack or head-per-track). This column is not necessary for Tables 3-2 and 3-3, as these tables list those attributes that pertain only to Direct I/O or disk files.

The Read/Write column describes the ability of the attribute to be accessed (read) or set (written). The entries in the Read/Write column have the following meanings:

RW The attribute can be read and written.

RO The attribute can be read but cannot be written.

WO The attribute can be written but cannot be read.

The Read column gives the condition under which the attribute may be read (accessed), and the Write column gives the conditions under which attribute may be written (set). Entries in the Read and Write columns have the following meanings:

C File must be closed.

O File must be open.

U Physical file must not be assigned to a logical file.

A Physical file must be assigned to a logical file.

V Variable - may be accessed or set any time.

The Type column gives the data type required when using this attribute. The entries in the type column have the following meanings:

A Array-valued attribute.

B Boolean-valued attribute.

E Event-valued attribute.

I Integer-valued attribute.

P Pointer-valued attribute.

R Real-valued attribute.

T Translatetable-valued attribute.

Entries in the column labeled Permanent indicate whether or not the value of the attribute is saved with the physical file. When the value of an attribute is saved with the physical file, the value can be used when the file is reopened. The entries appearing in the Permanent column have the following meanings:

D The value is saved with the physical file only if the file is a disk file.

T The value is saved with the physical file only if the file is a labeled tape file.

D,T The value is saved with the physical file if the file is either a disk file or a labeled tape file.

N No value is saved with the physical file regardless of the kind of file.

NOTE

A dash (-) in any column indicates that the column is not applicable.

Table 3-1. General File Attributes

Name	Kind	Read/Write	Read	Write	Type	Permanent
APL	Disk	RW	O	C	B	D
AREAALLOCATED	Disk	RO	A	-	B	D
AREACLASS	Disk	RW	V	V	I	D
AREALENGTH	Disk	RW	V	U	I	D
AREAS	Disk	RW	V	C	I	D
AREASIZE	Disk	RW	V	C	I	D
ASSIGNTIME	Data Comm	RO	O	-	I	N
ATTERR	General	RO	V	-	B	N
ATTVALUE	General	RO	V	-	R	N
ATTYPE	General	RO	V	-	I	N
AVAILABLE	General	RO	V	-	I	N
BACKUPKIND	Printer/Punch	RW	V	C	I	N
BLOCK	Disk/Tape	RO	V	-	I	N
BLOCKSIZE	General	RW	V	C	I	D,T
BLOCKSTRUCTURE	General	RW	V	C	I	D,T
BUFFERS	General	RW	V	C	I	N
CARRIAGECONTROL	Printer	RW	V	C	I	N
CENSUS	Data Comm/Port	RO	O	-	I	N
CHANGEDSUBFILE	Port	RO	V	-	I	N
CHANGEEVENT	Port/Subfile	RO	V	-	E	N
CONTROLMESSAGE	Reader Sorter	WO	-	O	A	N
COPIES	Disk	RW	V	C	I	N
COPYNAME	Disk	RO	A	-	P	D
CREATIONDATE	Disk/Tape	RO	A	-	I	D,T
CRUNCHED	Disk	RO	A	-	B	D

Name	Kind	Read/Write	Read	Write	Type	Permanent
CURRENTBLOCK	General	RO	V	-	I	N
CURRENTRECORD	General	RO	V	-	I	N
CYCLE	Disk/Tape	RW	V	V	I	D,T
CYLINDERMODE	Diskpack	RW	V	C	B	D
DATE	(Nonpreferred synonym. See CREATIONDATE.)					
DENSITY	Tape	RW	V	C	I	T
DEPENDENTSPECS	General	RW	V	C	B	N
DIRECTION	Tape/Papertape Reader	RW	V	V	I	N
DISPOSITION	Data Comm	RO	O	-	I	N
DUPLICATED	Disk	RW	V	C	B	D
ENABLEINPUT	Data Comm	RO	O	-	B	N
EOF	Tape	RO	O	-	B	N
ERRORTYPE	Disk	RO	O	-	I	N
EXCLUSIVE	Disk	RW	V	U	B	N
EXTMODE	General	RW	V	C	I	D,T
FAMILY	(Nonpreferred synonym. See STATIONLIST.)					
FAMILYINDEX	Disk	RW	V	V	I	D
FAMILYNAME	Disk	RW	V	C	P	D
FAMILYSIZE	(Nonpreferred synonym. See STATIONCOUNT.)					
FCM	Reader Sorter	RW	V	C	R	N
FILEKIND	Disk/Tape	RW	V	V	I	D,T
FILENAME	General	RW	V	V	P	D,T
FILEORGANIZATION	Disk	RW	V	C	I	D
FILESECTION	Tape	RW	V	C	I	T
FILESTATE	General	RO	V	-	I	N
FILETYPE	General	RW	V	C	I	D,T
FILEUSE	General	RW	V	C	I	N
FLEXIBLE	Disk	RW	V	V	B	N
FORMID	Printer/Punch	RW	V	C	P	N

Name	Kind	Read/Write	Read	Write	Type	Permanent
FORMMESSAGE	(Nonpreferred synonym. See FORMID.)					
FRAMESIZE	General	RW	V	C	I	D,T
GENERATION	Disk/Tape	RW	V	C	I	N
HOSTNAME	General	RW	V	V	P	N
IAD	Disk	RO	V	-	B	D
IMAGEFILEID	Reader Sorter	RW	V	V	P	N
IMAGEFILEMAP	Reader Sorter	WO	-	O	A	N
INPUTEVENT	Port/Subfile	RO	V	-	E	N
INPUTTABLE	General	WO	-	V	T	N
INTERCHANGE	Diskpack	RW	V	C	B	D
INTMODE	General	RW	V	C	I	N
INTNAME	General	RW	V	U	P	N
IOCLOCKS	General	RO	A	-	R	N
IOINERROR	Disk/Tape	RO	O	-	B	N
KIND	General	RW	V	C	I	N
LABEL	General	RW	V	U	I	N
LABELTYPE	(Nonpreferred synonym. See LABEL.)					
LASTRECORD	Disk	RW	A	A	I	D
LASTSTATION	(Nonpreferred synonym. See LASTSUBFILE.)					
LASTSUBFILE	Data Comm/Port	RW	O	O	I	N
LINENUM	Printer	RW	V	V	I	N
MAXCENSUS	Subfile	RW	V	V	I	N
MAXRECSIZE	General	RW	V	C	I	D,T
MAXSUBFILES	Port	RW	V	C	I	N
MINRECSIZE	General	RW	V	C	I	D,T
MYNAME	Port	RW	V	C	P	N
MYUSE	General	RW	V	C	I	N
NEWFILE	General	RW	V	C	B	N
NORESOURCEWAIT	Disk	RW	V	V	B	N

Name	Kind	Read/Write	Read	Write	Type	Permanent
OPEN	General	RW	V	V	B	N
OPTIONAL	General	RW	V	C	B	N
OUTPUTEVENT	Port	RO	V	-	E	N
OUTPUTTABLE	General	WO	-	V	T	N
PACKNAME	(Nonpreferred synonym. See FAMILYNAME.)					
PAGE	Printer	RW	V	V	I	N
PAGESIZE	Printer/Data Comm	RW	V	V	I	N
PARITY	Tape/Paper Tape	RW	V	C	I	T
POPULATION	Data Comm/Disk	RO	A	-	I	N
PRESENT	General	RO	V	-	B	N
PROTECTION	Disk/Tape	RW	V	C	I	D,T
RECEPTIONS	Data Comm	RO	O	-	I	N
RECORD	General	RO	V	-	I	N
RECORDINERROR	Disk/Tape	RO	O	-	I	N
RECOVERYFILEID	Reader Sorter	RW	V	V	P	N
RECOVERYINDEX	Reader Sorter	RW	V	V	I	N
REEL	(Nonpreferred synonym. See FILESECTION.)					
REMOTEID	Reader Sorter	RO	A	-	P	N
RESIDENT	General	RO	V	-	B	N
ROWADDRESS	Disk	RO	A	-	I	D
ROWSINUSE	Disk	RO	A	-	I	D
RSERROR	Reader Sorter	RO	A	-	I	N
RSEXTENDIBLE	Reader Sorter	RW	V	C	B	N
RSRESTART	Reader Sorter	RW	V	C	B	N
RSRESULT	Reader Sorter	RO	A	-	R	N
RSUNITNO	Reader Sorter	RW	V	C	I	N
SAVEFACTOR	Tape/Disk	RW	V	C	I	D,T
SCREEN	Data Comm	RO	O	-	B	N
SCREENSIZE	Data Comm	RO	O	-	I	N

Name	Kind	Read/Write	Read	Write	Type	Permanent
SECURITYGUARD	Disk	RW	V	V	P	D
SECURITYTYPE	Disk/Port	RW	V	V	I	D
SECURITYUSE	Disk	RW	V	V	I	D
SENSITIVEDATA	Disk	RW	V	C	B	D
SERIALNO	Tape/Disk	RW	V	V	R	D,T
SINGLEPACK	(Nonpreferred synonym. See SINGLEUNIT.)					
SINGLEUNIT	Disk	RW	V	C	B	D
SIZEMODE	General	RW	V	C	I	D,T
SIZEOFFSET	General	RW	V	C	I	D,T
SIZE2	General	RW	V	C	I	D,T
SORTFILEID	Reader Sorter	RW	V	V	P	N
SORTTABLECODE	Reader Sorter	RW	V	V	I	N
SPEED	HPT	RW	V	C	I	D
STATE	General	RO	A	-	R	N
STATIONCOUNT	Data Comm	RO	O	-	I	N
STATIONLIST	Data Comm	WO	-	O	P	N
STATIONNAME	Data Comm	RO	O	-	P	N
STATIONSDENIED	Data Comm	RO	O	-	I	N
SUBFILEERROR	Subfile	RO	V	-	I	N
TANKING	Data Comm	RW	V	C	I	N
TAPEREELRECORD	Tape	RO	A	-	I	T
TIMELIMIT	Data Comm	RW	O	V	R	N
TITLE	General	RW	V	V	P	D,T
TRAINID	Printer	RW	V	C	I	N
TRANSLATE	General	RW	V	C	I	N
TRANSLATING	General	RO	A	-	B	N
TRANSMISSIONNO	Data Comm	RO	O	-	I	N
TRANSMISSIONO	(Nonpreferred synonym. See TRANSMISSIONNO.)					
TRANSMISSIONS	Data Comm	RO	O	-	I	N

Name	Kind	Read/Write	Read	Write	Type	Permanent
UCRID	Reader Sorter	RW	V	V	P	N
UNITNO	General	RW	V	U	I	N
UNITS	General	RW	V	C	I	D,T
UPDATEFILE	Disk	RW	V	C	B	N
USECATALOG	Disk/Tape	RW	V	C	B	N
USEDATE	Disk	RO	A	-	I	D
VERSION	Disk/Tape	RW	V	V	I	D,T
WIDTH	Data Comm	RO	O	-	I	N
YOURNAME	Subfile	RW	V	C	P	N
YOURUSERCODE	Subfile	RW	V	C	P	N

Table 3-2. Direct I/O Buffer Attributes

Name	Read/Write	Read	Write	Type
IOADDRESS	RO	V	-	I
IOCANCEL	RW	V	V	B
IOCHARACTERS	RO	V	-	I
IOCOMPLETE	RO	V	-	B
IOCW	RW	V	V	R
IOEOF	RO	V	-	B
IOERRORTYPE	RO	V	-	I
IOMASK	RW	V	V	R
IOPENDING	RO	V	-	B
IORECORDNUM	RO	V	-	I
IORESULT	RO	V	-	B
IOTIME	RO	V	-	I
IOWORDS	RO	V	-	I

Table 3-3 lists the DCALGOL diskheader attributes and includes only the Name and Type columns. These attributes pertain to disk (head-per-track or diskpack) and can be accessed (read) or set (write). In this table, an asterisk precedes those diskheader attributes that function identically to those file attributes listed in Table 3-1. The other diskheader attributes are explained in the B 5000/B 6000/B 7000 Series DCALGOL Reference Manual.

Table 3-3. Diskheader Attributes

Name	Type
*BLOCKSIZE	I
DATE	I
*DUPLICATED	I
EOFBITS	I
EOFSEGMENT	I
EUNUMBER	I
*FILEKIND	I
*FILETYPE	I
IAD	B
LASTACCESSDATE	I
*MAXRECSIZE	I
*MINRECSIZE	I
MODE	I
ROWADDRESS	I
ROWCLASS	I
ROWS	I
ROWSIZE	I
SAVEFACTOR	I
*SIZEMODE	I
*SIZEOFFSET	I
*SIZE2	I
*UNITS	I

APL

Disk, read/write, open/closed, Boolean, Disk

The attribute APL is used to satisfy APL file access restrictions. The APL attribute may only be set by an APL code file before the physical file is created. If a non-APL code file attempts to create a file with APL equal to TRUE, it is aborted. The APL attribute may be read when the file is open and returns the value for the physical file. The default value is FALSE.

AREAALLOCATED

Disk, read only, assigned, Boolean, Disk

The attribute AREAALLOCATED indicates whether or not a specific area of a disk file has been allocated. AREAALLOCATED requires at least one parameter, the area number. If the physical file is duplicated, AREAALLOCATED requires the copy number as a second parameter. Area numbers begin at 0; copy numbers begin at 1. AREAALLOCATED returns TRUE if the specified area has been allocated and FALSE if the specified area has not been allocated.

Examples

```
BOOL:=F(0).AREAALLOCATED;       %FOR NONDUPLICATED FILES
BOOL:=F(0,1).AREAALLOCATED;     %FOR DUPLICATED FILES
```

AREACLASS

Disk, read/write, anytime/anytime integer, Disk

NOTE

On the B 6900 and later systems, the value of the AREACLASS attribute is ignored. The file attribute FAMILYINDEX is supported as an alternative means of controlling disk allocation. As of the 3.4 release, AREACLASS will be ignored by all systems. Refer to the discussion of the FAMILYINDEX attribute.

The attribute AREACLASS assigns or returns the class value of the family member upon which the disk area resides. AREACLASS can have a value from zero to 255. Zero is the default value and is the same as being unclassed. The AREACLASS attribute requires that an index, the row or area number, be specified when it is set or accessed unless the file is closed and unassigned. If the file is duplicated, the copy number is also required.

Example

```
DSK(ROWNBR).AREACLASS :=3;

DUPLDSK(ROWNBR,COPYNBR).AREACLASS :=23;
```

Different copies of a duplicated disk file can have different AREACLASS values.

Area numbers begin at 0; copy numbers begin at 1.

If the AREACLASS attribute has been set before a new file is created, (for example, while the logical file is closed), all the rows of the file will be assigned that AREACLASS when the file is opened.

The AREACLASS attribute must be set for an area before the disk space is allocated for that area. The row address word for each area will contain the AREACLASS value. The values may be changed for an individual area after the file is OPEN, but before the area is allocated. Attempting to set AREACLASS for an area that has already been allocated causes an attribute error.

A family member assumes a class value whenever an area is allocated on it which has been assigned a nonzero AREACLASS. The member retains this class value until all areas with that classification have been removed from it. Areas which have a nonzero AREACLASS are allocated from a family member with the same class if space is available. If space is not available on a member with the same class, space is allocated from a member with a class of zero (class zero is an unclassed family member), changing the class of that member to the value of AREACLASS. While a family member has assumed a class, no area with a different, nonzero AREACLASS can be allocated from it. Areas with zero AREACLASS will be allocated from any family member that has the required available space.

If the area specified by the index has been allocated, AREACLASS returns the value for the physical area.

The AREACLASS of areas of a disk file are preserved by Library Maintenance, but can be overridden when copying the files back from tape. If predictable. behavior and complete control of family member area allocation is required, the FAMILYINDEX attribute is the recommended alternative to AREACLASS.

AREALENGTH

Disk, read/write, anytime/unassigned, integer, Disk

The value of the attribute AREALENGTH is the number of FRAMESIZE units per area of a disk file.

If AREALENGTH has never been set, AREASIZE is used instead. If AREALENGTH has been set, its value is used regardless of the value of AREASIZE. An attribute error is reported if an attempt is made to set AREASIZE after AREALENGTH has been set.

AREALENGTH may be set only when the file is unassigned. Assigning a value to AREALENGTH is meaningful only when a new file is to be created because once the file is assigned, AREALENGTH always reflects the value for the physical. file regardless of the value for the logical file. When a new file is created, the value of AREALENGTH is adjusted so that it is divisible by the value of BLOCKSIZE. If AREALENGTH is equal to zero, the default value of AREALENGTH is used. The default value of AREALENGTH is a value close to MAXRECSIZE * 1000 that is also divisible by BLOCKSIZE. The maximum allowable value of AREALENGTH depends on the size of the disk or pack hardware.

If the BLOCKSIZE of the file is not a multiple of the sector size of the disk, some space in the last sector in each block is not used. The AREALENGTH is the total size of the area not including the unused space, if any.

AREAS

Disk, read/write, anytime/closed, integer, Disk

The value of the attribute AREAS is the number of areas (or rows) a disk file can allocate.

The AREAS attribute may be set only while the file is closed. If AREAS is zero, the default value of twenty (20) is used when creating a new disk file. The maximum value is 1000. The setting of the AREAS attribute is not used when opening a permanent disk file, or reopening a file closed with retention. The AREAS attribute can be overridden when expanding the end of the file if the attribute FLEXIBLE has been set to TRUE.

When the file is assigned, the AREAS attribute returns the number of areas in the physical file.

AREASIZE

Disk, read/write, anytime/closed integer, Disk

The value of the attribute AREASIZE is the number of logical records in an area of a disk file.

AREASIZE may be set only when the file is closed. Assigning a value to AREASIZE is meaningful only when a new file is to be created because once the file is assigned, AREASIZE always reflects the value for the physical file regardless of the value for the logical file. When a new disk file is created, the value of AREASIZE is adjusted so that it is also a multiple of records-per-block. If AREASIZE is unspecified or equal to zero and AREALENGTH is unspecified, a value equal or close to 1000 is assumed. If a nonzero value is specified for AREASIZE and AREALENGTH is unspecified, that value is used. The maximum value is 1,048,575.

If the file contains variable-length records (BLOCKSTRUCTURE has a value other than FIXED, or FILETYPE is between 1 and 6 inclusive), AREASIZE is expressed in blocks instead of records.

The attributes KIND, NEWFILE, MYUSE, and (prior to the 3.5 release) AREASIZE determine whether a new file is created or a permanent file is assigned when the file is opened. As of to the 3.5 release, AREASIZE will no longer affect the assignment of files in this way. Refer to New File vs. Permanent File in Section 4 of this manual.

ASSIGNTIME

Data Comm, read only, open, integer, not permanent

The attribute ASSIGNTIME returns the time at which the station was assigned to the file. The ASSIGNTIME attribute requires that an index, the relative station number (RSN), be specified. The value returned is the time, relative to midnight, in seconds when the station was assigned.

ATTERR

General, read only, anytime, Boolean, not permanent

The attribute ATTERR returns true if the last file attribute action was in error or a file was opened with incompatible attribute values.

Most attribute errors are nonfatal, but may be informative to the object program. The ATTERR attribute is reset to FALSE after a successful attribute action. ATTERR used in conjunction with ATTYPE and ATTVALUE may be helpful in correcting programming errors.

ATTVALUE

General, read only, anytime, real (word), not permanent

Following an attribute error, the attribute ATTVALUE will contain a value which may aid in correcting the error. If the file is open and there is an attempt to set an attribute which requires the file to be closed, ATTVALUE will return a word with all bits on (i.e., REAL(NOT FALSE)). This is a small negative number and will print as zero. If the attribute in error is a pointer-valued attribute, ATTVALUE returns the pointer (string descriptor) as an operand (i.e., tag equal zero); otherwise, the value to which the attribute was to be set will be returned. Refer to descriptions of ATTERR and ATTYPE in this section.

ATTYPE

General, read only, anytime, integer, not permanent

The attribute ATTYPE returns the number of the last file attribute which was incorrectly referenced.

ATTYPE along with ATTERR and ATTVALUE are useful in determining programming errors related to files.

The VALUE function may be used, in the languages that provide it, to ascertain the number of an attribute when the attribute name is given as a parameter. (Refer to the respective language manuals for syntax.)

Example

```
IF F.ATTYPE = VALUE(BLOCKSIZE) THEN HANDLEPROBLEM;
```

The ALGOL statement above uses ATTYPE and the VALUE function to determine if BLOCKSIZE was the last attribute to be incorrectly referenced.

AVAILABLE

General, read only, anytime, integer, not permanent

The AVAILABLE attribute attempts to open a file and, when impossible, reports the reason for the failure without suspending the program and requiring operator intervention. The operator is required to resolve duplicate file situations when they arise.

For files other than port files, when the AVAILABLE attribute is interrogated, the open operation performed on the file and the value returned to the program are the same as when the OPEN function is used. If the file is a port file, the open operation performed is equivalent to that performed by the OPEN function when the open option AVAILABLE is specified and the subfile index is 0. (Refer to Port Subfile Assignment in Section 4 of this manual.)

AVAILABLE returns the following values. Odd values indicate successful results; even values indicate unsuccessful results.

0	The permanent file exists but is not available (the program is locked out).
1	The open operation was successful.
2	The permanent file does not exist or cannot be opened because the logical file and the permanent file are incompatible. If the file is a port file, no offered complementary subfile was found.
4	The genealogy of the file does not match.
6	The serial number does not match.
8	The file is cataloged as a nonresident backed-up file.
10	No resources available for a tape file.
12	A required pack is not mounted.
14	The file is not available due to ACCESSCODE protection in the user's directory.
38	The file is a port file, and the host became unreachable during the open operation.
40	The file is a port file, and the specified subfile was not in the CLOSED state.
42	The file was a port file with MAXSUBFILES greater than 1.

BACKUPKIND

Printer/Punch, read/write, anytime/closed, integer, not permanent

This attribute, used in conjunction with KIND attribute, allows specification of printer and punch backup devices. BACKUPKIND may only be set when the file is closed.

The allowable mnemonic values for BACKUPKIND are as follows. DONTCARE is the default.

Mnemonic	Meaning
DONTCARE	Use the system default values.
DISK	Use head-per-track disk.
PACK	Use diskpack.

TAPE	Use any available tape.
TAPE7	Use 7 track NRZ tape.
TAPE9	Use 9 track NRZ tape.
TAPEPE	Use 9 track PE tape.

The value of BACKUPKIND is used when the file is opened to select an output device for printer and punch files. When the value is DONTCARE (which means nothing was specified), other system and task options are used to select the output device. Normal SB substitution may occur, as well, resulting in selection of a different output device.

Accessing BACKUPKIND while the file is unassigned returns the value most recently assigned the attribute. If the file is assigned and associated with a backup device, the value returned is the KIND for the associated device. If the file is assigned but is not associated with a backup device (because it is not a printer or punch file or it is directly attached to a printer or punch), a value of DONTCARE (nothing specified) will be returned.

BLOCK

Disk/Tape, read only, anytime, integer, not permanent

The attribute BLOCK returns the number of the logical block referenced in the last I/O statement. When the file is closed, or if it is open but no I/O has been performed upon the file, BLOCK will return -1. BLOCK will also return -1 after an attempted I/O beyond the end of the file.

The counting of blocks is zero relative; that is, the first block in a file is BLOCK zero (0).

BLOCKSIZE

General, read/write, anytime/closed, integer, Disk/Tape

The value of the BLOCKSIZE attribute is the length of a block. If the FRAMESIZE attribute does not equal 48 (or the UNITS attribute equals CHARACTERS), then BLOCKSIZE is given in INTMODE units, otherwise in words. BLOCKSIZE may be set only when the file is closed. The maximum value is 65,535. If BLOCKSIZE is less than MAXRECSIZE it will be set to MAXRECSIZE when the file is opened.

TEXT DELETED

The default value of BLOCKSIZE is dependent upon the physical unit (KIND) assigned to the file and the value of the attribute MAXRECSIZE. See the discussion under the MAXRECSIZE attribute in this section.

Card files and Data Comm (KIND equal to REMOTE) files have restricted upper bounds for BLOCKSIZE (and therefore MAXRECSIZE). Card files have upper bounds of 80 if character-oriented; 10 words if BCL; 14 words if EBCDIC; or 20 words if BINARY mode. The upper limit for Data Comm files is approximately 500 words (the size of a GETAREA row -8). These restrictions are enforced when the file is opened and, if necessary, the program receives a nonfatal run-time error and the BLOCKSIZE value reduced appropriately. If a Direct Data Comm file attempts to write a buffer that is too long, the I/O is cancelled.

BLOCKSTRUCTURE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute BLOCKSTRUCTURE specifies the format of the records and the structure of the file.

The following table lists each of the possible mnemonic and integer values of BLOCKSTRUCTURE together with its meaning and corresponding FILETYPE value.

Mnemonic	Integer Value	Corresponding Value of FILETYPE	Description
FIXED	0	0	Blocked or unblocked fixed-length records. This value is the default value of BLOCKSTRUCTURE.
EXTERNAL	1	3	Variable-length records. Neither the record itself nor the structure of the file contain information about the length of the record; length information must be specified externally in the I/O statement.
VARIABLE	2	1	Variable-length records. The record length is contained (expressed in decimal) in the first four INTMODE characters of the record.
LINKED	3	6	FORTRAN-linked, variable-length records. The link words are maintained by the logical I/O subsystem and are not part of the records. The INTMODE of the file is assumed to be BINARY and software translation is never attempted.

If BLOCKSTRUCTURE has not been set, the structure of the file is determined by the value of FILETYPE. If FILETYPE is also unspecified, the default is used. If BLOCKSTRUCTURE has been set: (1) it determines the structure of the file (overriding the value of FILETYPE), and (2) when the file is opened, FILETYPE is assigned the value corresponding to the value of BLOCKSTRUCTURE. An attribute error is reported if an attempt is made to set FILETYPE after BLOCKSTRUCTURE has been set.

If BLOCKSTRUCTURE is FIXED, MINRECSIZE is set equal to MAXRECSIZE when the file is opened.

Variable-length, blocked records (BLOCKSTRUCTURE equal to EXTERNAL, VARIABLE, or LINKED) are processed differently than fixed-length records. If the length of the record just written plus MINRECSIZE plus the offset into the block is greater than BLOCKSIZE or the record to be written plus the offset into the block is greater than BLOCKSIZE, the current block is written to the physical file. In the latter case, the record to be written becomes the first record of the next block. If the logical file is assigned to a peripheral unit that allows variable-length blocks (for example, tape files), only the part of each

block containing valid data is written. However, if the logical file is assigned to a peripheral that requires fixed-length blocks (for example, disk files), a field of null characters equal in length to MINRECSIZE is written into the remainder of the block as an end-of-block marker, and the entire block is written to the physical file.

The value of BLOCKSTRUCTURE affects the default value of MINRECSIZE. Files of variable-length records containing link words or length fields (BLOCKSTRUCTURE equal to VARIABLE or LINKED) require a value for MINRECSIZE equal to or greater than the size of these fields. Files of fixed-length records (BLOCKSTRUCTURE equal to FIXED) use a MINRECSIZE equal to MAXRECSIZE.

If BLOCKSTRUCTURE is read when the file is open, the value is computed from the physical file and returned even if BLOCKSTRUCTURE has not been set for the logical file. If an attempt is made to read the value of BLOCKSTRUCTURE for an open file whose structure was determined by a FILETYPE value for which there is no corresponding BLOCKSTRUCTURE value, an attribute error is returned.

BLOCKSTRUCTURE has special meaning for port files. The only valid values of BLOCKSTRUCTURE when KIND equals PORT are FIXED and EXTERNAL. When KIND equals PORT and BLOCKSTRUCTURE equals FIXED, the buffer is blank-filled on the right after each read; when KIND equals PORT and BLOCKSTRUCTURE equals EXTERNAL, only the data received is placed in the buffer. Blank fill does not occur after a write in either case. The quantity of data actually transferred can be ascertained by interrogating the CURRENTRECORD attribute.

BUFFERS

General, read/write, anytime/closed, integer, not permanent

The attribute BUFFERS specifies the number of buffers assigned to a file. The maximum number of buffers which may be assigned to a file is 63. The default value of BUFFERS for disk files with UPDATEFILE equal to TRUE is 3; the value for all other files is 2. Only in exceptional cases does more than two buffers add to the efficiency of the I/O operations of a file.

The BUFFERS attribute returns the actual number of buffers used if read when the file is open.

CARRIAGECONTROL

Printer, read/write, anytime/closed, integer, not permanent

The attribute CARRIAGECONTROL indicates how the first character in the record of a printer file is to be interpreted.

The values and mnemonics of the CARRIAGECONTROL attribute are as follows:

STANDARD(0)
: This is the default value. STANDARD carriage control makes no attempt to interpret the first character of a record but instead treats it as valid data. CARRIAGECONTROL is handled through the normal <record number or carriage control> syntax of ALGOL and COBOL.

NOTE

Nonstandard carriage control is only allowed for character-oriented (FRAMESIZE not equal to 48 or UNITS equal to CHARACTERS) EBCDIC (INTMODE equal to EBCDIC) Printer files.

CTLASA(1) The line is printed after carriage motion has been completed according to the first character of the record as follows:

Character	Action
" " (blank)	Single spacing
"0" (zero)	Double spacing
"-" (minus)	Triple spacing
"+"	No carriage motion
"1"	Skip to channel 1
"2"	Skip to channel 2
"3"	Skip to channel 3
"4"	Skip to channel 4
"5"	Skip to channel 5
"6"	Skip to channel 6
"7"	Skip to channel 7
"8"	Skip to channel 8
"9"	Skip to channel 9

It is possible to skip to channel 10, 11 or 12. There are, however, no graphic EBCDIC characters that can be used as the first character for those cases. To skip to channel 10, the first 8 bits should be 4"FA", which is a logical extension EBCDIC number set. For channel 11, use 4"FB" and for channel 12, use 4"FC".

CTL360(2) The first character controls paper motion using the various fields in the character as follows:

Field	Action
0:1	If this field is 1, then printing occurs before carriage motion.
1:1	If this field is 1, then no printing will occur, only carriage control.
2:1	Ignored.
6:4	The value of this field is either channel number (if skipping) or the line count (if spacing) according to 7:1.
7:1	If this field is 1, then a skip to channel is the carriage control; otherwise, the printer is spaced the number of spaces specified by 6:4.

CENSUS

Data Comm/Port, read only, open, integer, not permanent

The attribute CENSUS returns the number of messages currently in the input queue of a port file, port subfile, or Data Comm file. If the file is a Data Comm file, the value returned is the total number of messages queued for the file. If the file is a port file and no subfile index is given, the value returned is the sum of the messages queued for all subfiles in the file. If a subfile index is given, the value returned is the number of messages queued for the specified subfile only.

CHANGEDSUBFILE

Port, read only, anytime, integer, not permanent

The CHANGEDSUBFILE attribute returns the index of an arbitrary subfile whose CHANGEEVENT has HAPPENED. If no subfile CHANGEEVENT has HAPPENED for the file, a value of zero is returned.

CHANGEEVENT

Port/Subfile, read only, anytime, event, not permanent

The CHANGEEVENT attribute is caused (set to HAPPENED) whenever the value of FILESTATE for the file changes; CHANGEEVENT is reset (set to NOT HAPPENED) when the FILESTATE attribute for the file or subfile is interrogated. The value of CHANGEEVENT for the file is HAPPENED as long as the value of CHANGEEVENT for any of that file's subfiles is HAPPENED. CHANGEEVENT for the file is reset by the system after all subfile CHANGEEVENTs are reset.

CONTROLMESSAGE

Reader Sorter, write only, open, array, not permanent

This attribute is used to initialize the data area specified by the CONTROLMESSAGE clause in the Data Section of a readersorter UCR. The content of the array will be passed to the UCR.

COPIES

Duplicated disk, read/write, anytime/closed, integer, not permanent

The COPIES attribute accepts or returns the number of copies within a duplicated disk file. The maximum values of COPIES is 15. When a duplicated disk file is created, the default number of COPIES is two (2). Duplicated disk files are created only when the DUPLICATED attribute is TRUE, regardless of the value of the COPIES attribute.

The COPIES attribute returns the number of physical copies of the file if read when the file is assigned.

COPYNAME

Duplicated disk, read only, assigned, pointer, Disk

The attribute COPYNAME returns the external file name of the specified copy of a duplicated file. This attribute may be read only when the logical file is assigned to a physical disk file. The COPYNAME attribute requires the copy number as a parameter. A null value is returned (".") if the file is not a duplicated file or the copy number specified is greater than the number of copies.

When a duplicated disk file is created with the external file name AAA for example, the copies are given external file names of AAA/"COPY#01", AAA/"COPY#02" and so on. The copies of a permanent file are assigned indices each time the file is opened, and the indices are assigned to the copies in the order in which they occur in the directory. Because the file name of any copy may be changed or the copy itself removed, the index assigned to a copy, and even the number of copies, may change for subsequent openings of the file.

Refer to DUPLICATED FILES in Section 4 of this manual for more information regarding duplicated files.

CREATIONDATE

Disk/Tape, read only, assigned, integer, Disk/Tape

The attribute CREATIONDATE returns the creation date of a file. The value is returned as an integer in form YYDDD, where YY is the year, and DDD is the day in Julian form.

When a file is created, "today's date" is always used as the creation date of the file.

CRUNCHED

Disk, read only, assigned, Boolean, Disk

The CRUNCHED attribute indicates whether or not a permanent disk file was closed with CRUNCH, i.e., has returned the unused portion of the last area of the file to the system. When a disk file has been crunched, the available space for the file in its last area is rounded up to a block boundary. A crunched file can be rewritten (or updated) only when the BLOCKSIZE of the logical file is the same as the BLOCKSIZE used to create the file; otherwise, the file is read only. End of file may be extended up to the last block boundary.

When the CRUNCH system option is set, code files and backup disk files are automatically crunched when they are closed. Other disk files can be crunched (and made permanent files if they were not already) by a special close option in each language. For example, in ALGOL, the following equivalent syntax may be used:

```
CLOSE(FILEID,CRUNCH);

LOCK(FILEID,CRUNCH);

LOCK(FILEID, *);
```

CURRENTBLOCK

General, read only, anytime, integer, not permanent

The CURRENTBLOCK attribute returns the size, in logical units (i.e., INTMODE units if it is a character-oriented file, otherwise in words), for the block currently in use. Normally, this value is the same as the value of the attribute BLOCKSIZE. The value of CURRENTBLOCK becomes of interest for a tape file when the system encounters a short block. Refer to the STATE attribute in this section for more information.

CURRENTRECORD

General, read only, anytime, integer, not permanent

The CURRENTRECORD attribute returns the size of the last record read or written. If BLOCKSTRUCTURE does not equal EXTERNAL (FILETYPE does not equal 3), CURRENTRECORD always equals MAXRECSIZE. If the file is open but no I/O action has occurred, or if the file is closed, CURRENTRECORD returns zero.

CYCLE

Disk/Tape, read/write, anytime/anytime, integer, Disk/Tape

The attribute CYCLE, in conjunction with the VERSION attribute, is used to denote the different generations of a permanent file. The initial, and default, CYCLE value is one (1). The maximum value for CYCLE is 9999. The value of the CYCLE attribute may be set only when the file is closed, except in the case of the creation of a disk file, where CYCLE can be changed anytime before the file is entered into the Directory (see the PROTECTION attribute).

Setting the CYCLE attribute to zero (0) has the special effect of resetting both CYCLE and VERSION to their default values of one (1) and zero (0) and marks the logical file as not requiring the specific genealogy checking when the logical file is assigned to a permanent file. In this case, the permanent file with the best genealogy is assigned to the logical file. Best genealogy is defined to be the highest CYCLE and the highest VERSION of that CYCLE.

If CYCLE or VERSION has been set to a legitimate value (this includes explicitly setting one or both of them to their default values), only a permanent file with matching genealogy (along with all the other prerequisites) will be assigned to the file. If the proper file cannot be found, an UNMATCHED GENEALOGY notification is given to the operator. The operator can respond by making the file available, or by the FA or DS System Input Messages.

The CYCLE attribute returns the value for the physical file if read when the file is assigned.

CYLINDERMODE

Diskpack, read/write, anytime/closed, Boolean, Disk

When the attribute CYLINDERMODE is TRUE (and AREALENGTH is less than or equal to the cylinder size), the areas of the diskpack file are assigned so that no area spans a cylinder boundary. The default value is FALSE.

DATE

DATE is a nonpreferred synonym for CREATIONDATE.

TEXT DELETED

DENSITY

Tape, read/write, anytime/closed, integer, Tape

The attribute DENSITY specifies the recording density of a magnetic tape file.

The values, mnenomics and meanings of the DENSITY attribute are as follows:

0 BPI800
1 BPI556
2 BPI200
3 BPI1600
4 BPI6250

BPI556 is not valid for a 9-track tape (TAPE9). BPI1600 is valid only for phase-encoded tapes (PETAPE). The mnemonic BPI16250 is for GCR tapes. There are two exceptions in the use of the DENSITY attribute. In the creation of a multifile tape, the density of the first file is used for all subsequent files. In the creation of a multireel file, the density setting remains constant from reel to reel as long as it is valid for the tape unit. The default DENSITY value is the density setting of the tape unit selected for output files and the density at which the tape was written for input file. Any new DENSITY becomes the hardware required value of the next reel.

The equivalent mnemonics for the DENSITY attribute are supported as nonpreferred synonyms to the above, as follows:

LOW = BPI200
MEDIUM = BPI556
HIGH = BPI800
SUPER = BPI1600

DEPENDENTSPECS

General, read/write, anytime/closed, Boolean, not permanent

If the attribute DEPENDENTSPECS is TRUE when the file is opened, the logical file assumes the structure of the associated physical file. In other words, the attributes BLOCKSIZE, BLOCKSTRUCTURE, FRAMESIZE, MAXRECSIZE, and MINRECSIZE (as well as FILETYPE, SIZEMODE, SIZEOFFSET, SIZE2, and UNITS) assume values equal to those used to create the physical file. If a new file is being created or the associated permanent file is unlabeled, DEPENDENTSPECS is ignored.

If neither DEPENDENTSPECS nor BLOCKSTRUCTURE has been set and FILETYPE equals 7 or 8, FILETYPE determines the structure of the logical file. If DEPENDENTSPECS has not been set and FILETYPE is not equal to 7 or 8, the structure of the logical file is determined by default or by user-specified values for the attributes listed in the preceding paragraph. If DEPENDENTSPECS is specified, its value determines the structure of the file, regardless of the value of FILETYPE. If DEPENDENTSPECS is specified and FILETYPE equals 7 or 8, FILETYPE is assigned a value of 0 when any attempt is made to open the file. An attribute error is reported if DEPENDENTSPECS is specified and an attempt is made to set FILETYPE equal to 7 or 8.

If DEPENDENTSPECS is TRUE for the logical file and, when the file is opened, the FILETYPE of the associated physical file does not correspond to any BLOCKSTRUCTURE value (FILETYPE equals 2, 4, or 5), an attribute error is reported, but FILETYPE for the logical file assumes the FILETYPE value for the physical file, and BLOCKSTRUCTURE reverts to a never set condition.

DIRECTION

Tape/Paper tape reader, read/write, anytime/anytime, integer, not permanent

The attribute DIRECTION indicates the direction in which records will be accessed from a file.

The value and mnemonics of the DIRECTION attribute are as follows:

0 FORWARD

1 REVERSE

The default value is FORWARD.

TEXT DELETED

The DIRECTION attribute is only meaningful for paper tape reader files when the NORVRSPAPERTAPE run-time system option is reset; otherwise, it is ignored.

Direct I/O may change the value of the DIRECTION attribute at any time. When the DIRECTION attribute is changed while a Direct I/O file is open, the direction bit in the I/O control word is also changed. If change in direction, forward to reverse, is concurrent with a change from writing to reading, system label records are written on the tape before the read is initiated.

A file cannot be read in a reverse direction unless BLOCKSTRUCTURE equals FIXED (FILETYPE equals 0) or the file is unblocked.

DISPOSITION

Data Comm, read only, open, integer, not permanent

The attribute DISPOSITION returns the disposition of a station in the Data Comm file. The DISPOSITION attribute requires that an index, the relative station number (RSN), be specified. The values returned and their meanings are as follows:

0 Unknown

1 Assigned

2 Denied assignment

4 Assignment postponed

6 Denied assignment - illegal use attempted

Example:

```
I := DCFILE (FRSN).DISPOSITION;
```

DUPLICATED

Duplicated disk, read/write, anytime/closed, Boolean, Disk

The attribute DUPLICATED indicates whether or not a disk file has area-by-area redundancy maintained by the Input-Output Subsystem. The default value is FALSE. If DUPLICATED is TRUE when creating a new file, the amount of redundancy (the number of copies of the file) is determined by the COPIES attribute. If COPIES is not specified the default value is 2. (Refer to the discussion of the COPIES attribute in this section.) When accessing an existing duplicated file, DUPLICATED must be TRUE for the logical file.

ENABLEINPUT

Data Comm, read only, open, Boolean, not permanent

The attribute ENABLEINPUT returns TRUE if the station (if a Relative Station Number (RSN) was specified) is both assigned and enabled for input, or if any station in the Data Comm file is both assigned and enabled for input.

EOF

Tape, read only, open, Boolean, not permanent

The attribute EOF indicates whether or not (TRUE or FALSE) end-of-file has been reached on a tape file.

Depending on the situation, it may be possible to reset EOF by accessing a valid record. This may be done by reading backward or by backspacing. The EOF attribute is not valid for Direct I/O files. Refer to the IOEOF attribute in this section for the corresponding information concerning Direct I/O files.

ERRORTYPE

Disk, read only, open, integer, not permanent

The attribute ERRORTYPE indicates the type of error that has occurred involving the buffer that is to be used by the next logical I/O.

The values and mnemonics of the ERRORTYPE attribute are as follows:

0 NOERROR (No error occurred)

1 SUNOTREADY (Storage unit not ready)

2 READPARITYERROR (Read parity error)

3 READCHECKFAILURE (Read-check failure)

When a file has more than one buffer, the buffers are rotated after each physical I/O. ERRORTYPE indicates the type of error, if any, that occurred to the top buffer. Refer to the IOINERROR attribute.

For duplicated files, the copy number must be specified and the ERRORTYPE attribute indicates the type of error that occurred on the specified copy.

EXCLUSIVE

Disk, read/write, anytime/unassigned Boolean, not permanent

The attribute EXCLUSIVE allows a program to open a permanent disk file and lock out all other programs and unopened files while the permanent file is open.

When EXCLUSIVE is TRUE, a program opening a permanent disk file implicitly or through the attributes OPEN or PRESENT or by an explicit open will:

a. Get the usual no file action, if the permanent file does not exist.

b. Be suspended. A "WAITING ON" notification is given to the system operator if the file exists but is already opened.

c. Open the file exclusively, locking out all others until the file is closed.

The attribute RESIDENT is unaffected by the setting of the EXCLUSIVE attribute. RESIDENT returns TRUE, if the permanent file exists, whether or not the file can be assigned at that moment. Refer also to the AVAILABLE attribute description.

EXTMODE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute EXTMODE specifies the external or physical character size (mode) of the records in a file. The values and mnemonics are as follows:

0	SINGLE	(Word mode, binary or 48 bit)
2	HEX	(4-bit, packed decimal)
3	BCL	(6-bit)
4	EBCDIC	(8-bit)
5	ASCII	(8-bit)
6	BINARY	(Card files only - 12 bits per column, 20 words per record)

The EXTMODE attribute can be overridden by the physical mode of a permanent file or unit type. The default value for the EXTMODE attribute is the value of the INTMODE attribute, unless the INTMODE value is in conflict with the requirements of the associated physical unit. The EXTMODE of Data Comm files (KIND equal to REMOTE) or console files (KIND equal to SPO) is unconditionally EBCDIC. Card files (READER/PUNCH) can only have BCL, EBCDIC or BINARY for their EXTMODE values.

Whenever the attribute EXTMODE and INTMODE differ, there is a possibility that the data is being translated. Refer to the discussion on software translation.

FAMILY

FAMILY is a nonpreferred synonym for STATIONLIST.

TEXT DELETED

FAMILYINDEX

Disk, read/write, anytime/anytime, integer, Disk

The attribute FAMILYINDEX returns the continuation number of a member of a disk family where the space is allocated. A value of zero (0) indicates that the attribute will not be used in file assignment or area allocation; otherwise these values begin at one (1).

When allocating disk space for a file, the FAMILYINDEX is used to identify the family member from which areas are to be allocated. If the FAMILYINDEX is zero, the disk areas are allocated in the system's normal rotational order. Unless overridden by setting the FAMILYINDEX for a specific area, the value of the FAMILYINDEX attribute when the file is opened is used to allocate each area of the file. The FAMILYINDEX can be set for a specific area, after the file is opened, but only before the disk space is allocated for that area. When a physical file is assigned to the logical file, the FAMILYINDEX requires an index, the area number, as a parameter. If the file is duplicated, the copy number is also required.

Example

```
DISKFILE(ROWNUMBER).FAMILYINDEX := 3;
DUPDISKFILE(ROWNUMBER,COPYNUMBER).FAMILYINDEX := 23;
```

The FAMILYINDEX of areas of a disk file that were assigned by the use of the FAMILYINDEX attribute will be preserved by Library maintenance, but can be overridden when copying the file.

FAMILYINDEX returns the value for the specified area of the physical file if read when the file is assigned.

FAMILYNAME

Disk, read/write, anytime/closed, pointer, Disk

The attribute FAMILYNAME indicates the name (label) of the disk (head-per-track or pack) on which the physical file is located. FAMILYNAME, like the INTNAME attribute, must be a simple identifier less than or equal to 17 characters. If FAMILYNAME is not specified, areas for the disk file are allocated from a family with a label of DISK or PACK depending on the value of the KIND attribute. If KIND equals DISK, the default FAMILYNAME is DISK; if KIND equals PACK, the default FAMILYNAME is PACK. (Refer to the discussion of FAMILYNAME under File-Naming Conventions in Section 4 of this manual.)

If family substitution is in effect for the specified, or default, FAMILYNAME of the file, the substitute or alternate FAMILYNAME is used. (Refer to the FAMILY statement in the B 7000/B 6000 Series CANDE Reference Manual for specific information on family substitution.)

FAMILYNAME returns the name of the family of the physical file if read when the file is assigned.

As of to the 3.3 release, the values of DISK and PACK for the KIND attribute will be synonymous, and the default value for FAMILYNAME will be "DISK" for both disk and disk pack files (KIND equals either DISK or PACK for the logical file). (Refer to the discussion of FAMILYNAME under File-Naming Conventions in Section 4 of this manual.)

To aid the transition to a single default value for FAMILYNAME, the following run-time warning is reported whenever a file from the family named "PACK" is assigned to a logical file with KIND equal to PACK and FAMILYNAME unspecified.

```
ATTRIBUTE ERROR: <intname>. FAMILYNAME @ (<line number>)
ON MARK 33, "FAMILYNAME=PACK" MUST BE SPECIFIED
```

FAMILYSIZE

FAMILYSIZE is a nonpreferred synonym for STATIONCOUNT.

TEXT DELETED

FCM

Reader Sorter, read/write, anytime/closed, real (word), not permanent

This attribute specifies which exception conditions are to be reported to the program. The exception conditions arise in the reader sorter and the RSP. This attribute value is bit-oriented and bits 1 to 10 are either set or not set as desired. Setting a particular bit indicates a desire to receive an exception indication for each occurrence of a specific event. Not setting a bit indicates a desire to not receive any exception indication due to the occurrence of a specific event. Setting a bit means giving it a value of one, not setting is a value of zero.

The bit numbers and meanings are as follows:

Bit 1	The readersorter unit became not ready.
Bit 2	A Black-band item was detected.
Bit 3	A document jam occurred.
Bit 4	A missort was detected.
Bit 5	A too-late-to-read condition occurred.
Bit 6	A too-late-to-pocket-select condition occurred.
Bit 7	Initiation or restart of the UCR was completed.
Bit 8	The UCR requested a stop-flow.
Bit 9	More disk space is needed for the recovery file.
Bit 10	The flow-stopped procedure was completed.

FILEKIND

Disk/Tape, read/write, anytime/anytime integer, Disk/Tape

The FILEKIND attribute describes the internal structure and purpose of a record of a disk file. For a tape file, the FILEKIND attribute describes, in more detail than the attribute LABEL, the tape label format. FILEKIND is read-only when the logical file is assigned to a tape file and returns the FILEKIND of the physical file. FILEKIND can be set before a disk file is opened. It is assigned at the time the file is opened and if the physical FILEKIND is inconsistent with the privilege of the program, a run time attribute error is given. Compilers are privileged since they may change a data file to a code file. Noncompilers are limited to code execution and changing FILEKIND to noncode values.

The values and format of FILEKIND for tape files are as follows:

0	B6500 USASI
1	UNLABELED
2	B5500 STANDARD
3	SYSTEM STANDARD
4	B3500 USASI
5	B3500 STANDARD
6	B6500 BACKUP TAPE
7	B6500 LIBRARY TAPE
8	B6500 LOAD CONTROL TAPE
10	B5500 BACKUP TAPE
11	B5500 LIBRARY TAPE
12	NON-STANDARD USASI
13	USER DEFINED

There are restrictions on FILEKIND changes for disk files; the restrictions may be defined in terms of the following FILEKIND value ranges:

System files:		FILEKIND <		COMPILERCODEFILE
Compilers:		FILEKIND =		COMPILERCODEFILE
Code:	COMPILERCODEFILE <	FILEKIND < OR =		CODEFILE
Data:	ALGOLSYMBOLIC < OR =	FILEKIND < OR =		DBDATA

A user program may change one data FILEKIND to another, and may change anything but a system file to a data file. A compiler may change a data file to a code file. Only the operating system may change or assign a system FILEKIND, or assign a compiler FILEKIND. The default value is DATA.

The values and mnemonics of FILEKIND for disk files are as follows:

0	NULLFILE	37	JOVIALCODE	95	DASDLSYMBOL
1	DIRECTORY	39	ESPOLCODE	96	DMALGOLSYMBOL
	VERSIONDIRECTORY	40	DCALGOLCODE	97	DCPSYMBOL
2	SYSTEMDIRECTORY	41	BASICCODE	98	NDLSYMBOL
3	CATALOG	42	XFORTRANCODE	100	RSNETFILE
4	BACKUPDISK	43	JOBCODE	101	UCRFILE
5	RECONSTRUCTIONFILE	44	DMALGOLCODE	102	RSSORTTABLE
6	SYSTEMDIRFILE	45	NEWPCODE	103	RSPCODE
7	JOBDESCFILE	46	SANSCODE	104	MDLCODE
8	ARCHIVELOG	47	PASCALCODE	105	MDLSYMBOL
15	XDISKFILE	48	RPGCODE	106	VFORTRANCODE
16	BACKUPPRINTER	62	BOUNDCODE	107	VLINKEDCODE
17	BACKUPPUNCH	63	CODEFILE	108	VMCPCODE
18	REMOTEBACKUP	64	ALGOLSYMBOL	109	LCOBOLSYMBOL
19	REMOTEAUDIT	65	COBOLSYMBOL	110	LCOBOLSL3CODE
20	COMPILERCODEFILE	66	FORTRANSYMBOL	111	LCOBOLSL5CODE
21	CHECKPOINTFILE	67	XALGOLSYMBOL	112	SFORTRANCODE
22	CPJOBFILE	68	PLISYMBOL	113	SLINKEDCODE
23	DCPCODE	69	JOVIALSYMBOL	114	SMCPCODE
24	NDLCODE	71	ESPOLSYMBOL	115	FIRMWARE
26	RECOVERYFILE	72	DCALGOLSYMBOL	191	BDDATA
27	SCHEDULEFILE	73	BASICSYMBOL	192	DATA
28	INFOFILE	74	XFORTRANSYMBOL	193	SEQDATA
29	LIBRARYCODE	75	JOBSYMBOL	194	GUARDFILE
30	INTRINSICFILE	77	VFORTRANSYMBOL	195	APLDATA
31	MCPCODEFILE	78	SFORTRANSYMBOL	196	APLWORKSOURCE
32	ALGOLCODE	79	NEWPSYMBOL	197	CDATA
33	COBOLCODE	80	SANSSYMBOL	198	CSEQDATA
34	FORTRANCODE	81	PASCALSYMBOL	199	DBRESTARTSET
35	XALGOLCODE	82	RPGSYMBOL	200	DBDATA
36	PLICODE	94	BINDERSYMBOL		

NOTE

The "type" of a disk file, as defined for CANDE, is similar to the FILEKIND attribute; valid CANDE types are a subset of valid FILEKIND values.

FILENAME

General, read/write, anytime/anytime, pointer, Disk/Tape

The attribute FILENAME is the external file name of a file. Its primary purpose is to identify the physical file. The default value for FILENAME is the value of the INTNAME attribute.

If the KIND attribute for the file equals DISK or PACK, the values of the FILENAME and TITLE attributes may differ. When the file is open, the TITLE attribute includes both the FILENAME and the FAMILYNAME separated by the word ON, whereas the FILENAME attribute does not include the FAMILYNAME. If KIND does not equal DISK or PACK, FILENAME and TITLE are synonymous. Refer to the discussions of the TITLE and FAMILYNAME attributes in this section.

If the file is a disk file, FILENAME may be changed at any time. Changing the FILENAME of a disk file when the file is open also changes the FILENAME of the physical file. The FILENAME of a file that is not a disk file may be changed only when the file is unassigned. FILENAME may be read at any time. If read when the file is assigned, FILENAME returns the value for the physical file.

FILEORGANIZATION

Disk, read/write, anytime/closed, integer, Disk

The FILEORGANIZATION attribute is available so that files may be restricted as to the organization under which they were opened. In order to open the file, the attribute must be set to the value used when the file was created, unless the file is direct. The currently defined values, mnemonics, and meanings of FILEORGANIZATION are as follows:

0	NONRESTRICTED	The file does not adhere to any FILEORGANIZATION restrictions.
1	RELATIVE	The file was created under a RELATIVE file organization and adheres to the restrictions of such. This organization is derived from COBOL74.
2	INDEXED	The file was created under an INDEXED file organization and adheres to the restrictions of such.

The default value is NONRESTRICTED.

FILEORGANIZATION returns the value for the physical file if read when the file is assigned.

FILESECTION

Tape, read/write, anytime/closed, integer, Tape

The attribute FILESECTION is the ISO, BSI and ANSI file section number of the first file header label record. The initial and default value is one (1). The maximum value for FILESECTION is 9999. The FILESECTION attribute is file relative, that is, the value of FILESECTION is incremented only when the file is involved in a reel switch (the data of the file requires more than one physical reel of tape) and the value of FILESECTION is reset to one (1) when the file is closed.

The FILESECTION attribute is used in permanent tape file assignment along with the attributes KIND, TITLE and, when appropriate, CYCLE and VERSION. The FILESECTION attribute is also used in automatic input reel switching.

FILESTATE

General, read only, anytime, integer, not permanent

The FILESTATE attribute returns a value that indicates the logical state of the file. If the file is a Data Comm or port file and more than one subfile exists, FILESTATE requires a subfile index as a parameter.

The mnemonics and meanings of the values returned by FILESTATE are given below. The values CLOSED and OPENED are meaningful for all files; the other values are meaningful only for Data Comm and port files.

Mnemonic	Integer Value	Meaning
CLOSED	0	Indicates the file or subfile is closed.
AWAITINGHOST	1	Indicates that the host specified by the HOSTNAME subfile attribute cannot be reached. The subfile remains in this state until the host can be reached. The value of FILESTATE then changes to OFFERED, OPENED, or CLOSED.
OFFERED	2	Indicates an open operation was attempted on the subfile, and the host specified by HOSTNAME is can be reached. However, no matching subfile was found. All I/O operations to a subfile in this state return end-of-file.
OPENED	3	Indicates the file or subfile is open and ready for input or output.
SHUTTINGDOWN	4	Indicates the system operator wants to terminate communication with the host involved in the subfile dialog. This value provides advanced notice to the program using the port so that it may go to normal end-of-task. However, the port remains open, and all I/O operations are valid.
BLOCKED	5	Indicates that the remote host has become temporarily unreachable. The port remains open and all I/O operations are valid.
CLOSEPENDING	6	Indicates that the user program closed the subfile, but the complementary subfile has not acknowledged the closure. When the complementary subfile does acknowledge closure, the value of FILESTATE changes to CLOSED.
DEACTIVATIONPENDING	7	Indicates that the complementary subfile closed, and the local subfile has queued input. All write operations return end-of-file, but all read operations are valid.

Mnemonic	Integer Value	Meaning
DEACTIVATED	8	Indicates that the complementary subfile closed, and the local subfile has no queued input. A close is the only valid operation for a subfile in this state; all read and write operations return end-of-file.
DENIED	9	Indicates that file assignment was denied.
POSTPONED	10	Indicates that file assignment was postponed.
DENIEDILLEGALUSE	11	Indicates that an illegal open was attempted.

FILETYPE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute FILETYPE specifies the format of the records and the structure of the file.

The attributes BLOCKSTRUCTURE and DEPENDENTSPECS can override and change the value of FILETYPE. Refer to the discussions of BLOCKSTRUCTURE and DEPENDENTSPECS in this section.

The values and meanings of the FILETYPE attribute are as follows:

0 Blocked or unblocked fixed-length records. This value is the default value for FILETYPE.

1 Variable-length records. The record length is contained in the first four INTMODE characters of the record and expressed in decimal characters. If INTMODE equals SINGLE (words), then the record length is stored in the first word of the record in binary.

2 Variable-length records. The record length contained in the first two INTMODE characters of the record. The record length is expressed in binary. If INTMODE equals SINGLE (words), then the record length is stored in the first word of the record.

3 Variable-length records. Neither the record itself nor the structure of the file contains information about the length of the record; this must be specified by the I/O statement. Unblocked tape and Data Comm files are able to discover and return the true length of the record. Refer to the discussions of the CURRENTRECORD and STATE attributes in this section.

4 Variable-length records. The record length is contained in a fixed location in the record. The attributes SIZEOFFSET, SIZE2, and SIZEMODE determine the position of the record length information, the size of the field containing the information, and the mode of the information.

5 XALGOL-linked variable-length records. A link word (maintained by the Input-Output Subsystem) between each record specifies the length, in INTMODE units, of the previous record (first half of the link word) and the length, in INTMODE units, of the record in use (second half of the link word). Except in the case where the first backward link and the last forward link have length of zero, the record length information includes the size of the link word.

NOTE

As of the 3.4 release, files with FILETYPE equal to 5 will no longer be supported. Currently, the following message is given whenever a file is opened with FILETYPE equal to 5:

ON 34, "FILETYPE=5" FILES WILL NO LONGER BE SUPPORTED.

6 FORTRAN-linked variable-length records. The link words are maintained by the Input-Output Subsystem and are not part of the records. The record's mode is assumed to be SINGLE.

7 The format of the records and the structure of the logical file are to be determined by the structure of the associated permanent file; that is to say, the attributes FILETYPE, MINRECSIZE, MAXRECSIZE, BLOCKSIZE, SIZEOFFSET, SIZE2, SIZEMODE, UNITS, and INTMODE will be changed to the values used in the permanent file. The INTMODE attribute is changed to the mode of the permanent file, which was, in fact, the EXTMODE value of the creating logical file. If no permanent file is associated with the logical file (i.e., a new file is being created), FILETYPE will be set to zero (0) (or three (3), depending upon the initial value of MINRECSIZE), and all the above-mentioned attributes will be set to their default values if they had not been explicitly set before the file was opened.

8 The action is the same as for FILETYPE equal seven (7), with the following exceptions: the INTMODE value of the logical file is not changed when a permanent file is associated with the logical file; and if a new file is being created, all the attributes (FILETYPE, MINRECSIZE, MAXRECSIZE, BLOCKSIZE, SIZEOFFSET, SIZE2, SIZEMODE, and UNITS) are reset to their default values.

If FILETYPE is zero, and MINRECSIZE is greater than zero but less than MAXRECSIZE, then FILETYPE will be changed to three (3) when the file is opened.

The blocking technique used with variable length blocked files is as follows: if the length of the record just written plus MINRECSIZE plus the offset into the block is greater than BLOCKSIZE, or the record to be written plus the offset into the block is greater than BLOCKSIZE, a physical write is initiated for the block. In the later case, the record becomes the first record of the next block. If the logical file is assigned to a peripheral unit (KIND) that allows variable length blocks (e.g., tape files), then only the part of the block that is valid will be written. For peripheral units that require fixed length physical blocks (e.g., disk files), a MINRECSIZE field of nulls (4"00") will be added to the block as an end-of-block marker and the whole block will be written.

FILETYPE 5 and 6 are always blocked. All other FILETYPE's will be blocked only when BLOCKSIZE is greater than MAXRECSIZE.

The value of FILETYPE has an effect upon the default value of MINRECSIZE. Variable length files which have link words or record length fields contained within the record require MINRECSIZE to be at least large enough to hold this information. Files with FILETYPE equal to zero use a value for MINRECSIZE that is equal to MAXRECSIZE.

TEXT DELETED

Attempting to specify a space to an externally specified variable length record (FILETYPE=3) causes an ILLEGAL SEEK error.

FILEUSE

General, read/write, anytime/closed, integer, not permanent

The FILEUSE attribute specifies how the file will be used. The mnemonics of the FILEUSE attribute are as follows:

IN - input only
OUT - output only
IO - both input and output

The default value is IO. If the value of FILEUSE is either IN or OUT, a write or a read, respectively, will cause a runtime error: either WRITE ON INPUTFILE or READ ON OUTPUTFILE.

Unlike the MYUSE attribute, the values of the FILEUSE attribute have no effect on the file search algorithms or on the setting of the UPDATEFILE attribute.

FLEXIBLE

Disk, read/write, anytime/anytime, Boolean, not permanent

The attribute FLEXIBLE indicates whether or not a disk file can be allocated more areas, if needed, than the number originally specified by the AREAS attribute. The default value for FLEXIBLE is TRUE. If FLEXIBLE is TRUE and more areas are needed, room for a minimum of ten more is allocated. The value of the FLEXIBLE attribute is not retained by the physical file and the action of the Input-Output Subsystem is controlled by the setting of the attribute in the logical file that currently has the file open. The setting of FLEXIBLE is ignored if the file has been crunched, is duplicated, or is IAD.

FORMID

Printer/punch, read/write, anytime/closed, pointer, not permanent

If the attribute FORMID is set, the message assigned to the attribute is displayed on the operator's console at file open time. The program is suspended until the operator responds with an FM or OU System Input Message. If the file goes to backup, the message is displayed just prior to the printing or punching of the backup file and not during creation of the backup file.

The FORMID attribute accepts a string of 15 characters or less, terminated with a period.

The FORMID attribute is not valid for input files. FORMID may be read anytime and will return a null string (".") if it is not set. It may be reset by assigning a null string.

FORMMESSAGE

FORMMESSAGE is a nonpreferred synonym for FORMID.

TEXT DELETED

FRAMESIZE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute FRAMESIZE specifies the number of bits transferred as a unit of data during an I/O operation on a file. The attributes MINRECSIZE, MAXRECSIZE, BLOCKSIZE, AREALENGTH, CURRENTRECORD, and CURRENTBLOCK are expressed in FRAMESIZE units.

The possible values for FRAMESIZE are as follows:

- 4 Data is transferred in units of 4 bits or as hexadecimal characters. INTMODE must equal HEX.
- 8 Data is transferred in units of 8 bits or as EBCDIC or ASCII characters. INTMODE must equal EBCDIC or ASCII.
- 48 Data is transferred in units of 48 bits or as full words. This value of FRAMESIZE is compatible with all values of INTMODE.

If FRAMESIZE and INTMODE are not compatible when the file is opened, an open error is reported.

If FRAMESIZE is unspecified, either UNITS or the default for UNITS is used. If FRAMESIZE is specified, it is used regardless of the value of UNITS, and UNITS is changed to a value compatible with FRAMESIZE when the file is opened. An attribute error is reported if FRAMESIZE is specified and an attempt is made to specify UNITS.

If FRAMESIZE is read when the file is open, the value is computed from the physical file and returned even if FRAMESIZE was not specified for the logical file.

GENERATION

Disk/Tape, read/write, anytime/closed, integer, not permanent

Files with a common TITLE and family of creation, but which vary in time of update, CYCLE, or VERSION are grouped in the System Catalog as different generations of the same file. The GENERATION attribute can be used to select a copy of the file other than the one with the best genealogy (see the CYCLE attribute) and latest time of update. The larger the value of GENERATION, up to the limit of the references in the Catalog, the less recent the update (or creation) and lower the genealogy.

HOSTNAME

General, read/write, anytime/anytime, pointer, not permanent

HOSTNAME is a pointer-valued attribute used during file assignment to specify the host on which the physical file or complementary port file exists.

For files other than port files, HOSTNAME may only be set when the file is closed and unassigned.

HOSTNAME may be read at any time. If the file is closed and unassigned and HOSTNAME has never been set, it returns the name of the local host. If the file is assigned, HOSTNAME returns the name of the host at which the file resides.

IAD

Disk, read only, anytime, Boolean, Disk

The IAD attribute may be used to determine if the file is an Installation Allocated Disk file rather than a normal disk file.

NOTE

Installation Allocated Disk is not supported on B 6900 and later systems.

IMAGEFILEID

Reader Sorter, read/write, anytime/anytime, pointer, not permanent

IMAGEFILEID specifies the TITLE of the disk file containing image matching records. See also IMAGEFILEMAP attribute.

IMAGEFILEMAP

Reader Sorter, write only, open, array, not permanent

This attribute selects the image file records for use in image-matching by the UCR. The array must contain at least one, and not more than 500, number pair(s).

Each number pair specifies a contiguous range of records beginning with the first number and ending with the second number inclusively. The final pair must contain the value zero for beginning and ending numbers. The maximum number of records spanned by a single pair is 65535.

The IMAGEFILEMAP is sent to the RSP at UCR load time, restart of flow for an unmasked FCM condition, or when the IMAGEFILEMAP has been changed.

INPUTEVENT

Port/Subfile, read only, anytime, event, not permanent

When a subfile index is given, the INPUTEVENT attribute returns HAPPENED if the CENSUS attribute for the specified subfile is greater than zero. When no subfile index is given, the INPUTEVENT attribute returns HAPPENED if the CENSUS attribute for the file is greater than zero.

Example:

```
SF(2).INPUTEVENT
F.INPUTEVENT
```

INPUTTABLE

General, write only, anytime, translatetable, not permanent

Setting the attribute INPUTTABLE allows the program to modify the Input-Output Subsystem's input software translation. INPUTTABLE is only used when the TRANSLATING attribute is TRUE; otherwise, the translate table (if one was assigned) is ignored. A translate table assigned to INPUTTABLE which is declared in a program must either be the first table in a list of translate tables, or the only table in that declaration. Each time the logical file is closed, INPUTTABLE reverts to its default system value. An example of the ALGOL syntax for assigning a translate table to INPUTTABLE is as follows:

```
FILEID.INPUTTABLE := ASCIITOBCL;
```

INTERCHANGE

Diskpack, read/write, anytime/closed, Boolean, Diskpack

When TRUE, the INTERCHANGE attribute indicates that the file is stored or to be stored on a Burroughs interchange pack. A Burroughs interchange pack is a disk pack that has been initialized to a multisector format that allows disk pack compatibility among Burroughs systems.

The default value is FALSE.

See the discussion of disk files in the section on physical files and related attributes for the restrictions imposed upon Interchange packs.

INTMODE

General, read/write, anytime/closed, integer, not permanent

The attribute INTMODE specifies the internal or logical character size (mode) of the records in a file. The values and mnemonics are as follows:

0	SINGLE	(Word Mode, Binary or 48-Bit)
2	HEX	(4-Bit, packed decimal)
3	BCL	(6-Bit)
4	EBCDIC	(8-Bit)
5	ASCII	(8-Bit)

The default value for the INTMODE attribute is EBCDIC. This default can be overridden by the dollar option BCL in FORTRAN, COBOL, ALGOL, etc. Whenever the attributes INTMODE and EXTMODE differ, there is a possibility that the data is being translated. Refer to the discussion on software translation. INTMODE, in combination with the attribute FRAMESIZE (or UNITS), determines whether the data transfer is word or character oriented. Refer to the discussions of the FRAMESIZE and UNITS attributes in this section.

INTNAME

General, read/write, anytime/unassigned, pointer, not permanent

The attribute INTNAME is the internal file name. File label equation is accomplished by matching the internal file name to the file named on control cards. The compiler generated default internal file name is the identifier (up to 17 characters) used by the program in the file declaration. The INTNAME attribute can be programmatically changed to any simple identifier less than or equal to 17 characters (i.e., A/B is not a valid internal name). The attribute may only be set while the logical file is closed and not assigned to a physical file. The INTNAME attribute cannot be set by Work Flow Language. When the internal name of a file is changed, label-equation action is initiated using the new internal name.

Setting the INTNAME file attribute via a multiple file attribute assignment, or by single attribute assignment, will cause label-equation statements for the new INTNAME to be processed. The following example will attempt to open a disk file:

Example:

```
?COMPILE X ALGOL GO;
?FILE NEWINT (KIND=DISK);
?DATA
BEGIN
FILE F (KIND=TAPE);
F (MAXRECSIZE=22, INTNAME="NEWINT.");
F.OPEN:=TRUE;
END.
?END
```

IOCLOCKS

General, read only, assigned, real, not permanent

The attribute IOCLOCKS returns the accumulated I/O time for the file in units of 2.4 microseconds.

IOINERROR

Disk/Tape, read only, open, Boolean, not permanent

If the attribute IOINERROR returns TRUE, there has been an error in physical I/O on the file. If the file has more than one buffer (the usual case) and a logical I/O statement initiates a physical I/O, the physical I/O is completed asynchronously with the object program. When a physical I/O is initiated, the buffers are rotated so that the object program can continue using the next buffer. The error analysis for the physical I/O is done when the buffer is rotated back to the top position. Error analysis for logical and physical reads coincides but the logical I/O result descriptor (see the STATE attribute) returned by the read statement includes the information to be gained from IOINERROR and is probably more useful. Error analysis for physical writes is always the number of buffers behind the position of the logical file. If IOINERROR returns TRUE, the attributes ERRORTYPE and RECORDINERROR can be consulted for more information.

For duplicated disk files, IOINERROR returns TRUE, if any copy of the file encountered an error associated with the buffer.

KIND

General, read/write, anytime/closed, integer, not permanent

The attribute KIND describes the peripheral unit associated with the logical file.

The values and mnemonics of the KIND attribute are as follows. The first mnemonic for each value is preferred.

0	DONTCARE
1	DISK or SERIAL
2	SPO
3	REMOTE or DC
4	PAPERREADER or PAPER or PTR
5	PAPERPUNCH or PTP
7	PRINTER
8	READERSORTER or RS
9	READER
11	PUNCH or CP
13	TAPE7
14	TAPE9
15	TAPEPE or PETAPE
17	PACK or DISKPACK
19	PORT
45	TAPE

The value DONTCARE is converted to TAPE7 when the file is opened in a manner such that a new file is created.

The KIND attribute returns the value for the physical file, if read when the file is assigned.

As of the 3.3 release, the values DISK and PACK will be synonymous. (Refer to FAMILYNAME under File-Naming Conventions in Section 4 of this manual for a discussion of the ramifications of this change.)

LABEL

General, read/write, anytime/unassigned, integer, not permanent

The attribute LABEL indicates whether or not the file has a label record.

The values and mnemonics are as follows:

0	STANDARD
1	OMITTED
3	OMITTEDEOF

If LABEL is equal to STANDARD when creating a tape file, the logical I/O subsystem writes ANSI (USASI) labels as the beginning and ending records of the file, succeeded and preceded, respectively, by tape marks. If the value is equal to OMITTED or OMITTEDEOF, the label records are not included. For a printer file, if the value is STANDARD, the file is preceded by a skip to channel one; no action is taken if the value is OMITTED or OMITTEDEOF. Labeled punch files use the format discussed under card files. Note, however, that only Direct I/O files can create an unlabeled punch file. The value of LABEL is ignored by Data Comm files and the value of OMITTED is used.

When reading a tape file, the differences between OMITTED and OMITTEDEOF become important. When reading an unlabeled tape and a tape mark is encountered, if LABEL equals OMITTED, a reel switch is attempted; if LABEL equals OMITTEDEOF, an end-of-file (EOF) action occurs.

For paper tape files, the value of OMITTEDEOF functions similar to unlabeled tape. Reelswitch is not attempted and end-of-file is returned to the program. If OMITTED is specified, reel switching is dependent upon the operator response to an FR message.

If a file is closed retaining a physical file, setting LABEL gives an error.

LABELTYPE

LABELTYPE is a nonpreferred synonym for LABEL.

TEXT DELETED

LASTRECORD

Disk, read/write, assigned/assigned, integer, Disk

The attribute LASTRECORD returns the record number of the last record in the file, calculated in terms of the blocking of the logical file. LASTRECORD may not be correct during periods when the file is being expanded, because the end-of-file calculations are made only when necessary, during the transition

from writing to reading, for example. The counting of records is zero relative; that is to say, if there is only one record in the file, LASTRECORD returns zero (0). LASTRECORD returns minus one (-1) for a null (empty) disk file.

The LASTRECORD attribute may be set by a program if and only if PROTECTION is not equal to PROTECTED, DUPLICATED and CRUNCHED are FALSE, the FILEKIND is DATA, DBDATA, or DBRESTARTSET, the logical file is closed with retention, and no other logical file is currently assigned to the physical file.

No space is allocated or deallocated by changing the value of this attribute.

LASTSTATION

LASTSTATION is a nonpreferred synonym for LASTSUBFILE.

TEXT DELETED

LASTSUBFILE

Data Comm/Port, read/write, open/open, integer, not permanent

If the file is a port file, the LASTSUBFILE attribute is read only and returns the index of the subfile used for the last successful I/O operation. The value of LASTSUBFILE is updated only upon completion of a successful I/O operation.

If the file is a Data Comm file used for normal (non-Direct) I/O, LASTSUBFILE returns the Relative Station Number (RSN) of the station from which the last input was received (that is, the last record read by the object program), unless LASTSUBFILE was explicitly set by the program. Output is always directed to the station specified by the LASTSUBFILE attribute.

Valid RSNs are greater than zero. A write statement when LASTSUBFILE is zero, for normal (non-Direct) files, is broadcast to every station assigned to the file. If there is only one station assigned to the file (POPULATION equals 1), then a broadcast write is the same as a write directed to the single station. The attribute LASTSUBFILE can also be set by using the [STATION <aexp>] form of the <record number or carriage control> part of the ALGOL write statement.

Direct I/O Data Comm files do not use LASTSUBFILE to determine the originating RSN. The IORECORDNUM buffer attribute is used since it is associated with the particular buffer rather than the file. The output RSN is specified by IORECORDNUM when LASTSUBFILE is zero, and by LASTSUBFILE when LASTSUBFILE is nonzero. A write with an explicit setting of LASTSUBFILE can be used to send a broadcast as well as specifying a specific RSN.

For example: (in ALGOL)

"WRITE(<file id>[STATION 0],.. .. ..)"

The default value of LASTSUBFILE is zero.

LINENUM

Printer, read/write, anytime/anytime, integer, not permanent

The attribute LINENUM indicates the current line number of the logical page, as defined by the PAGESIZE attribute. The attribute is only meaningful for printer files where the PAGESIZE attribute has been set greater than zero (0).

LINENUM may be set to any value between zero (0) and 255. The default value is zero. If the PAGESIZE attribute is nonzero when the printer file is opened, LINENUM is initialized to one (1).

The [LINE <aexp>] form of an ALGOL write statement is provided for positioning the output page. The action taken depends on the values of <aexp>, PAGESIZE, and LINENUM. The [LINE <aexp>] form of the ALGOL write statement temporarily suspends the usual action of space after printing and spaces forward to the logical line specified by <aexp> before printing.

If <aexp> is greater than PAGESIZE, the page is spaced forward, if necessary, to the end of the logical page before the line is printed. No spacing occurs after the line is printed. An end-of-page is returned to the program, LINENUM is set to 1, and PAGE is incremented.

If <aexp> is greater than or equal to LINENUM and less than or equal to PAGESIZE, the page is spaced forward, if necessary, to the logical line specified by <aexp> before the line is printed. No spacing occurs after the line is printed. LINENUM is set to <aexp>.

If <aexp> is less than LINENUM, the line is printed without spacing before or after. LINENUM is set to 1, PAGE is incremented, and an end-of-file is returned to the program.

The [SKIP <aexp>] form of the ALGOL write statement is equivalent to a skip to channel on the carriage control tape. A skip to channel may affect LINENUM. A skip to channel 1 causes LINENUM to be updated to 1 after the skip. A skip to any other channel does not update LINENUM; therefore, after such a skip, LINENUM may not indicate the actual position on the page.

When the carriage control is SPACE, the page is spaced forward the number of lines specified. LINENUM is incremented by the number of lines spaced and if the sum at least equals PAGESIZE, LINENUM is set to 1, PAGE is incremented, and an end-of-page is returned to the program.

Every serial write statement increments LINENUM (while PAGESIZE is greater than zero). A serial write that results in end-of-page will have been printed. An end-of-page does not cause the next write statement to start on the top of the next physical page. Any special action after the end-of-page result must be done by the object program. After an end-of-page result, LINENUM is set to 1 and PAGE is incremented.

The LINENUM attribute can be set at anytime to a value between 0 and 255. However, setting LINENUM has no effect on positioning of the page. It can only cause end-of-page detection to occur at a different time. Setting LINENUM to a value at least equal to PAGESIZE will cause the next write to return end-of-page, LINENUM to be set to 1, and PAGE to be incremented.

The end-of-page result has the same format as an end-of-file result (see the STATE attribute).

MAXCENSUS

Subfile, read/write, anytime/anytime, integer, not permanent

The value of the MAXCENSUS attribute specifies the number of input messages that may be queued for a subfile before the complementary subfile returns NO BUFFER AVAILABLE. The default, and maximum, value of MAXCENSUS is 63.

MAXRECSIZE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute MAXRECSIZE specifies the maximum size of records in the logical file. If FRAMESIZE does not equal 48 (or UNITS equals CHARACTERS), then MAXRECSIZE is expressed in INTMODE units, otherwise in words. MAXRECSIZE can be assigned a value between zero (0) and 65,535. If MAXRECSIZE is left unset or set to zero, a default value is assigned when the file is opened. The default value depends upon the assigned peripheral unit (KIND) and the value of the attribute BLOCKSIZE.

If the file is a port file, the value of MAXRECSIZE specifies the maximum text size for all subfiles in the port file. As each subfile is opened, the value of MAXRECSIZE for that subfile is adjusted so that it is compatible with the value of MAXRECSIZE for the complementary subfile it is assigned to. (Refer to Port File Input/Output Operations in Section 2 of this manual.) The actual value of MAXRECSIZE for a specific subfile may be ascertained (but not altered), when the subfile is open, by interrogating MAXRECSIZE using the subfile index as a parameter.

The attributes MAXRECSIZE, MINRECSIZE, FILETYPE, and KIND are closely related.

TEXT DELETED

MAXRECSIZE must be less than or equal to BLOCKSIZE. If BLOCKSTRUCTURE equals FIXED (FILETYPE equals 0), BLOCKSIZE must be a multiple (possibly 0 or 1) of MAXRECSIZE. If MINRECSIZE is set greater than MAXRECSIZE, it will be set to MAXRECSIZE. If FILETYPE is equal to 0 and MINRECSIZE is greater than 0 and less than MAXRECSIZE, FILETYPE is set to 3 (unless BLOCKSTRUCTURE has been set). If MAXRECSIZE is equal to 0 and BLOCKSIZE is greater than 0, the value of MAXRECSIZE is set to the value of BLOCKSIZE.

If both MAXRECSIZE and BLOCKSIZE are equal to zero when the file is opened, the default value depends upon the type of peripheral unit assigned to the file. The default values are as follows:

KIND	Default MAXRECSIZE - BLOCKSIZE
DISK or PACK	30 words.
SPO	10 words.
REMOTE	12 words.
PAPERREADER	10 words.
PAPERPUNCH	10 words.
PRINTER	If EXTMODE is BCL, then 17 else 22 words.
CARDREADER or CARDPUNCH	If EXTMODE is BINARY, then 20 else if EXTMODE is EBCDIC, then 14 else 10 words.
TAPE	10 words.
READERSORTER	10 words.
PORT	320 words.

If the FRAMESIZE attribute does not equal 48 (UNITS equals CHARACTERS), these values are multiplied by the number of characters per word, as specified by the INTMODE attribute. Card files (READER or PUNCH) are an exception; if the file is character oriented, the maximum value is 80 characters.

MAXSUBFILES

Port, read/write, anytime/closed, integer, not permanent

The MAXSUBFILES attribute specifies the maximum number of subfiles that may be opened for the file. The subfiles are assigned indices from one to MAXSUBFILES, inclusive. If MAXSUBFILES is unspecified and the file is a port file, it is assigned a value of one when the file is opened. If the file is opened after setting MAXSUBFILES to a value less than one, an attribute error is reported.

MINRECSIZE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute MINRECSIZE specifies the minimum size of records in the logical file. If the FRAMESIZE attribute does not equal 48 (UNITS equals CHARACTERS), then MINRECSIZE is expressed in INTMODE units; otherwise, in words. MINRECSIZE may be assigned a value between zero (0) and 65,535. If MINRECSIZE is left unset or is set to zero, a default value will be assigned when the file is opened, which depends upon the value of the attribute MAXRECSIZE. If MINRECSIZE was set greater than the value of MAXRECSIZE, it will be reset equal to MAXRECSIZE.

The minimum record size used by the Input-Output Subsystem for deblocking the file is determined from the maximum of: MINRECSIZE, the physical minimum record size (the value used when the file was created), and the minimum allowable record size (which is dependent upon the values of the attributes BLOCKSTRUCTURE and FILETYPE).

Files with BLOCKSTRUCTURE equal to FIXED (FILETYPE greater than zero) require that the minimum record size be large enough to contain the link word or record length information. Refer also to the discussion of the FILETYPE and MAXRECSIZE attributes.

MYNAME

Port, read/write, anytime/closed, pointer, not permanent

The MYNAME attribute is used during the subfile matching process. The value of MYNAME for the local subfile must match the value of YOURNAME for the complementary subfile before the file can be opened. The default value of MYNAME is a null string ("."). Refer to Port Subfile Assignment in Section 4 of this manual.

MYUSE

General, read/write, anytime/closed, integer, not permanent

The attribute MYUSE specifies how the file will be opened. The values and mnemonics of the MYUSE attribute are as follows:

0	CLOSED
1	IN
2	OUT
3	IO

If the file is not explicitly opened, then unless the MYUSE value is IO, the opening I/O action (READ or WRITE statement) determines the setting of the MYUSE attribute (IN or OUT respectively). If the opening I/O action is a SEEK or SPACE statement, or if MYUSE has not been set and the file is opened explicitly, MYUSE is set to IN (unless the assigned peripheral unit (KIND) is an output-only device, in which case MYUSE is set to OUT).

As of the 3.4 release, the MYUSE attribute will no longer imply a request for the update method of access when the logical file is assigned to a disk file. As of the 3.4 release, only the value of the UPDATEFILE attribute will determine the method of access, and a value of IO for the MYUSE attribute will specify only that the file may be used for both input and output. (Refer to the discussion of the UPDATEFILE attribute in this section.)

To aid this transition, a warning message and attribute error message will be displayed whenever a disk file is opened with MYUSE equal to IO and UPDATEFILE unspecified. The warning given when executing COBOL programs is as follows:

```
PROGRAM MUST BE RECOMPILED BEFORE 34 TO CONTINUE
UPDATE I/O ACTION.
```

The warning given when executing other programs is as follows:

```
ON 34, "UPDATEFILE=TRUE" MUST BE SPECIFIED FOR
UPDATE I/O ACTION.
```

The attributes KIND, NEWFILE, MYUSE, and (prior to the 3.5 release) AREASIZE determine whether a new file is created or a permanent file is assigned when the file is opened. Refer to New File vs. Permanent File in Section 4 of this manual for more specific information.

If the KIND attribute equals REMOTE (Data Comm file) then MYUSE equal OUT means output only. MYUSE equal IN or IO both allow input and output on Data Comm files. See the discussion of Data Comm files.

Once the file is opened, except in the case of files declared in a COBOL program segment, the value of MYUSE is ignored by the logical subsystem.

NEWFILE

General, read/write, anytime/closed, Boolean, not permanent

The attribute NEWFILE has three states: never-set, TRUE and FALSE. If NEWFILE has never been set and KIND equals DISK or PACK, then the values of the MYUSE and (prior to the 3.5 release) AREASIZE attributes determine whether a permanent file will be assigned or a new file created. (Refer to New File vs. Permanent File in Section 4 of this manual.) If NEWFILE is set TRUE, then a new file will be created irrespective of the settings of other attributes. If NEWFILE is set FALSE, then an existing file will be sought, irrespective of the settings of other attributes. An open error results if NEWFILE is set incompatibly with the device type, such as FALSE for a PRINTER or TRUE for a READER.

The NEWFILE attribute may be read at any time. The value is TRUE if NEWFILE has been set TRUE. If the attribute has not been set, the value is FALSE whether or not a new file was created according to the default criteria.

Tape labels are written when a tape is opened with NEWFILE=TRUE and MYUSE not specified. Also, if MYUSE is unspecified and NEWFILE is TRUE, and the device is an I/O device (TAPE or DISK), then MYUSE is set to OUT.

NORESOURCEWAIT

Disk, read/write, anytime/anytime, Boolean, not permanent

This attribute applies to Direct I/O files only.

The attribute can be set or reset at any time. The default setting is FALSE; if set to TRUE, a program using the file will not wait when calling "GETUSERDISK" to get a new row or to activate an old row. If disk is not available or the appropriate continuation pack is not present, the program will get an immediate error result for that I/O. If the error occurs, BUFFER.IORESULT will be 4"101", BUFFER.IOERRORTYPE will be 5, and BUFFER.IOEOF will be true. For nondisk files the attribute will have no effect. The following conditions may still cause the program to wait:

a. A no file condition on open because the AVAILABLE attribute was not used.

b. The required removal of another file with same title in close or open if the file is protected when the option AUTORM is not set.

c. Open of a protected file may cause a wait.

d. Close lock on a temporary file may cause segments to be required on the directory.

e. Interchange pack files may wait if the pack goes not ready after the first row on that pack has been activated but before all rows on that pack have been activated.

f. Interchange pack files may wait on segments for the header if getting the first row for a file on a given continuation pack.

g. Flexible files may have to wait on directory segments when stretching the header.

h. If a pack goes not ready during an I/O, IOERROR will wait on a RSVP which will cause the I/O to wait also.

OPEN

General, read/write, anytime/anytime, Boolean, not permanent

The attribute OPEN indicates whether or not (TRUE or FALSE) the file is open.

If the file is closed, setting OPEN to TRUE will open the file. If the file is open, setting OPEN to FALSE will close the file and retain the peripheral unit (equivalent to CLOSE (FILEID,REWIND) in ALGOL or CLOSE FILEID in COBOL). If the file cannot be opened following the setting of the OPEN attribute to TRUE, an attribute error is given.

OPTIONAL

General, read/write, anytime/closed, Boolean, not permanent

The attribute OPTIONAL indicates whether or not (TRUE or FALSE) the assignment of a permanent file is optional. If the permanent file described by the attributes TITLE, KIND, etc. is not present when the file is opened, a "NOFILE" message is sent to the operator's console and the program is suspended. If OPTIONAL is TRUE, then the operator may respond with an OF system input message, and the program proceeds without a physical file assigned to the logical file. If OPTIONAL is FALSE and the appropriate file is not present, then the operator may be able to respond with an FA or DS System Input Message.

Nonpresent optional files are input-only files. All read statements result in end-of-file action (or in the case of Direct I/O, the I/O is cancelled). Writing on the file is illegal and will cause program termination.

If it is desired to open a permanent file only if it is present and available, either the AVAILABLE or PRESENT attributes should be used to open the file.

The default value for OPTIONAL is FALSE.

OUTPUTEVENT

Port, read only, anytime, event, not permanent

The OUTPUTEVENT attribute returns HAPPENED when output buffers are available and NOT HAPPENED when no output buffers are available.

OUTPUTTABLE

General, write only, anytime, translatetable, not permanent

Setting the OUTPUTTABLE attribute allows the program to modify the Input-Output Subsystem's output software translation. OUTPUTTABLE is only used when the TRANSLATING attribute is TRUE; the translate table (if one was assigned) is otherwise ignored.

A translate table assigned to OUTPUTTABLE which is declared in a program must either be the first table in a list of tables, or the only table in that declaration. Each time the logical file is closed, OUTPUTTABLE reverts to its default system value.

An example of the ALGOL syntax for assigning a translate table to OUTPUTTABLE is as follows:

```
FILEID.OUTPUTTABLE : = BCLTOASCII;
```

PACKNAME

PACKNAME is a nonpreferred synonym for FAMILYNAME.

TEXT DELETED

PAGE

Printer, read/write, anytime/anytime, integer, not permanent

The attribute PAGE indicates the number of the current logical page of a printer file. The attribute PAGE can be set between zero (0) and 65,535, inclusively. The default value is zero (0). Changing the value of PAGE has no effect upon the output of the current physical page. If the PAGESIZE attribute is nonzero, when the printer file is opened, PAGE is initialized to one (1). As long as PAGESIZE is nonzero, whenever the LINENUM attribute becomes greater than or equal to PAGESIZE, the attribute PAGE is incremented by one (1), LINENUM is reset to one (1) and the write statement returns an end-of-page result (which has the same format as end-of-file).

PAGESIZE

Printer/Data Comm, read/write, anytime/anytime, integer, not permanent

The attribute PAGESIZE indicates the number of lines on a logical page. PAGESIZE can have a value between zero (0) and 255, inclusively. When PAGESIZE is nonzero for a printer file, the Input-Output Subsystem maintains the values of the PAGE and LINENUM attributes (see their discussions). When PAGESIZE is zero for a printer file (this is the default case), the attributes PAGE and LINENUM are ignored.

When PAGESIZE is nonzero, the Input-Output Subsystem suppresses the normal skip to channel 1 that occurs after sensing a channel 12 in the carriage control tape. Skipping to channel 1 must be done in the user program and the Input-Output Subsystem will supply an end-of-page result, at the appropriate time, to signal the program that it has filled a logical page.

After the file is open, PAGESIZE cannot be changed from or to a value of zero. It may be altered to any value between one (1) and 255 if it is not zero.

When PAGESIZE equals 255, carriage actions are the same as for other nonzero PAGESIZE settings, except (1) end-of-page is not returned to the program, (2) PAGE is not incremented, and (3) LINENUM is counted continuously, wrapping around from 255 to 0. The automatic skip to channel 1 after channel 12, however, is still suppressed.

For a Data Comm file (KIND = REMOTE) the PAGESIZE attribute is read only, and only when the file is open. PAGESIZE is a station attribute (that is to say, to access the attribute, a Relative Station Number (RSN) must be specified). The attribute returns the number of lines on a page as specified in the Network Information File (NIF) description of the station. For example:

```
NUMBEROFLINES :=REMOTEFILEID (RSN).PAGESIZE;
```

PARITY

Tape/Paper tape, read/write, anytime/closed, integer, Tape

The attribute PARITY indicates the parity used on the file.

The values and mnemonics of the PARITY attribute are:

0	STANDARD (Binary or odd parity)
1	NONSTANDARD (Alpha or even parity)

PARITY may be set to NONSTANDARD only for 7-track magnetic tape files (KIND=TAPE7) and paper tape files. Setting PARITY to NONSTANDARD on tape files when KIND equals TAPE9 or TAPEPE will generate a file attribute error when the tape file is opened.

POPULATION

Data Comm/Disk, read only, assigned, integer, not permanent

The attribute POPULATION returns the number of logical files assigned to a disk file or the number of stations assigned to a Data Comm file. To access POPULATION for a Data Comm file, the file must be open (assigned includes closed with retention). If the POPULATION of a Data Comm file is not equal to its STATIONCOUNT, then the attributes STATIONSDENIED and DISPOSITION can be used to discover the file's state. See the discussion of Data Comm files.

PRESENT

General, read only, anytime, Boolean, not permanent

The attribute PRESENT allows a program to open a logical file whenever the permanent file exists or physical unit is available, without causing the operator to respond to NO FILE or FILE REQUIRES conditions. If PRESENT returns TRUE, then the file has been opened. If PRESENT returns FALSE, the file does not exist. If the attribute EXCLUSIVE is TRUE or the object program is run in an environment where disk files are opened exclusively, the program can be suspended until the required disk file is unlocked. In this case, the operator is notified by a WAITING ON message. The AVAILABLE attribute can be used to prevent this condition. To test whether or not a file exists without causing the file to be opened, the attribute RESIDENT should be used.

The PRESENT attribute may also return TRUE when no physical file exists. This situation may occur when KIND is DISK or PACK and a new file is to be created. In other words, if the file would be created upon opening the file, it is considered to exist.

The PRESENT attribute will return FALSE if the family specified by FAMILYNAME is not present, or a file with the appropriate SERIALNO, CYCLE, or VERSION does not exist on the specified family.

The PRESENT attribute returns FALSE if the physical file exists, but would cause an OPEN ERROR (Invalid Translation or Incompatible Blocking, for example) if assigned to the logical file.

PROTECTION

Disk/Tape, read/write, anytime/closed, integer, Disk/Tape

The attribute PROTECTION indicates the amount of extra effort desired to preserve a file in case of a system failure. The values and mnemonics of the PROTECTION attribute are as follows:

0	TEMPORARY
1	SAVE
2	PROTECTED

The default value of PROTECTION is TEMPORARY. This means that a new disk file is thrown away when the program is discontinued or the block in which the file was declared is exited, unless the file is explicitly closed by execution of an overriding close statement (for example, LOCK or CRUNCH). If the disk file is locked or crunched, an entry is made in the directory and the file becomes permanent. If the file is a tape file, the value TEMPORARY means no special action is performed; the file is not discarded when it is closed.

If PROTECTION is set to SAVE, an entry in the directory is made immediately when the disk file is opened. The file becomes a permanent disk file and remains after the program has finished unless the file is explicitly purged.

If PROTECTION is set to PROTECTED, for a disk file, an entry is made immediately in the directory when the file is opened, and as the disk areas are allocated, they are encoded with a pattern which makes it possible to discover the last valid block written on that area in the event of a Halt/Load. An output tape file with PROTECTION equal to PROTECTED (end-of-tape is found uptape after a Halt/Load) will have tape marks written on the tape before it is rewound, thus making the file readable after the system is reinitialized. PROTECTION equal to SAVE is meaningless for tape files.

PROTECTION returns the value for the physical file if read when the file is assigned.

RECEPTIONS

Data Comm, read only, open, integer, not permanent

The attribute RECEPTIONS is the number of messages from the station (if a Relative Station Number (RSN) was specified) or the number of messages from all the stations read from the Data Comm file.

RECORD

General, read only, anytime, integer, not permanent

The attribute RECORD returns the current position in the file. When the file is closed, is open but no I/O has been performed upon the file, or was opened with a READ[NO] operation, RECORD returns minus one (-1). The counting of the RECORD attribute is zero-relative, that is, the first record in the file is RECORD zero (0).

When reading reverse, the RECORD attribute starts at zero, the same as forward.

RECORDINERROR

Disk/Tape, read only, open, integer, not permanent

The attribute RECORDINERROR returns the record number (zero relative) of the first record in the block which is contained in the top buffer. Refer to the discussion of the IOINERROR attribute.

RECOVERYFILEID

Reader Sorter, read/write, anytime/anytime, pointer, not permanent

This attribute specifies the TITLE of a disk file to be used by the RSP for recovery purposes. The file must be an existing file.

RECOVERYINDEX

Reader Sorter, read/write, anytime/anytime, integer, not permanent

This attribute specifies the record position in the recovery file where the RESTART procedure of the UCR begins reading records. The record position is taken modulo 10,000,000 and is the zero relative record number.

REEL

REEL is a nonpreferred synonym for FILESECTION.

TEXT DELETED

REMOTEID

Reader Sorter, read only, assigned, pointer, not permanent

This attribute is used to obtain the information contained in the RSNET file as the "remote station" string. This may be a name of a remote station or any encoded string that is meaningful to the program.

RESIDENT

General, read only, anytime, Boolean, not permanent

The attribute RESIDENT allows a program to test whether or not a permanent file exists or a physical unit is available without causing the logical file to be opened or the program to be suspended. If the logical file is already open or the file exists, RESIDENT returns TRUE. The attribute RESIDENT is unaffected by the setting of the EXCLUSIVE attribute. Whether or not a disk file is locked, RESIDENT returns TRUE if the file exists. Whether or not a disk file exists, RESIDENT returns TRUE if opening the file will cause creation of a new file.

If RESIDENT returns TRUE and DEPENDENTSPECS equals TRUE (or FILETYPE equals 7 or 8), then the other attributes which describe the permanent file, such as MINRECSIZE, MAXRECSIZE, BLOCKSIZE, SIZEMODE, SIZE2, SIZEOFFSET, UNITS, and INTMODE (FILETYPE=7 only), are changed to reflect the values of the permanent file. The FILETYPE attribute is not changed in any case, so the dependent specification action still takes place when the file is ultimately opened. For additional information, refer to the AVAILABLE and PRESENT attributes in this section.

RESIDENT will return false if the family specified by FAMILYNAME is not present or a file with the appropriate SERIALNO, CYCLE, or VERSION does not exist on the specified family.

ROWADDRESS

Disk, read only, assigned, integer, Disk

The attribute ROWADDRESS returns the physical disk address of an area of a disk file. The ROWADDRESS attribute requires an index, the row number, as a parameter. If the disk file is DUPLICATED, the copy number is also required as a parameter.

Example

```
ADDRESS := DSK(ROWNBR).ROWADDRESS;

ADDRESS := DUPLDSK(ROWNBR,COPYNBR).ROWADDRESS;
```

Row numbers begin at zero (0); copy numbers begin at one (1).

The value returned by the ROWADDRESS contains the FAMILYINDEX in the field 29:8 and the segment address in 21:22. A zero is returned if the row is not assigned.

ROWSINUSE

Disk, read only, assigned, integer, Disk

The attribute ROWSINUSE returns the actual number of areas allocated for a disk file. Since disk areas are allocated only when needed, it is possible using random I/O to create a disk file where areas in the logical middle of the file are never referenced and hence never allocated.

RSERROR

Reader Sorter, read only, assigned, integer, not permanent

When a flow-stop condition or UCR abort occurs, this attribute returns an integer value which indicates the error condition. The values and meanings are:

1	NOTREADY (Unit not ready)
2	BLACKBAND (Black-band item detected)
3	JAM (A jam occurred)
4	MISSORT (A missort was detected)
5	LATEREAD (Too-late-to-read condition)
6	LATESELECT (too-late-to-select condition)
7	INITCOMPLETE (UCR was initiated)
8	UCRSTOPFLOW (UCR requested stop flow)
9	EXTENDRECOVERY (More disk space is needed)
10	FLOWSTOPCOMPLETE (Flow-stopped procedure finished)
11	REMAPREQUEST (Image file remap is requested)
31	MULTIFLOWSTOP (Multiple conditions occurred see RSRESULT)
33	UCRABORT (The UCR aborted)
35	RSPERROR (Abort due to RSP error)

Integers 1 through 31 apply to flow stopped and 33 through 35 apply to UCR abort.

RSEXTENDIBLE

Reader Sorter, read/write, anytime/closed, Boolean, not permanent

This attribute is set TRUE to indicate that the recovery file is to be written in a linear (non-wrap-around) fashion. If RSEXTENDIBLE is FALSE, the recovery file will be written in wrap-around mode. Wrap-around occurs when the end of the file is reached and simply means that writing will continue starting at the beginning of the file.

RSRESTART

Reader Sorter, read/write, anytime/closed, Boolean, not permanent

If RSRESTART is TRUE at file open time, the UCR RESTART procedure will be executed when the UCR is loaded. If RSRESTART is FALSE at file open time, the UCR INITIATION procedure will be executed when the UCR is loaded.

RSRESULT

Reader Sorter, read only, assigned, real (word); not permanent

This attribute returns a result descriptor after a flow-stopped condition. One bit will be set in the result to indicate a specific condition. The conditions may be programmatically masked out (see FCM attribute) and therefore never seen. Setting a bit to 1 indicates the occurrence of the condition. The bits and conditions are:

Bit 0 Flow-stop occurred and one or more of the following bits is on.

Bit 1 The readersorter went not ready.

Bit 2 A Black-band item was detected.

Bit 3 A jam occurred.

Bit 4 A missort was detected.

Bit 5 Too-late-to-read occurred.

Bit 6 Too-late-to-pocket-select occurred.

Bit 7 A UCR INITIATION or RESTART procedure finished.

Bit 8 The UCR requested a stop flow.

Bit 9 More disk space is needed for the recovery file.

Bit 10 A UCR FLOW=STOPPED procedure finished.

Bit 11 Image file remap was requested.

RSUNITNO

Reader Sorter, read/write, anytime/closed, integer, not permanent

This attribute returns the logical readersorter unit number (as specified in the RSNET file) that is assigned. Setting RSUNITNO may cause TITLE to be overridden but will not cause KIND to be overridden. The UNITNO attribute will override RSUNITNO.

SAVEFACTOR

Tape/Disk, read/write, anytime/closed, integer, Disk/Tape

The attribute SAVEFACTOR indicates the expiration date of a file in terms of the number of days past the creation date. A tape file mounted with a write ring is marked as scratch by the system if the expiration date has passed. This is also the case when a tape file is created without setting SAVEFACTOR since the default value is zero. The system does not purge a tape whose SAVEFACTOR has expired unless explicitly told to do so; removing the write ring allows the tape to be read.

The SAVEFACTOR has no system defined meaning for a disk file. However, when the disk file is the output file of a Sort, the Sort locks the file if SAVEFACTOR is nonzero and otherwise closes the file with release.

SAVEFACTOR returns the value for the physical file if read when the file is assigned.

SCREEN

Data Comm, read only, open, Boolean, not permanent

The attribute SCREEN returns TRUE if the station (indicated by the specified Relative Station Number (RSN)) is declared in the Network Information File (NIF) to be a screen device.

SCREENSIZE

Data Comm, read only, open, integer, not permanent

The SCREENSIZE attribute returns the number of lines per page as defined in the Network Information File (NIF) description of the specified station. The SCREENSIZE attribute requires a Relative Station Number (RSN) as a parameter.

Example

```
NUMBEROFLINES := REMOTEFILEID (RSN).SCREENSIZE;
```

SECURITYGUARD

Disk, read/write, anytime/anytime, pointer, Disk

The attribute SECURITYGUARD identifies the guard file if GUARDED or CONTROLLED security is to be invoked for the file. Refer to the discussion of the SYSTEM/GUARDFILE utility program in the B 7000/B 6000 System Software Operational Guide, Volume 1.

The SECURITYGUARD attribute returns the TITLE of the GUARDFILE for the physical file, if requested by the owner (as identified by the usercode) of the physical file. If the logical file has not been assigned to a physical file and a GUARDFILE has been specified (the SECURITYGUARD attribute specified) for the logical file, then the name of the specified GUARDFILE is returned. In all other cases a null name (".") is returned.

The format of the SECURITYGUARD attribute is that of an external file name. (Refer to the TITLE attribute.) The SECURITYGUARD information is a property of the physical or permanent disk file and is associated with the physical file when it is created. If the SECURITYGUARD attribute is changed while the file is opened, the guard file of the physical file is changed. This may only be done by the owner of the physical file or a privileged user. When a permanent disk file protected by GUARDED security is opened by a user other than the owner of the file (or a file protected by CONTROLLED security is opened by anyone), access to the file is determined by the contents of the guard file. If SECURITYGUARD is not specified for the permanent disk file or if the specified guard file is not present, then PRIVATE security instead of GUARDED is invoked.

SECURITYTYPE

Disk/Port, read/write, anytime/anytime, integer, Disk

The attribute SECURITYTYPE specifies who (as identified by the USERCODE) may access a file. The values and mnemonics of the SECURITYTYPE attributes are as follows. CLASSA and CLASSB are nonpreferred synonyms for PUBLIC and GUARDED, respectively.

0	PRIVATE
1	PUBLIC or CLASSA
2	GUARDED or CLASSB
3	CONTROLLED

PRIVATE: Implies that only a privileged user or the owner may access the file.

PUBLIC: Allows access to any user who references the file using the (USERCODE) TITLE form of the file name.

GUARDED: Security requires permission from the guard file (identified by the SECURITYGUARD attribute) before a nonprivileged user, other than the owner, may access the file.

CONTROLLED: This value of SECURITYTYPE is identical to GUARDED with the added restriction that even the owner (if nonprivileged) requires permission from the guard file before he may access the file.

The default value of SECURITYTYPE is PRIVATE if the file is created under a USERCODE (i.e., all CANDE created files); it is PUBLIC otherwise.

The only valid values of SECURITYTYPE for port files are PRIVATE and PUBLIC.

SECURITYTYPE returns the value for the physical file if read when the file is assigned.

SECURITYUSE

Disk, read/write, anytime/anytime, integer, Disk

The attribute SECURITYUSE specifies the manner in which a disk file that is protected by security may be accessed. The default value for SECURITYUSE is IO. SECURITYUSE is ignored if a guard file is invoked. The values and mnemonics of the SECURITYUSE attribute are:

0	SECURED
1	IN
2	OUT
3	IO

SECURED implies that no one may access the file, with the exception that if it is a code file, it may be executed.

IN, OUT, and IO indicate that read-only, write-only, and read/write access (respectively) are allowed to the file.

SECURITYUSE returns the value for the physical file if read when the file is assigned.

SENSITIVEDATA

Disk, read/write, anytime/closed, Boolean, Disk

The SENSITIVEDATA attribute causes the disk or pack areas assigned to the file to be overwritten with an arbitrary pattern before the disk space is returned to the system for reallocation. SENSITIVEDATA files are entered into the directory, as if PROTECTION were equal to SAVE, when they are created.

The default for SENSITIVEDATA is FALSE.

SERIALNO

Tape/Disk, read/write, anytime/anytime, real (string), Disk/Tape

The attribute SERIALNO returns the serial number of the labeled tape or base member of the disk family to which the logical file is assigned. The serial number of a tape is established by the SN system input message. The RC, IV, or LB system input messages can be used to establish a serial number for a disk peripheral.

A serial number is one word of data consisting of an alphanumeric string of up to six EBCDIC characters left justified in a field of blanks.

The SERIALNO attribute can be set only when the file is closed and unassigned. When setting this attribute, an optional index may be used to indicate an individual element of a SERIALNO list. A SERIALNO list may contain one or more elements. A list of more than one element can have unassigned elements indicated by adjacent commas, scattered throughout.

SERIALNO can be read at any time. When the logical file is open or closed with retention and has a labeled tape or a disk file assigned to it, the serial number of the physical file or base unit (respectively) is returned. When a physical file is not assigned to the logical file, an optional index may be used to read the value of an individual element of a SERIALNO list. If the logical file is assigned to a nondisk, nonlabeled tape physical file, is positioned beyond the last file of a multifile tape, or is unassigned, and the SERIALNO attribute has not been set for the particular element of the list, an attribute error is given.

When assigned a value, the SERIALNO attribute is used in file assignment. When searching for a permanent disk file or selecting a mass storage family on which to create a disk file, the first element in a SERIALNO list, if assigned a value, is used to identify the base unit of the family.

Setting SERIALNO for a particular volume indicates that serial number checking is requested when a tape file is to be assigned. For permanent tape files, the serial number of the physical tape must match the SERIALNO attributes value along with the KIND, TITLE, and CYCLE and VERSION attributes when genealogy checking is desired. (See the File Assignment section for further discussion.) When creating a tape file, if SERIALNO is specified, file assignment will succeed only if the serial numbers match. The tape must have a write ring and cannot be locked, saved, assigned, uptape or not ready. If the tape is not scratched, the tape is rewound so that new labels will be written at the beginning of the volume (in effect purging the tape). Setting SERIALNO to all null characters (all bits zero) indicates that the attribute is not to be considered during file assignment.

An optional reel number (in ALGOL, F(REELNBR).SERIALNO) may be specified for multireel files. The attribute may be set while the logical file is assigned as long as the element in the SERIALNO list as indicated by the reel number (one relative) is not the reel currently in use.

If a permanent tape file is found that meets all the requirements except for the serial number, an UNMATCHED SERIALNO notification is given to the operator. The operator can respond by making the file available, or by the IL, OF, FA, or DS system input messages.

On output, the FILE REQUIRES message is displayed with the desired serial number in brackets. Operator response may be DS, OU, or SN.

The RESIDENT and PRESENT attributes return false on UNMATCHED SERIALNO conditions. The AVAILABLE attribute returns a "6" on UNMATCHED SERIALNO, and a "4" on an UNMATCHED GENEALOGY.

The MCP run-time option SERIALNUMBER (option 24) inhibits the assignment of scratch tapes unless the serial numbers match. The tape is rewound and purged if it has a write ring and a matching SERIALNO.

The SERIALNO attribute may be assigned a REAL value via an ALGOL statement or file declaration; the value may be the following:

a. An integer value from 1 through 999999.

b. One to six EBCDIC alphanumeric characters, left-justified with blank fills.

c. Normal zero (the null value for SERIALNO).

The following ALGOL and WFL text specify the same SERIALNO values:

ALGOL	WFL	SERIALNO
F.SERIALNO:=123	F(SERIALNO=123)	[000123]
F.SERIALNO:="AB "	F(SERIALNO="AB")	[AB]
F.SERIALNO:="AB"	F(SERIALNO=49602)	[049602]

The third line illustrates an easy mistake, since ALGOL short strings are right-justified with NUL fill and thus have integer values.

TEXT DELETED

SINGLEPACK

SINGLEPACK is a nonpreferred synonym for SINGLEUNIT.

TEXT DELETED

SINGLEUNIT

Disk, read/write, anytime/closed, Boolean, Disk

The attribute SINGLEUNIT indicates whether or not (TRUE or FALSE) areas for the disk file are to be allocated from a single family member. The default value for SINGLEUNIT is FALSE, which provides for the distribution of areas over the entire family. If additional members are added to the family subsequent to the opening of the logical file, they will be inserted into the rotational scheme of allocation.

When space is unavailable on the original unit, the System Message "REQUIRES FAMILYINDEX(N) SEGMENTS/SECTORS" is displayed. Operator response is OK to override the familyindex requirement allowing the file to overflow to another unit in the family or DS of the program.

SIZEMODE

General, read/write, anytime/closed, integer, Disk/Tape

The attribute SIZEMODE indicates the mode of the record length information contained in the field defined by the SIZEOFFSET and SIZE2 attributes. The SIZEMODE attribute is used with files of FILETYPE equal to four (4).

The values, mnemonics and meanings of the SIZEMODE attributes are as follows:

0	SINGLE	(Word mode, Binary, 48-bit)
2	HEX	(4-bit, packed decimal)
3	BCL	(6-bit)
4	EBCDIC	(8-bit)
5	ASCII	(8-bit)

The default value is SINGLE.

SIZEOFFSET

General, read/write, anytime/closed, integer, Disk/Tape

The attribute SIZEOFFSET points to the field in the variable length record where the record length information is contained. The SIZEOFFSET attribute, along with the SIZEMODE and SIZE2 attributes, is used with files of FILETYPE equal to four (4). The default value is zero (0). The maximum value of the SIZEOFFSET attribute is 65,535. The value of the SIZEOFFSET attribute is in terms of SIZEMODE units.

SIZE2

General, read/write, anytime/closed, integer, Disk/Tape

The attribute SIZE2 indicates the length of the field in the variable length record pointed to by the attribute SIZEOFFSET. The SIZE2 attribute is used with files of FILETYPE equal to four (4). The default is zero (0). The maximum value of SIZE2 is 65,535 in SIZEMODE units. If SIZEMODE is SINGLE (words), then a value of one (1) is used for SIZE2.

SORTFILEID

Reader Sorter, read/write, anytime/anytime, pointer, not permanent

This attribute specifies the TITLE of the sort tables to be used by the UCR.

SORTTABLECODE

Reader Sorter, read/write, anytime/anytime, integer, not permanent

This attribute specifies which table, within the sort table file, is to be used. The encoded information selects a set of tables according to the number of pockets on the readersorter and a specific table within the set. The default value (zero) selects the first matching readersorter set size and the first table in that set.

SPEED

Head-per-track disk, read/write, anytime/closed, integer, Head-per-track disk

The attribute SPEED defines the acceptable range of data transfer speeds, when selecting head-per-track disk devices from which to allocate the areas of the file. The values and mnemonics of the SPEED attribute are as follows:

0	FAST (the default)
1	MEDIUMFAST
2	MEDIUMSLOW
3	SLOW

SPEED returns the value for the physical file if read when the file is assigned.

NOTE

The SPEED attribute is ignored by B 6900 and later systems and will be deimplemented on all systems as of the 3.4 release. Currently, the following warning message is reported if a file is opened with SPEED explicitly set for the logical file:

ON 34, THE "SPEED" ATTRIBUTE WILL BE DEIMPLEMENTED

STATE

General, read only, assigned, real (word), not permanent

The attribute STATE returns a copy of the logical I/O result descriptor that corresponds to the last I/O operation performed on the logical file.

This attribute is not valid for Direct I/O files. The fields of this word contain the following information regarding the last I/O operation performed on the file:

Field	Contents
[0:1]	If this bit is on, an error has occurred and the following error field is nonzero.
[16:16]	Error information is as follows:

Field	Contents
[1:1]	A flow stopped condition occurred for a READERSORTER unit. This is the attention bit for duplicated files (see Duplicated Files in Section 4).
[2:1]	READERSORTER only. A UCR abort occurred.
[3:1]	PORT only. If this bit is on, an invalid subfile index was specified.
[4:1]	If this bit is on, a data error has occurred (size error for variable length records).
[5:1]	If this bit is on, a deleted or duplicated record was referenced in the file whose FILEORGANIZATION is RELATIVE.
[7:1]	If this bit is on, a parity error has occurred.
[8:1]	If the file is a port file and this bit is on, an I/O operation failed for one of the following reasons: a. A broadcast write failed for one or more subfiles. b. A write with the DONTWAIT option failed because no buffer was available. c. A read with the DONTWAIT option failed because no data was available. If the file is a disk file and this bit is on, a deleted disk area was referenced.
[9:1]	If this bit is on, an end-of-file (EOF) or end-of-page was encountered.
[10:1]	If this bit is on, a short block has been read. This is reported without the attention field [0:1] being turned on.
[13:1]	If this bit is on, the Data Comm file has encountered a break on output.
[13:2]	Used for hardware channel addressing. This is reported without the attention field [0:1] being turned on. (Does not occur for Data Comm I/O.)
[15:1]	If this bit is on, the I/O TIMELIMIT has been exceeded.
[16:1]	If this bit is on, a security violation has been attempted.

Field	Contents
[24:8]	Qualification information for Data Comm files, indicating the reason for EOF: Value Meaning 1 File assignment denied. 2 File assignment postponed. 3 Illegal file use. Other values are used for READERSORTER files, see RSERROR attribute.
[27:16]	These bits indicate the duplicated disk file error mark.
[47:20]	Data size of the last logical record read or written in INTMODE units or words as specified by the FRAMESIZE or UNITS attribute. BLOCKSTRUCTURE must equal FIXED (FILETYPE equal to 3) for this field to reflect the actual size of the record instead of the declared record size (MAXRECSIZE).

STATIONCOUNT

Data Comm, read only, open, integer, not permanent

The attribute STATIONCOUNT returns the number of stations in the Data Comm file's family. Unless the STATIONLIST attribute has been used to add or subtract stations from the file, STATIONCOUNT is equal to the number of stations specified by the Data Comm file's description in the Network Information File (NIF). If the STATIONLIST attribute has been used to subtract stations from the file the largest valid Relative Station Number (RSN) may be larger than the STATIONCOUNT.

STATIONLIST

Data Comm, write only, open, pointer, not permanent

The attribute STATIONLIST may be used to add (or subtract) a station or set of stations (the family of a file described in the Network Information File (NIF)) to (or from) a Data Comm file. The format used with the STATIONLIST attribute is that of an external file name. Refer to the discussions of the STATIONNAME and TITLE attributes in this section.

The syntax for the use of the STATIONLIST attribute is as follows:

```
ALGOL:   REPLACE FILEID.STATIONLIST BY * + "DC1.";
         REPLACE FILEID.STATIONLIST BY * - "L1011.";

COBOL:   SET FILEID (STATIONLIST) UP BY "TFX.".
         SET FILEID (STATIONLIST) DOWN BY "B747.".
```

The STATIONLIST attribute makes it possible to dynamically alter the stations associated with an open Data Comm file. However, if the file or station to be added or subtracted is not described in NIF, or is already in the family (in the case of addition), or is not in the family (in the case of subtraction), then no change is made to the file.

Attempting to add/subtract a station/file not found in NIF causes an attribute error. Subtracting all stations from the file does not close the file; however, subsequent I/O statements while STATIONCOUNT is zero will cause end-of-file action.

When a station is subtracted from a file, the STATIONCOUNT and all other relevant attributes are modified, the controlling MCS of the station is sent a FILE CLOSE message for the station, and all knowledge of that station from the logical file's point of view is lost. When a station is added to a Data Comm file's family, the STATIONCOUNT is increased, and the controlling MCS is sent a FILE OPEN message for the station. To reference the individual stations of a Data Comm file, refer to the LASTSUBFILE attribute.

STATIONNAME

Data Comm, read only, open, pointer, not permanent

The attribute STATIONNAME returns the name of the specified station as defined in the Network Information File (NIF). STATIONNAME requires a Relative Station Number (RSN) as a parameter. If the RSN given is invalid, a null string (".") is returned.

STATIONSDENIED

Data Comm, read only, open, integer, not permanent

The attribute STATIONSDENIED returns the number of the stations in a family which have been denied assignment to the file by their controlling MCS(s).

Because Data Comm file assignment is an asynchronous process requiring the cooperation of Message Control Systems (MCS), there is a period when the POPULATION attribute can return a value of zero (0) because the MCS has not yet reacted to the file open request(s). With the use of the STATIONSDENIED and STATIONCOUNT attributes, the object program can distinguish this interim period from the case where POPULATION is zero because all stations have been denied assignment.

SUBFILEERROR

Subfile, read only, anytime, integer, not permanent

The SUBFILEERROR attribute returns an integer value corresponding to the errors, if any, which occurred on the last I/O, OPEN, or CLOSE operation that affected the specified subfile. The SUBFILEERROR attribute requires a subfile index as a parameter if MAXSUBFILES is greater than 1.

The following table gives the mnemonic and meaning for each value of the SUBFILEERROR attribute:

Mnemonic	Integer Value	Meaning
NOERROR	0	No error occurred.
DISCONNECTED	1	Communication with the complementary subfile was lost.
DATALOST	2	The subfile may have closed before transmission of data was complete.
NOBUFFER	3	A write with the DONTWAIT option failed because no buffer space was available.
NOFILEFOUND	4	An open operation failed because a matching subfile was not found.
UNREACHABLE	5	The remote host became unreachable during the open operation.

TANKING

Data Comm, read/write, anytime/closed, integer, not permanent

This attribute is associated with Data Comm output only. It may be used to specify or deny output tanking or leave the decision to the MCS. The attribute values and mnemonics are:

0	UNSPECIFIED
1	NONE
2	SYNC
3	ASYNC

UNSPECIFIED is the default and means that tanking will not occur unless overridden by the MCS at file assignment time.

NONE will cause no tanking to occur and will prevent the MCS from specifying tanking.

SYNC will cause the file to be tanked. In addition, the task will not be able to continue past file close, for the file specifying SYNC, until all tanked output has been transmitted. The associated MCS can not override SYNC.

ASYNC will cause the file to be tanked. The task is allowed to proceed beyond file close, for the file specifying ASYNC, and even go to end-of-task. The Data Comm output will continue to be detanked and transmitted until exhausted.

Causing break-on-output for a file having tanked output will result in discarding the currently tanked output for the file.

TAPEREELRECORD

Tape, read only, assigned, integer, Tape

The attribute TAPEREELRECORD returns the record number of the last record read or written, relative to the first record on the current reel of the tape file.

For example, after reading the first record of the second reel of a tape file, TAPEREELRECORD will return zero (0). TAPEREELRECORD is only valid for labeled tape files. The value returned before the first I/O is minus one (-1).

NOTE

The CLOSE REEL construct on an input tape invalidates the value returned by TAPEREELRECORD.

TIMELIMIT

Data Comm, read/write, open/anytime, real, not permanent

The attribute TIMELIMIT is a positive real number in units of seconds. The attribute can be set via label equation, in the file declaration, at run time with the file either opened or closed, or in read or write statements (in ALGOL using the [TIMELIMIT <aexp>] form of the <record number or carriage control> part of the I/O statement). TIMELIMIT may only be read while the file is open.

The default value of TIMELIMIT is 0. When TIMELIMIT equals 0, I/O operations wait indefinitely. The action when TIMELIMIT is greater than zero is as follows:

a. On a read statement, if no input is received within TIMELIMIT seconds, the read statement is terminated with a TIMELIMIT error.

b. On a write statement, if the data cannot be buffered for output within TIMELIMIT seconds, the write is terminated with a TIMELIMIT error.

A TIMELIMIT error is reported by the logical I/O result descriptor having the attention bit [0:1] and bit [15:1] turned on. Refer to the STATE attribute in this section.

TEXT DELETED

TITLE

General, read/write, anytime/anytime, pointer, Disk/Tape

The attribute TITLE is an external file name. It is used to associate a logical file with a physical file or match complementary port files. The default TITLE for the file is the value of the INTNAME attribute.

If the file is a disk file, the TITLE attribute may also be set using a string of the form <file identifier> ON <familyname>. When TITLE is set to a value of this form, the FILENAME, FAMILYNAME, and KIND attributes are affected as follows:

a. The FILENAME attribute is set to <file identifier>.

b. The FAMILYNAME attribute is set to <familyname>.

c. The KIND attribute is set to PACK.

When the TITLE of a disk file is set to a value of the form <file identifier> (without the ON <familyname> phrase), only the FILENAME is changed; the FAMILYNAME and KIND attributes are not affected.

The TITLE attribute will return the name of the physical file whenever one is assigned to the logical file. The TITLE returned for an unassigned file or a tape file that is positioned beyond the last file on the tape is a null identifier (".") . When a disk file is assigned to the logical file the TITLE attribute may return a string containing an asterisk (*) or usercode and an ON <familyname> phrase.

If the file KIND=PACK, the default familyname is "PACK". If KIND=DISK, the default familyname is "DISK".

NOTE

As of the 3.3 release, the values DISK and PACK of the KIND attribute will be synonymous, and the default value of FAMILYNAME will be "DISK" for both disk and disk pack files (KIND equals either DISK or PACK for the logical file). Refer to the discussion of FAMILYNAME under File-Naming Conventions in Section 4 of this manual.

The TITLE attribute may be changed when the logical file is open only when it is assigned to a disk file. Under these conditions, changing the TITLE of the file causes the file name of the physical disk file to be changed also.

For duplicated disk files whose TITLE is AAA, the file names of the individual files will be: AAA/"COPY#01", AAA/"COPY#02" and so on.

If the file is a port file, the TITLE must be a simple name from 1 to 17 characters in length.

ALGOL requires that the name be terminated by a period and enclosed in quotes. Work Flow Language disallows quotes around the whole name and disallows the terminating period. The COBOL VALUE OF ID clause requires quotes and disallows the terminating period. However, setting the TITLE attribute in a set statement in COBOL does require the terminating period. The FORTRAN file card permits neither the quotes nor the period.

When the logical file is assigned to a Data Comm file, the Relative Station Number (RSN) may be specified and the TITLE attribute will return the name of that station as it is defined in the Network Information File (NIF). If the RSN is not valid, a null name (".") is returned. If the RSN is zero, the TITLE of the file as declared in the program or label equated is returned.

TRAINID

Printer, read/write, anytime/closed, integer, not permanent

This attribute is used to specify a particular print train for a train printer. When this attribute is set to a value other than NOID, the translate table will be loaded, as needed, from the file SYSTEM/TRAINTABLES and the operating system will wait until the selected print train is made available.

The standard values and mnemonics of TRAINID are as follows. NOID is the default.

0	NOID (Any available train or drum printer)
1	EBCDIC18
4	EBCDIC48
5	EBCDIC72
12	BCL64
15	ASCII72
16	EBCDIC96

Files that are printed on a train printer with no TRAINID specified (with value NOID) are printed just as they would be printed on a drum printer. The character set used will be the BCL64 set or similar set of 64 characters. Characters that would print as question marks on the drum printer will also be printed as question marks on the train printer even though the print train on the train printer may contain more than 64 characters. TRAINID must be specified to print more than the default 64 characters.

TRANSLATE

General, read/write, anytime/closed, integer, not permanent

The attribute TRANSLATE indicates the scope of translation. The values, mnemonics, and meanings of the TRANSLATE attribute are as follows:

0 DEFAULTTRANS: On the B 7800 and earlier systems, DEFAULTTRANS is the default value of TRANSLATE when the run-time system option FULLTRANSLATION is reset. (Refer to the B 5000/B 6000/B 7000 Series Operator Display Terminal Reference Manual for specific information regarding the FULLTRANSLATION run-time system option.)

Software translation will only occur when translation is required and hardware translation is not provided. (Refer to Table 4-2 for specific information regarding the translation available to a file with TRANSLATE equal to DEFAULTTRANS.)

1 FULLTRANS: On the B 7800 and earlier systems, FULLTRANS is the default value of TRANSLATE when the run-time system option FULLTRANSLATION is set. On the B 6900 and later systems, FULLTRANS is the default value of TRANSLATE regardless of the value of FULLTRANSLATION.

Software translation will only occur when translation is required and hardware translation is not provided. (Refer to Table 4-2 for specific information regarding the translation available to a file with TRANSLATE equal to FULLTRANS.)

2 SOFTONLY: Software translation will take place whenever translation is required.

3 FORCESOFT: Software translation will take place using tables that may be provided by the program whether or not translation is required as long as software translation is not specifically disallowed (neither INTMODE or EXTMODE is equal to SINGLE or BINARY, FILETYPE is not equal to 4 or 6, and it is not a Direct I/O file). Refer to the INPUTTABLE and OUTPUTTABLE attributes.

4 NOSOFT: An invalid translation open error will occur whenever translation is required and hardware translation is not provided.

5 NOTRANS: An invalid translation open error will occur whenever translation is required.

TRANSLATING

General, read only, assigned, Boolean, not permanent

The attribute TRANSLATING returns TRUE if software translation is being performed on the records of the file. The attribute is set or reset when the file assignment is complete, taking into consideration the values of the INTMODE, EXTMODE, BLOCKSTRUCTURE (or FILETYPE), and TRANSLATE attributes.

TRANSMISSIONNO

Data Comm, read only, open, integer, not permanent

The attribute TRANSMISSIONNO returns the transmission number of the input last received by the object program from the station indicated by the specified Relative Station Number (RSN). If the description of the station in the Network Information File (NIF) does not require the DCP to assign transmission numbers to the station's messages, the value returned is minus one (-1).

TRANSMISSIONO

TRANSMISSIONO is a nonpreferred synonym for TRANSMISSIONNO.

TEXT DELETED

TRANSMISSIONS

Data Comm, read only, open, integer, not permanent

The attribute TRANSMISSIONS returns the number of output messages sent to the station (if a Relative Station Number (RSN) is specified) or the number of output messages sent to all the stations in the file.

UCRID

Reader Sorter, read/write, anytime/anytime, pointer, not permanent

This attribute specifies the TITLE of the code file for the UCR that is to be invoked.

UNITNO

General, read/write, anytime/unassigned, integer, not permanent

The attribute UNITNO accepts or returns the hardware unit number to which the file is assigned, or is to be assigned. The attribute can be set only before a physical file is assigned to the logical file (closed without retention). If the UNITNO attribute has been set to a value which is associated with a physical disk unit, or to which no physical unit is assigned, or the KIND attribute of the file is not the same type as the physical device, the program is terminated.

UNITS

General, read/write, anytime/closed, integer, Disk/Tape

The attribute UNITS indicates whether or not the transfer of data in the file will be word or character oriented and whether or not the attributes MINRECSIZE, MAXRECSIZE, BLOCKSIZE, CURRENTBLOCK, AREALENGTH, and CURRENTRECORD are expressed in words or characters. All Direct I/O data transfers are basically word oriented. The values and mnemonics of the UNITS attribute are as follows:

0	WORDS (48-bit units)
1	CHARACTERS (INTMODE-defined units)

If UNITS equals CHARACTERS, then unless INTMODE equals SINGLE, the file will be character oriented. The default value is WORDS.

The attribute FRAMESIZE overrides and may change the value of UNITS. Refer to the discussion of the FRAMESIZE attribute in this section.

UPDATEFILE

Disk, read/write, anytime/closed, Boolean, not permanent

The UPDATEFILE attribute allows the user to explicitly indicate when a disk file is to have the Update I/O accessing method.

The default value of UPDATEFILE is FALSE. Prior to the 3.4 release, if the file is a disk file and MYUSE equals IO, UPDATEFILE is set to TRUE when the file is assigned. If the UPDATEFILE attribute is explicitly set FALSE, Update I/O action will not occur regardless of the KIND and MYUSE attributes.

NOTE

As of the 3.4 release, explicitly setting UPDATEFILE to TRUE will be the only way to specify update I/O action. Refer to the MYUSE attribute in this section.

When a disk file is open and UPDATEFILE is TRUE, a serial (nonrandom) write following a read rewrites the record just read, including any changes made to the record. When a disk file is open and UPDATEFILE is FALSE, a serial write following a read writes to the next record in the file.

Testing the attribute when the file is closed will indicate whether or not the attribute has been set.

When the file is opened, UPDATEFILE indicates whether or not update I/O action will be performed upon the file. Regardless of the setting of the attribute, if the logical file is assigned to a nondisk peripheral unit, the value of the attribute will be FALSE. The UPDATEFILE attribute will return TRUE, if the attribute was not specifically set and the disk file has the MYUSE attribute set to IO.

Update I/O is performed by the MCP logical I/O routines (FIBSTACK) with the assistance of the FIBSTACK update disk release routines. In order to retain the efficiency gained via update I/O, the default number of buffers for a file with UPDATEFILE=TRUE is three. Having less than three buffers results in FIBSTACK waiting for look-ahead reads to complete.

USECATALOG

Disk/Tape, read/write, anytime/closed, Boolean, not permanent

When the USECATALOG attribute is TRUE, disk or tape permanent file searches use the information in the System Catalog. When creating a permanent tape or disk file, the file is entered into the System Catalog. The default value of USECATALOG is the value of the run-time system option USECATDEFAULT if cataloging is enabled, otherwise the default value is FALSE. (Refer to the B 5000/B 6000/B 7000 Series Operator Display Terminal Reference Manual.)

USEDATE

Disk, read only, assigned, integer, Disk

The attribute USEDATE returns the date when the file was last read or written by a user program, or for a code file, when it was last executed. The value returned is in the Julian form YYDDD. The USEDATE of a file is unaffected by Library Maintenance. USEDATE will not be updated for files on write lock-out protected disk or disk pack.

VERSION

Disk/Tape, read/write, anytime/anytime, integer, Disk/Tape

The attribute VERSION, in conjunction with the CYCLE attribute, is used to distinguish successive iterations of the same generation of a permanent file. The initial, and default, value of VERSION is zero (0). The maximum value for VERSION is 99. The VERSION attribute may be set while the file is closed or is open but not yet in the directory (for example, a new disk file with PROTECTION equal to TEMPORARY and SENSITIVEDATA equal to FALSE). See the CYCLE attribute.

VERSION returns the value for the physical file if read when the file is assigned.

WIDTH

Data Comm, read only, open, integer, not permanent

The attribute WIDTH returns the logical line length, in characters, for the station (indicated by the specified Relative Station Number (RSN)) as specified by the station's description in the Network Information File (NIF).

Example

```
I := DCOM(RSN).WIDTH;
```

YOURNAME

Subfile, read/write, anytime/closed, pointer, not permanent

The YOURNAME attribute is used by the subfile matching algorithm to match complementary subfiles. The value of YOURNAME must equal the value of MYNAME for the port file to which the complementary subfile belongs. The default value of YOURNAME is a null string ("."). Refer to Port Subfile Assignment in Section 4 of this manual.

YOURUSERCODE

Subfile, read/write, anytime/closed, pointer, not permanent

The YOURUSERCODE attribute is used by the subfile matching algorithm to match complementary subfiles. The YOURUSERCODE attribute must equal the usercode under which the program offering the complementary subfile is running. The default value of YOURUSERCODE is the usercode of the task that opens the file. Refer to Port Subfile Assignment in Section 4 of this manual.

DIRECT I/O BUFFER ATTRIBUTES

Direct I/O is an I/O technique that allows the program to get as close to the hardware as possible and still preserve the integrity of the operating system. Instead of having the logical I/O subsystem handle buffering and deblocking, direct arrays are used for all data transfer with direct files.

Direct I/O writes are not allowed:

a. To codefiles except by programs with FILEKIND=COMPILERCODEFILE (see the MC System Input Message).

b. To crunched files if the blocksize does not equal the original blocksize.

c. To a duplicated copy of a file.

The normal data transfer using Direct I/O is word oriented although character oriented files are allowed. It should be noted that the SIZE parameter in a Direct I/O READ or WRITE statement is in the form of whole words and characters included in the last word. The fields are word count = [16:17] and characters = [19:03].

Buffer attributes are provided as aids in controlling the Direct I/O and to describe its results. Buffer attributes are accessed in ALGOL and COBOL. The following is an ALGOL example.

```
ADDR := DIRECTARRAYID.IOADDRESS;
```

Notice that the direct array (used as the buffer) is referenced, not the file identifier.

IOADDRESS

Read only, integer

The attribute IOADDRESS returns the absolute memory address of the first word of the buffer.

IOCANCEL

Read/write, Boolean

The attribute IOCANCEL returns TRUE if the last I/O attempted on this buffer was cancelled. If this is the case, the IORESULT attribute will return TRUE. Setting IOCANCEL to TRUE on a buffer where IOPENDING is TRUE (for a nondisk or Data Comm I/O) will purge the I/O queue and cancel all the outstanding I/O's for that device.

IOCHARACTERS

Read only, integer

The attribute IOCHARACTERS returns the number of characters read into this buffer. The values returned after reading reverse, or after a write are unpredictable.

It should be noted that the value returned from a 7-track tape drive is the number of 6-bit frames required to store the record on tape and not, if the data is EBCDIC or ASCII, the number of 8-bit characters in the record.

IOCOMPLETE

Read only, Boolean

The attribute IOCOMPLETE returns TRUE if the I/O action for this buffer is complete.

IOCW

Read/write, real (word)

The attribute IOCW accepts or returns the hardware I/O control word for the buffer. If this attribute is set, it is used instead of the system supplied IOCW until it is reset. The IOCW attribute can be reset by setting it to zero (0).

IOEOF

Read only, Boolean

The attribute IOEOF returns TRUE if end-of-file is encountered as a result of the last I/O on this buffer.

IOERRORTYPE

Read only, integer

The IOERRORTYPE attribute identifies what error, if any, occurred as a result of the last I/O on this buffer. The values returned and their meanings are as follows:

Value	Meaning
Less than 0	Undefined error occurred (the error field of the hardware result descriptor is returned as a negative number).
0	No error.
1	Not Ready.
2	Parity.
3	Rewinding.
4	Descriptor error.
5	End of tape or beginning of tape.
6	End of file or write lock-out
7	Attempt to exceed disk row or read past EOF.
8	I/O was cancelled.

Value	Meaning
9	Short record.
10	Break on output (Data Comm I/O)
11	Diskpack program or operator error.
12	Diskpack hardware error.
13	Diskpack hardware or data error.
14	Diskpack data error.
15	Disk SU or blank tape timeout.
16	Row write locked out by DMS.

IOMASK

Read/write, real (word)

The attribute IOMASK provides the ability to set or read the I/O Error Mask field of the I/O Control Block. The IOMASK attribute allows the user to specify the error conditions that the system is to overlook (and not initiate recovery). It is assumed the user will handle these errors programmatically. Once the IOMASK attribute is set, it is used instead of the system supplied default until it is reset. The IOMASK attribute can be reset by setting it to zero (0).

IOPENDING

Read only, Boolean

The attribute IOPENDING returns TRUE if an I/O is queued or in process on this buffer.

IORECORDNUM

Read only, integer

The attribute IORECORDNUM returns the logical record number of the last disk I/O performed on this buffer or the value of the LASTSUBFILE attribute for direct Data Comm files.

IORESULT

Read only, Boolean

The attribute IORESULT returns the hardware result descriptor for the last I/O on this buffer. If a TRUE is returned, the I/O was cancelled.

IOTIME

Read only, integer

The attribute IOTIME returns the elapsed time in 2.4 microseconds for the last I/O on this buffer.

IOWORDS

Read only, integer

The attribute IOWORDS returns the number of words read into this buffer. The value returned is inclusive; that is, if the record is N words long plus M characters, IOWORDS returns N + 1. The values returned reading reverse or after a write are unpredictable.

It should be noted that the value returned from a 7-track tape drive is the number of 6-bit frames required to store the record on tape and not, if the data is EBCDIC or ASCII, the number of 8-bit characters in the record.

SPECIAL TOPICS

FILE ASSIGNMENT

Before a program can access or create data contained in a physical file, an association between the physical file and the program must be made. This is done by assigning a physical file to a logical file, the file declared and described in the user's program. The file search and possible file assignment is performed by the open operation on the logical file. The desired physical file is defined by values of attributes of the logical file. The file search and assignment logic of the Input-Output Subsystem attempts to associate the logical and physical files without requiring intervention by the system operator.

An OPEN statement is provided to initiate the open operation in ALGOL (refer to the B 5000/B 6000/B 7000 Series ALGOL Reference Manual for syntax), BDMSALGOL (refer to the B 7000/B 6000 Series DMSII HOST Reference Manual for syntax), DCALGOL (refer to the B 5000/B 6000/B 7000 Series DCALGOL Reference Manual for syntax), and DMALGOL. In most languages (all languages except COBOL and COBOL74), a file may be opened implicitly by the first I/O operation to the file. In the languages where it is provided, the OPEN statement may also be used as a function. The OPEN function returns values synonymous with those returned by the AVAILABLE attribute. Refer to the AVAILABLE attribute in Section 3 of this manual and Port Subfile Assignment in this section.

If a file is opened or closed implicitly (or the value returned by an OPEN or CLOSE function is discarded) and an open or close error occurs, the program is aborted with an error message of the following form:

FILE <title> OPEN/CLOSE ERROR: <error message>

If an error occurs on an I/O operation which discards the I/O result descriptor, the program is aborted with an error message of the following form:

FILE <title> I/O ERROR: <error message>

All OPEN/CLOSE and I/O errors are fatal unless the values returned by the OPEN, CLOSE, or I/O operation is used by the program.

New File vs. Permanent File

The attributes KIND, NEWFILE, MYUSE, and (prior to the 3.5 release) AREASIZE determine whether a new file is created or a permanent file is assigned when the file is opened. Table 4-1 shows the decisions made by the input/output subsystem based on the values of these attributes.

Table 4-1. Attributes Affecting New File/Permanent File Decision

KIND	NEWFILE	MYUSE	AREASIZE (1)	File Found	Open Action
Output only device (PRINTER or PUNCH)	--	--	--	--	New file (2)
Input only device (READER)	--	--	--	yes	Existing file (3)
				no	No file msg (4)
Input/Output device (TAPE)	TRUE	--	--	--	New file
	FALSE	--	--	yes	Existing file
				no	No file msg
	unspec	IN or IO	--	yes	Existing file
				no	No file msg
		OUT	--	--	New file
Input/Output device (DISK)	TRUE	--	--	--	New file
	FALSE	--	--	yes	Existing file
				no	No file msg
	unspec	IN or IO	--	yes	Existing file
				no	No file msg
		OUT	unspec or zero	yes	Existing file
				no	New file
			greater than zero	--	New file

NOTES

1. As of the 3.5 release, the value of AREASIZE will no longer affect the new file/permanent file decision.

2. "New file" means that a new file is created.

3. "Existing file" means that a permanent file is assigned.

4. "No file msg" means that a no file message is reported.

A dash indicates that, given the values of other attributes, this attribute does not affect the new file/permanent file decision.

When the logical file has as its declared KIND an output-only peripheral, for example PRINTER or PUNCH, a new file is to be created. When the KIND is declared to be an input-only peripheral, for example READER or PAPERREADER, the physical file is assumed to be a permanent file and the permanent file assignment logic is invoked. When the KIND attribute for the logical file specifies a peripheral device capable of both input and output (for example, KIND equals TAPE or DISK), the attributes NEWFILE, MYUSE, and (for disk files) AREASIZE determine whether a new file is created or an existing file is assigned.

As of the 3.5 release, AREASIZE will no longer affect file assignment. Prior to the 3.5 release, a run-time warning message is reported whenever a permanent disk file is assigned to a logical file with MYUSE equal to OUT and AREASIZE either unspecified or equal to zero.

The warning given when executing COBOL programs is as follows:

ON 35, OPEN INPUT/OUTPUT MUST BE USED TO WRITE ON PERMANENT FILE.

The warning given when executing other programs is as follows:

ON 35, "NEWFILE=FALSE" MUST BE SPECIFIED TO OVERRIDE "MYUSE=OUT".

Selecting a Peripheral for a New File

When the logical file is used to create a new disk file, the FAMILYNAME attribute identifies a family of mass storage peripherals where space may be allocated for the physical file.

When creating a new nondisk file, the peripheral is assigned on the basis of the availability of a scratch or unassigned peripheral. If the logical file is used to create a new tape file and the SERIALNO attribute is set, a tape with a matching serial number that is not locked, not saved, not uptape, not not-ready, or in use and that has a write ring will be selected. The tape does not have to be scratched. If it has not been, the tape will be rewound so that the new labels will be written at the beginning of the volume (in effect, purging the tape).

When the value of the KIND attribute is PRINTER or PUNCH, both the program and the system have the ability to override the file's immediate destination and to spool the data to an intermediate backup device. The conversion of a printer or punch file into backup file is accomplished by the interaction of system options and file attributes.

If the logical file is used to create new printer or punch file and the FORMID is not NULL, a printer or punch with a matching FORMID that is ready, is not locked, is not saved, and is not in use will be selected.

Finding a Permanent File

When a logical file is to be assigned to a permanent file, a number of attributes (KIND, TITLE, FILESECTION, SERIALNO, CYCLE, VERSION) are used in an attempt to uniquely describe the physical file. The KIND attribute is used to narrow the search to certain peripherals. The TITLE attribute gives the external file name of the permanent file and where appropriate, the family name.

Once the permanent file with the proper TITLE and the correct KIND is found, a more detailed selection process follows. If it is a tape file, the FILESECTION attribute must agree with the file section number of the permanent file. If it is a disk or tape file and genealogy checking is requested, the CYCLE and VERSION attributes are matched with those in the permanent file. If genealogy checking is not requested, the file with the best genealogy (the highest CYCLE value and the highest VERSION of that CYCLE) is selected. Peripherals which do not have settable genealogy ignore genealogy. Frequently, there is only one genealogy of the file available to the system. If the SERIALNO attribute is set for a tape file then the serial number of the physical tape must match the value of the SERIALNO attribute.

Interactions of the Myuse and Security Attributes

Once the existence of a permanent disk file has been determined, the security of the file as defined by the SECURITYTYPE and SECURITYGUARD attributes may prevent file assignment. If the intended use of the file as defined by the MYUSE attribute is less restrictive than the allowed use as defined by the security attributes, the attempt to open the file is considered a security violation. In this case, as far as the logical file is concerned, the permanent file does not exist. If the program would normally be suspended with a NO FILE notification (i.e., if not opened via the AVAILABLE attribute) and the logical file is not optional, the program will be DS'ed instead for a SECURITY VIOLATION.

Restrictions Imposed by Peripheral Association

Under certain conditions imposed by site management, the actual peripheral devices that a program can see will be limited. For example, if a site has a large number of small jobs that require fast turnaround, they might want to designate one particular card reader and one particular printer to handle these jobs. They might install these two devices in a room for people with this class of job to use. They would then want some way of restricting the search of a program entered via that card reader to those peripheral devices specified in this peripheral association group. (See the "PA" System Input Message). In this case, only those devices within the peripheral association group are searched for file assignment for KIND's in the peripheral association as described above.

Port Subfile Assignment

Port files differ from files of other KINDs in that I/O operations and file assignment do not occur between a logical file and a physical device but between subfiles of two or more logical files. In general, when an OPEN statement is executed for a port subfile, the FILESTATE of the subfile changes to OFFERED, and the subfile matching algorithm looks for a complementary subfile. If a complementary subfile is not found, the subfile remains OFFERED until a complementary subfile is offered or the subfile is closed. If a complementary subfile is found, the subfile's FILESTATE changes to OPENED. (Refer to the discussion of the FILESTATE attribute in Section 3 of this manual.)

When a port subfile is offered, the subfile matching algorithm uses the KIND, MYNAME, YOURNAME, TITLE, SECURITYTYPE, and YOURUSERCODE attributes to find a matching port subfile, called the complementary subfile. The values of these attributes for the complementary subfile must relate to the values of these attributes for the offered subfile as follows:

a. The value of the KIND attribute for both the offered subfile and the complementary subfile must equal PORT.

b. The value of the MYNAME attribute for the complementary subfile must equal the value of the YOURNAME attribute for the offered subfile.

c. The value of the YOURNAME attribute for the offered subfile must be null or equal to the value of the MYNAME attribute for the complementary subfile. A null value for the YOURNAME attribute of the offered subfile matches any value for the MYNAME attribute of the complementary subfile.

d. The value of the TITLE attribute for the offered subfile must equal the value of the TITLE attribute for the complementary subfile.

e. If the SECURITYTYPE attributes of both the offered subfile and the complementary subfile equal PUBLIC, then security checking is immediately successful. If the SECURITYTYPE attribute of the offered subfile is PRIVATE, the YOURUSERCODE attribute of the offered subfile must equal the usercode of the program offering the complementary file. If the SECURITYTYPE attribute of the complementary subfile equals PRIVATE, the YOURUSERCODE attribute of the complementary file must equal the usercode of the program offering the offered subfile.

When opening port subfiles, the OPEN statement requires one parameter, the subfile to be opened, and allows a second, called an open option. The subfile to be opened is specified using a file identifier and a subfile index. If the subfile index is zero, all subfiles in the port file with FILESTATE equal to CLOSED are opened. If the subfile index is greater than zero, and less than or equal to MAXSUBFILES, only the specified subfile is opened. If no subfile index is specified and MAXSUBFILES is greater than one, BADSUBFILEINDEX is returned. (BADSUBFILEINDEX and the other errors returned by the OPEN function are discussed below.) If no subfile index is specified and MAXSUBFILES equals one, then the only subfile is opened.

The following three open options apply to port files:

WAIT	This is the default value. The subfile is offered for matching, and the program is suspended until a matching subfile is found.
OFFER	The subfile is offered for matching, and the program does not wait for a matching subfile to be found.
AVAILABLE	The subfile is offered for matching, and a search is made for an available complementary subfile (a matching subfile that has already been offered). If one is found, the subfile is opened; if an available complementary subfile is not found, NOFILEFOUND is returned. The subfile is not left offered for subsequent matching.

The OPEN function returns one of the following values when used to open port subfiles. (When a subfile index of zero is specified, open results for individual subfiles must be obtained through use of the SUBFILEERROR attribute.)

1	The open operation was successful. If the open option was OFFER, the subfile successfully entered the OFFERED state.
2	The open option AVAILABLE was specified, and no available complementary subfile was found.
38	The host became unreachable during the open operation.
40	The specified subfile was not in the CLOSED state.
42	Either the subfile index given as a parameter was less than zero or greater than MAXSUBFILES, or the subfile index was zero, and the open option was AVAILABLE.
44	The subfile index was zero and an error occurred on one or more of the subfile open operations.

Port subfiles are closed explicitly by using the CLOSE statement or closed implicitly when the program that opened the subfile comes to end-of-task. When closing port subfiles, the CLOSE statement requires one parameter, the subfile to be opened, and allows a second parameter, called a close option. If no close option is specified, the program is suspended until the specified subfile is closed; if the close option DONTWAIT is specified, the close operation is initiated, and the program resumes execution immediately. All other close options are ignored.

The CLOSE statement may also be used as a function in which case it returns one of the following values when closing port subfiles. (When a subfile index of zero is specified, close results for individual subfiles can only be obtained by interrogating the SUBFILEERROR attribute.)

1	The close was successful.
30	The FILESTATE of the subfile was CLOSED at the time when the close operation was attempted.

32 Not all messages were acknowledged by the remote host.

42 The subfile index given as a parameter was less than zero or greater than MAXSUBFILES.

46 The subfile index was zero, and an error occurred on one or more subfile close operations.

Circumstances Requiring Operator Intervention

Tape, paper tape, and card files can be declared to be unlabeled files (LABEL=OMITTED or OMITTEDEOF). This lack of label records makes it impossible for the system to automatically associate a physical file with the logical file.

When selecting a permanent file, it is possible that no files or more than one file meets the assignment criteria. In these cases, a NO FILE condition or a DUPLICATE FILE condition exists.

Under these conditions (unlabeled files, NO FILE, or DUPLICATE FILE), the system operator may assign a physical file to the logical file with either an IL or UL System Input Message.

If the operator intervenes in the assignment of a labeled permanent file and DEPENDENTSPECS is TRUE (FILETYPE equals 7 or 8), the attributes affected will be set to the values in the permanent file assigned by the operator.

When selecting a peripheral for a new file, it is possible that no peripheral is available for assignment. In this case, the system operator may assign a peripheral to the logical file with the "OU" System Input Message. The operator may not use the OU message to assign a peripheral with a different general KIND than that requested by the logical file but may assign a different KIND for backup or a different specific KIND within the same general KIND.

If the KIND is PRINTER or PUNCH, there may be no peripheral that matches the FORMID attribute. In this case, the operator may FORM an available peripheral that otherwise meets the requirements of the logical file or may respond with an FM System Input Message.

The operator may also change the requirements for file assignment by changing some of the file attributes with the FA System Input Message.

SOFTWARE TRANSLATION

In general, the translation of records in a physical file is required whenever the mode of the records in the physical file (EXTMODE) is different from the mode of the logical file (INTMODE). Whether or not translation is involved for a file, the attributes MINRECSIZE, MAXRECSIZE, BLOCKSIZE, AREALENGTH, CURRENTBLOCK, and CURRENTRECORD are in terms of the logical units of the file as defined by the INTMODE and FRAMESIZE (or UNITS) attributes. Data are processed by the subsystem in terms of logical records. Both character and word oriented data transfers are allowed while the records are being translated. In fact, the whole process is transparent to the program.

The subsystem takes advantage of hardware translation in the cases where it is provided (for example, card, printer, and tape files) whenever INTMODE equals EBCDIC and EXTMODE equals BCL. Software translation is provided in the cases where hardware translation is not.

Seven-track tape (KIND equal to TAPE7) files (see * in Table 4-2, translation table) have two cases of hardware translation or transformation which all other physical unit types do not. Nonstandard (alpha, even) parity seven-track tapes always have EXTMODE equal to BCL, and the hardware translators allow the INTMODE to be either BCL or EBCDIC. Standard parity tapes coming from the B3500 require BCL-to-BCL or EBCDIC-to-EBCDIC transformation through the hardware. All other combinations are handled in 6-bit binary. When it concerns the hardware, as it does in the nonstandard parity case, all INTMODE's other than BCL are assumed to have an 8-bit frame size.

Paper tape files (see ** in the following translation table) had a special use for the combination INTMODE equal EBCDIC, EXTMODE equal ASCII that was implemented before the system handled 8-bit ASCII. With this combination of modes and the attribute TRANSLATE set to DEFAULTTRANS, the data is sent through the hardware translators. Otherwise, ASCII is transferred in 8-bit binary exactly as EBCDIC is handled.

The TRANSLATE attribute gives the program control over the invocation of translation, from disallowing all translation, hardware or software, to requiring software translation (perhaps to scramble the data) when no translator is required.

Before software translation was implemented, the logical subsystem was inconsistent in disallowing logical files to be opened that required translation when, in fact, no translation was provided. Because of this, the default (TRANSLATE equal to DEFAULTTRANS) invocation of software translation is confined to those cases where programs would have been terminated previously with an invalid translation open error.

Software translation is restricted to the cases where the data in the file is stored in character (HEX, BCL, EBCDIC, ASCII) mode; binary data cannot be translated and must remain unaltered in its 48-bit pattern; that is, logical files, where either INTMODE or EXTMODE equals SINGLE (0) or where, in the case of card files, EXTMODE equals BINARY (6), are never translated. The FORTRAN linked record format (BLOCKSTRUCTURE equals LINKED or FILETYPE equals 6) which implies that the data is binary, and the generalization of the COBOL variable length record format (FILETYPE equals 4) which allows the mixing of modes in the records, defy the possibility of translation. Direct I/O and port files do not receive software translation.

Word oriented data transfer, where the character frame sizes differ between the internal mode and external mode, requires either contraction or expansion of the records such that the logical and physical record and block sizes are not the same. Consider, for example, the difference between an EBCDIC and a BCL word oriented card file. In both cases, each logical record contains 80 characters, but the record size of the EBCDIC file is 14 words and the BCL file is 10 words.

Table 4-2 shows the relationship between hardware translation and the DEFAULTTRANS and FULLTRANS values of the TRANSLATE attribute.

Table 4-2. Table of Translation

INTMODE	EXTMODE	DISK/PACK	PRINTER	READER/PUNCH	TAPE7	TAPE9/TAPE PE	SPO/REMOTE	PAPERTAPE
H	B	F	↓	F	F	F	-	D
H	E	F	F	F	F	F	D	D
H	A	D	↑	-	D	D	-	D
B	H	F	↓	-	F	F	-	D
B	E	D	H	D	F	F	D	D
B	A	D	↑	-	D	D	-	D
E	H	F	↓	-	F	F	-	D
E	B	D	H	H	H*	F	-	H
E	A	D	↑	-	D	D	-	H**
A	H	D	↓	-	D	D	-	↓
A	B	D	↓	D	D	D	-	↓
A	E	D	D	D	D	D	D	D

A means ASCII
B means BCL
E means EBCDIC
H means HEX

D means DEFAULTTRANS (which does not handle cases marked F)

F means FULLTRANS (which includes DEFAULTTRANS)

H means hardware translation

- means not valid combination

↓ ↑ means the EXTMODE is changed to be the same as the case pointed to by the arrow.

For an explanation concerning the * and ** items, see corresponding text.

MV1904

DUPLICATED FILES

The purpose of duplicated files is to avoid the problem of encountering an error on disk which is fatal. The fatal problem occurs when a program does a READ from a disk file, and the MCP tries ten (10) times to do an error free READ without success. At that point, the user is faced with file restoration (also referred to as reconstruction).

Perhaps the best introduction of duplicated files would be to list the various features and functions as seen from the user standpoint:

A disk file is defined to be duplicated if one or more copies exist for each row of the file.

Duplicated files are identified as such at the time of their creation.

Duplicated files are implemented via a Boolean file attribute, DUPLICATED, which may be interrogated at any time.

Setting/resetting the DUPLICATED attribute can only be done if the file is closed.

Within duplicated files, there is no original; that is, since all images of row n are identical, then all images are referred to as copies.

An additional file attribute called COPIES has meaning only for duplicated files. The value of COPIES indicates the total number of physical row images which exist for the logical row; for example, COPIES = 2 indicates two physical rows for each logical row.

The MCP maintains a separate disk file for each COPY of the duplicated file

Those file attributes which apply to a physical row (AREAALLOCATED, FAMILYINDEX, ROWADDRESS, and AREACLASS) can be qualified to indicate the COPY number, where COPY number one (1) is the first, two (2) is the second, etc. For example, FID (m,n).ROWADDRESS, where m is row number and n is copy number.

Duplicated files may be blocked or unblocked. In blocked files, logical READs or WRITEs do not necessarily cause a corresponding physical READ or WRITE; that is, a logical READ may request a record which happens to exist in the current block, or a logical WRITE may specify a record within the current block.

Each time a physical WRITE must be done, the block is written serially to all copies.

The buffer containing the block is tied up until all copies have been written.

If any physical WRITE encounters an error, an attention bit (bit 1) is set in the logical result descriptor.

Each time a logical READ causes a physical READ, the MCP will initiate only one physical READ from only one copy (unless an error occurs on that READ).

Each physical READ is initiated to the next copy of the specified row, where the next copy is determined on a rotational basis.

When an error occurs on a physical READ, the MCP will cause another physical READ but on the next copy of the corresponding row.

An error-free physical READ from any copy will not cause branching to action labels.

An error on a physical READ from any row sets the attention bit for the block. All logical READs from that block will have the attention bit set in the logical result descriptor. Note that this means at least one (but not all) of the copies has an error in its corresponding block.

If all copies encounter an error, then branching to action labels is caused because the exception bit (bit 0) is set in the logical result descriptor.

A duplicated file may have all copies in the non-IAD areas, or all copies may be in the IAD areas. The copies may even be mixed into both types of allocated disk.

Since each copy of a duplicated file has a separate header, normal LIBRARY/MAINTENANCE may be performed on one or more copies by treating each as a separate file (or files).

Direct I/O READ and/or WRITE operations are NOT allowed on duplicated files.

Directory Structure

Each copy of a duplicated file is a complete and self-contained disk file. The header for an individual copy of a duplicated file is marked as being a duplicate.

NOTE

A file which is a copy of a duplicated file cannot have an external file name which consists of less than two identifiers.

LIBRARY/MAINTENANCE Facilities

Since each individual copy of a duplicated file is a separate file on disk, LIBRARY/MAINTENANCE functions may be performed on the copies of a duplicated file, just as on any other disk file.

When a duplicated file, with the external name A/B for example, is created with three copies, they are given the names A/B/"COPY#01",A/B/"COPY#02", and A/B/"COPY#03". Any; or all, of the copies can have their names changed via a System Input Message or file attribute manipulation. As long as the copies have common "first" names (in this case A/B), the "last" names can have any value. When the duplicated file A/B is be opened, the copies will be found. If any of the copies' names were changed to B/C/X say, then when the duplicated file A/B was opened, the file B/C/X would not be considered a copy and, in this example, the number of copies would now be two.

Logical I/O

BUFFER HANDLING

The implementation of duplicated files required additional considerations to the physical aspects of the I/O statements; that is, when a buffer is physically filled or emptied, certain things occur depending on whether a physical WRITE or READ is necessary.

PHYSICAL WRITE

Any time a buffer needs to be emptied (written on disk), the buffer (and the program's use of that buffer) is "suspended" until a physical WRITE has been performed to every copy.

A "result descriptor" is available for the overall buffer emptying, and a result descriptor is also available for each copy's physical WRITE. These may be investigated via the file attribute ERRORTYPE.

PHYSICAL READ

Whenever a buffer required filling (a READ from disk), there is only ONE physical READ initiated, and the copy number is determined on a rotational basis.

RESULT DESCRIPTOR(S)

An "overall result" is available for each logical READ or WRITE statement. This has always been the case, and the "action labels" associated with the READ or WRITE are automatically invoked, depending on the overall result. In fact, certain bits of the result (type "BOOLEAN") are investigated if the low-order bit is TRUE, to determine which error had occurred; specifically:

Bit 9 - End of file (EOF).

Bit 7 - Irrecoverable parity error.

Bit 4 - Format-versus-data error.

The user can use READ and WRITE statements as a Boolean expression:

```
BOOLEAN RESULT;
RESULT := READ (F, 10, A[*]);
IF READ (FYLE, FMT, LST) THEN . . .
```

The combination of action labels and using the READ/WRITE statement as an expression is NOT allowed in the same statement; either action labels are used, or the I/O statement is used as an expression, or neither is specified.

The overall result for a duplicated file READ/WRITE contains additional bits of the word:

Bits [27:16]	Error bit for each copy. Note that bit 12 is copy number 1, bit 13 is copy number 2, etc. This is valid only when bit 1 is TRUE and bit 0 is FALSE.
Bit [1:1]	Attention bit. This bit is TRUE if there are any bits on in [27:16] and bit 0 is FALSE.

Note that the overall result for a duplicated file I/O is TRUE (bit 0 = 1) only if all copies are in error.

Assume the following is in a program:

```
BOOLEAN RESULT;

RESULT := READ (F, 10 A[*]);
```

RESULT would contain the overall result of the READ statement. A fatal error could be detected via:

```
IF RESULT THEN GO TO HANDLECATASTROPHE;
```

However, when using duplicated files, the surveillance for a bad copy could be performed:

```
IF RESULT.[1:1] THEN GO TO HANDLEACOPYPROBLEM;
```

Since [1:1] is TRUE if any copy has a problem, it is conceivable that more than one copy has a problem.

```
X :=ONES (REAL(RESULT.[27:16]));
```

would produce an integer count of how many copies have problems.

The determination as to which copy has a problem is simplified by using the FIRSTONE intrinsic:

```
X:= REAL(RESULT.[27:16]); Y:= FIRSTONE (X);
```

Y would contain the copy number of the last copy that has a problem. Following corrective action or initiation of corrective action, the corresponding bit number of X would be reset and another test of X would be in order:

```
X.[Y-1:1] :=0;

IF Y:=FIRSTONE (X) NEQ 0 THEN GO TO HANDLEANOTHERCOPYSPROBLEM;
```

The subtract (i.e., in [Y-1:1]) is necessary, since FIRSTONE returns the bit number + 1; the result of FIRSTONE on a zero word is zero, if no bits are 1.

In concluding the overall result explanation, the FIRSTONE intrinsic works out very handily since investigation of specific result descriptors are requested by indicating the copy number, where 1 is the first, 2 is second, etc.

File Restoration/Reconstruction

The DISK FILE DATA RECOVERY section of this document discusses nonduplicated disk file failures, verification of written data, dump and audit trails, etc. This section addresses duplicated file failures and recovery.

IRRECOVERABLE READ PARITY ERROR

In the case of an irrecoverable read parity error, the MCP will proceed to another copy and try again (up to ten times) to do a successful READ operation. This process continues until a good read occurs or until ALL copies have been tried and failed.

In the majority of instances, experience has shown that when an irrecoverable read parity error occurs, if the record could be restored and rewritten (even by manual intervention via the maintenance panel), the record (block) could be subsequently read with no difficulties. Duplicated files provide a simple means of restoring a record (block). Consider a READ statement whose overall result indicates that one or more copies had an error (the attention bit only), but whose buffer contents are valid (since by definition it was filled by an error free READ from some copy). If the record (block) is to be subsequently written due to updating one or more fields therein, the WRITE will cause write broadcasting of the buffer contents to all copies (including any and all which had an irrecoverable read parity error).

IRRECOVERABLE FAILURES WORSE THAN READ PARITY

The foregoing discussion dealt with the kind of problem that constitutes the bulk of failures that can and do occur on a disk file system.

This portion of information concerns duplicated file recovery techniques that can be employed for the class of failures which are far more serious: head electronics failures, head crashes, storage unit failures, and electronics unit failures.

It should be noted that with multiple copies in a duplicated file, the head electronics, head crashes, storage unit failures, and electronics unit failures do not require immediate abandonment of the user programs. In other words, it is a catastrophe only if ALL copies encounter an error in the same record (block). When a buffer needs filling, the MCP finds a good record (block); and in the case of buffer emptying, the MCP does a broadcast write operation, thereby writing the information to those copies which can receive it.

However, it is prudent to consider the options available when the offending head, storage unit, or electronics unit is rehabilitated. Knowledge as to which rows of which copies are in error must be accumulated during normal processing. Then, when the hardware unit's rehabilitation is complete, simply do a READ and immediate re-WRITE of each and every record (block) of all rows which had an error(s). However, it should be noted that some of the reads will be done from bad copies, which is a problem.

If accumulated knowledge indicated which copy (or copies) is (are) not in error, a good copy's corresponding row could be read and used accordingly.

Another option available could be termed row relocation. This amounts to creating a one row nonduplicated file on another area of disk, copying from the row in error into the new row, and then doing an EXCHANGE on it and the bad copy's row. This method also requires care in that the corresponding rows of all the other copies in the host duplicated file must not be altered during the copying process. However, once the new row has been copied and verified that it agrees with the host duplicated file, normal processing can resume following the EXCHANGE. The syntax is:

```
EXCHANGE(NEWROW[0], DUPFYLE[ROWNUM, COPYNUM])
```

The above syntax indicates the following:

a. The second file (DUPFYLE) is the receiving file.

b. Rows are numbered from zero through whatever.

c. Copy numbers vary from one through the number of copies.

DISK FILE DATA RECOVERY

Introduction

This section describes the programming techniques employed by a user program to detect and recover from hardware errors associated with the disk file subsystem.

It should be emphasized that it is the user program's responsibility to be prepared for and to recover from such errors. This position is justified by the realization that knowledge of the data file structure, audit information, and the points at which the program can be restarted are known only to the user program. Any generalized MCP recovery procedure would have to save an inordinant amount of data, (most of which would not be necessary for data recovery), and would therefore be extremely inefficient. Therefore, this section describes the manner in which the designers of the system software have envisioned that disk file data recovery will be performed, and also describes the MCP and user language facilities provided to permit such error recovery action to take place. This section is divided into the following eight topics:

a. Disk file subsystem failure modes.

b. Data file recovery units.

c. Dumps and audit trails.

d. Normal I/O failure detection.

e. Direct I/O failure detection.

f. Row construction.

g. Data recovery techniques

h. Disk directory reconstruction.

Disk File Subsystem Failure Modes

In order to understand the preparatory steps for recovery, and the recovery action itself, it is necessary to discuss the ways in which the disk file subsystem can fail. No attempt is made to discuss the frequency of such failures. It can be said, however, that the probability of such failures increases as the number of disk units in the configuration increases. Hence, in a configuration that contains 30 or more storage units, disk subsystem failures can be anticipated, and data error recovery should be considered to be a part of the programming task. A brief description is provided for the following disk file subsystem failures:

a. Simple (recoverable) read parity errors.

b. Head electronics failures.

c. Head crashed.

d. Storage units failures (storage unit not ready).

e. Other failures.

SIMPLE (RECOVERY) READ PARITY ERRORS

Standard MCP action on disk read parity errors is to retry the read operation (up to 10 times) to determine if the data can be brought in correctly. If this retry action is successful, the error is classified as a simple read error. At this time, the object program continues as though no error had occurred. Should the object program access the related I/O buffer during this retry action, all indications will show that the I/O operation is still in process. There is no way an object program can discover that a simple read error has occurred, as the required data recovery action has been successfully performed by the MCP.

HEAD ELECTRONICS FAILURES

Head electronic failure conditions exist when there have been electronic failures in a disk file read/write head, which make it impossible to read the information stored under that head. Under these circumstances, many records are unreadable. A description of which disk addresses will be unreadable is extremely complicated, and is dependent upon the particular model of disk involved, and is therefore inappropriate for this document.

HEAD CRASHES

This condition occurs when sensors in the storage unit detect that a head has touched the surface of the disk. The heads immediately retract so as to minimize physical damage to the disk surface, and the entire storage unit (SU) appears not ready to the system. Note that any SUs on either side of the Not Ready SU are not affected, and may still be accessed. Recovery techniques involving manual intervention (and corrective action to the SU so that the information under all other heads becomes accessible after servicing has been completed) have not been considered.

STORAGE UNIT FAILURES (STORAGE UNIT NOT READY)

This condition can occur due to loss of power, or head crashes. The recovery technique of assuming that the data will be accessible after a power recovery has not been considered. Recovery techniques involving the reconstruction of the data are described in a later section.

OTHER FAILURES

Several failures are beyond the scope of this discussion, since manual detection and corrective action are required. In this class are those incurred in the electronics unit, disk file control, I/O processor, and central processor. It is not considered practical to attempt to program for these types of failures.

TEXT DELETED

Data File Recovery Units

The key to efficient data recovery is to segment the data into several small parts, so that recovery of any of the small parts can be accomplished much faster than recovery of the entire data. In order to ensure that a failure of some hardware component will not necessitate the simultaneous recovery of several of the small parts of the data, one must be able to specify that only one recoverable data segment be allocated to each hardware unit that might fail or, rather, to each hardware unit that can be serviced. For these reasons, the smallest recoverable segment of the data file shall be considered to be a row and it shall be allocated to a storage unit or electronics unit.

Multiple data files critically related to a given application introduce another dimension for disk file allocation, organization, management, and reconstruction processes. Recovery time and complexity are minimized if such files, or rows, reside on different storage units. Containment of the file or row within a storage unit lessens the probability and severity of failures even further. Failure frequency and effects are again reduced if related file never share electronics units. User-controlled distribution of rows and files is specified by the file attribute AREACLASS. This attribute has a range of zero to 255 as possible values, with zero used for the default case. An AREACLASS must be assigned to a row prior to assignment of disk space to that row. Assignment of an AREACLASS to a row which has been assigned a disk address will be ignored. The syntax for the AREACLASS attribute is as follows:

```
FILEID(ROWNO).AREACLASS:=(AEXP);     (ALGOL)
or
SET FILEID (AREACLASS,ROWNO) TO 1.   (COBOL)
```

Electronics units assume and maintain an AREACLASS value until all rows with that classification are removed. Acquisition of disk space is from an electronics unit with the same AREACLASS, if possible. Inability to obtain required space on an electronics unit with a class of zero. When an item with a nonzero AREACLASS has been assigned in an electronics unit, then no items with a different, nonzero AREACLASS will be allocated in that electronics unit. Space requests with a zero AREACLASS will be obtained from any electronics unit containing the required available storage.

All space allocated with a nonzero AREACLASS and AREASIZE less than or equal to a storage unit size will be contained within one storage unit. Files or rows may share storage units if sufficient space is available.

Dumps and Audit Trails

The basic technique for data recovery is to take a complete dump of the disk file occasionally and, from that point onward, to record (for example, on magnetic tape) all transactions written to the file. Thus, the file can be reconstructed by reloading the dump, and then reading each of the audited updates and writing them into the reloaded file.

The basic technique just described is simply stated, and purposely ignored questions such as the following:

a. How frequently the file should be dumped?
b. How many back-up dumps should be kept?
c. Should dumps and audits go to tape or disk?
d. Should dumps and audits be made in duplicate for safety?

One matter that should be considered here is the fact that both the dumps and audits should be row oriented. If this is accomplished, then reconstruction can be performed in a much more rapid manner. In the extreme case, suppose each row of the file were dumped to a separate reel of tape. Then, in order to perform data recovery on a row, only that one reel of tape would have to be passed, greatly expediting reloading the row. Similarly, if the audit trails of each row were directed to a separate reel of tape, then the updating of a reloaded row is greatly expedited since no information need be passed which will not be used.

It can be assumed that there is some practical point of compromise between data recovery time and the number of tape units one might choose to have on-line. If a file contains 60 rows, recovery will be fastest if 60 tape units are utilized. If the configuration permits only 10 tape units to be used for this purpose, then the audits for six rows must all be directed to the same tape unit, and recovery of any one of the rows will be prolonged by the necessity to read past audit information which does not apply to the row being reconstructed.

Since the MCP Library Maintenance routine does not dump or load files in a row-oriented manner, it is assumed that a user program will be written to perform the dump and load operations. Similarly, it is the responsibility of the user to assign names for, and to keep track of, the various tape files.

As will be shown later, the user program has no difficulty in determining which disk file row must be reconstructed. This information (the row number) and the file name are all that is required to determine which dump tape(s) and which audit tape(s) must be used for row reconstruction.

Normal I/O Failure Detection

The detection of disk read parity errors, disk read check errors, and storage unit not-ready conditions occur in the MCP at the time the I/O complete interrupt is acknowledged and the result descriptor is scanned in. The MCP will automatically perform retry procedures on all potentially recoverable disk errors. When these retry algorithms are unsuccessful, the appropriate buffer is marked to indicate that the I/O operation is finished, but that an error condition exists.

DETECTION THAT AN ERROR HAS OCCURRED

In normal I/O, the program is made aware that an error condition has occurred by causing it to take an error action branch. (If this error action branch is not provided, an irrecoverable error condition will cause the program to be terminated.)

On input, the error action branch is taken on the READ statement which would have permitted access to the data in error. If there is any data (as in the case of read parity), the MCP will transfer this data to the working areas of the program (as specified by the READ statement) before the error action branch is taken.

On output, the error action branch is taken on the WRITE statement which reuses the buffer that originally had the error. Thus, if a file is to have three buffers, the error will not be detected until the third WRITE statement after the error is detected. This last WRITE statement is executed normally.

DETERMINATION OF WHICH RECORD IS IN ERROR

The file attribute RECORDINERROR has the value of the logical record in error. RECORDINERROR always corresponds to the most recent error on the file; that is, the error associated with the most recent transfer to an action branch.

DETERMINATION OF WHICH ERROR CONDITION EXISTS

The file attribute ERRORTYPE describes the error incurred. Values for ERRORTYPE are as follows:

0	No error
1	Storage unit not ready
2	Read parity error
3	Read check failure

NOTE

ERRORTYPE, like RECORDINERROR, refers to the most recent error on the file.

DETERMINATION OF WHICH ROW IS IN ERROR

If a user's program cannot read a record, then the program can calculate the row number of the error as follows:

```
ROWNO := (RECNO * DISK.MAXRECSIZE) DIV DISK.AREALENGTH;
```

If the problem is caused by the failure of a storage unit, then the user can perform the following:

a. Run his program until the unit is referenced and calculate the row number as mentioned above, or

b. Obtain the contents of the storage unit from the disk directory.

Examples

ALGOL

```
READ (DISK[X], 30, B [*]) [EOF:ERR];
        .
        .
        .
        ERROR:IF DISK.ERRORTYPE = 1
        THEN GO TO SUNOTREADY ELSE ROWRECOVER (DISK.RECORDINERROR);
```

COBOL:

```
DECLARATIVES.
DI SECTION
USE AFTER STANDARD ERROR PROCEDURE ON DISK-OUT.
L1.MOVE DISK-OUT (ERRORTYPE) TO ERRORVALUE.
END DECLARATIVES.
.
.
.
L2.WRITE DISK-OUT.
IF ERRORVALUE = 3 GO TO L2 ELSE
IF ERRORVALUE = 1 PERFORM SU-NOT-READY-RECOVERY.
```

EMPLOYMENT OF SOFTWARE RESULT DESCRIPTOR

A program may also obtain information regarding the result of an I/O operation by means of the result descriptor returned by the I/O routines. The following are permissable language constructs:

```
ALGOL:     A := READ(CARD,FMT,X);
FORTRAN:   READ(5,1,RESULT=A)X
```

When either of these statements is executed, a result descriptor will be returned in the single-precision Boolean variable A. Refer to the STATE attribute for a description of the field contents of this data word.

Direct I/O Failure Detection

Detection that an error has occurred. In Direct I/O, it is the responsibility of the user programmer to check if an error has occurred. This check can be made using the buffer attribute IOERRORTYPE.

TEXT DELETED

DETERMINATION OF WHICH RECORD IS IN ERROR

The buffer attribute IORECORDNUM contains the logical record number of the record in the buffer.

DETERMINATION OF WHICH ERROR CONDITION EXISTS

The buffer attribute IOERRORTYPE is used for Direct I/O. Refer to the discussion of IOERRORTYPE in Section 3 of this manual.

TEXT DELETED

DETERMINATION OF WHICH ROW IS IN ERROR

The determination is the same in Direct I/O as in normal I/O.

Examples

```
DIRECT FILE DISK (KIND=1,...);   (ALGOL)
DIRECT ARRAY A[0:9];
READ (DISK, 10, A[*]);
IF BOOLEAN (WAIT(A[*])) THEN
    IF A.IOERRORTYPE=1, THEN GO TO SUNOTREADY
    ELSE ROWRECOVER(A.IORECORDNUM);

READ FI INTO A.        (COBOL)
WAIT A ON EXCEPTION
    MOVE A(IOERRORTYPE) TO ERRORVALUE.
```

Row Reconstruction

Row reconstruction can be performed while the other jobs are running, or it may be performed at some later time. In any case, it is assumed that the following information is known to the row reconstruction when it is initiated:

a. The name of the file to be reconstructed.

b. The row number of the row to be reconstructed.

c. The name (including volume number, revision number, creation date, etc.) of the tape (s) containing the latest dump of the row to be reconstructed.

d. How to find the proper row on the tape.

e. The name (including volume number, revision number, creation date, etc.) of the tape(s) containing the audit information for that row.

f. How to identify an audit item on the tape, and where that audit record must be placed in the file.

An outline of the row reconstruction is as follows:

a. Declare the file that must be reconstructed. This declaration must be identical to other declarations for this file. To simplify the discussion, suppose this file is called FILEID.

b. Declare a new file, containing any number of rows, but otherwise identical to the original file. (To simplify the discussion, assume this file is called NEWROWFILE.)

c. If the FAMILYINDEX of the original row is to be maintained, execute a statement to cause the FAMILYINDEX attribute in the NEWROWFILE file to be the same as the FAMILYINDEX of the FILEID file.

ALGOL:

```
NEWROWFILE(NEWROWNO).AREACLASS:=FILEID(ROWNO).AREACLASS;
```

COBOL:

```
SET NEWROWFILE(AREACLASS,NEWROWNO)
TO FILEID(AREACLASS,ROWNO).
```

a. Read the proper dump tape, writing the proper records into NEWROWFILE.

b. Read the proper audit trail tape, writing the proper records, at the proper record number (key).

c. Having finished creating a file called NEWROWFILE, containing the proper information, open FILEID if it is not open.

d. Close both files in a manner which will retain them. Either REWIND or FILE.OPEN:=FALSE.

e. Execute the statement:

ALGOL:

```
EXCHANGE (NEWROWFILE[NEWROWNO], FILEID[ROWNO]);
```

COBOL:

```
EXCHANGE NEWROWFILE(NEWROWNO) WITH FILEID(ROWNO).
```

Data Recovery Techniques

For a data base to be recoverable, it must be designed with recovery in mind. Files for which data recovery is intended should be set up so that their rows have nonzero AREACLASS; thus, the rows will fall within storage unit boundaries and a failure will be limited to a single file row. The programs which manipulate these files must follow conventions so that the information needed to recover a row of a file is available to the recovery routine. The two most useful techniques of keeping reconstruction information are via audit files and completely duplicated backup files.

The trade-offs between audit files and duplicated data are involved in the cost of recovery time. If immediate access to all data is required, then a master file-backup file organization is required. If the lost data can be ignored for a period of time, then an audit file can be processed against an old generation of the data to recreate the state of the data base prior to the failure. Audit files must contain a description of the change made to the master data base.

The number of audit files and the frequency of producing data base generations are functions of the time available for data reconstruction. In fact, the audit file and data generation in the extreme case can become one duplicate copy of the data base.

Record-processing discipline must be enforced in programs which alter the data base so that a record is always completely processed before any new update is started. The complete processing not only includes the data base update, but also includes the creation of the audit record or backup file update.

Audit files should be organized so that information required to reconstruct the records of a file row are grouped. This grouping is necessary to minimize data collection time in the recovery cycle.

On detection of an error, it is necessary to determine the severity and scope of the failure. The severity of the error is a combination of the difficulty of reconstructing the data and length of time which the system can function without said data. The scope of the error is dictated by the type of failure and the file organization of the data base.

For instance, an irrecoverable read parity error might affect only one record or part thereof. An electronics unit failure could cause loss of several rows of a file. The existence of a backup file means that no data is really lost, and all that is required is scheduling a reconstruction function for that row to guard against loss by future errors. If an audit file is the only source of reconstruction information, then a reconstruction program must recreate the data before it is usable. If the need for the data is critical, then the reconstruction process may begin immediately; otherwise, it can be postponed to a more convenient time. In any case, the information system must flag the data as being unavailable so that other processes do not repeat this decision process.

The simplest method of audit file recording is to record each alteration request. Then, when there is a loss of data, the transactions are merely reprocessed against the base data file. Other techniques are to record the changed field or to make a copy of the updated record. In both these cases, a special routine is needed to make the appropriate alteration to the base file to create the up-to-date data.

The process of reconstruction begins after the failed units are isolated. In the case of a storage unit type failure, the rows involved must be replaced with functioning storage units from the disk file pool. The last dump of the rows must be loaded into the file; then the reconstruction program processes the necessary audit files against this version of the data until all transactions are completed. At this point, the indicators which flagged the data as being unavailable are removed and processing continues as before the failure.

DISK FILE AND SYSTEM-ACCESS SECURITY

Introduction

The system provides a mechanism for identifying system users, limiting system access to valid users, and restricting disk file operations by securing disk files against operations by users other than the owners of the files.

Each individual user of the system is identified by a unique usercode, which is normally assigned by the installation management. One or more passwords may be associated with each usercode to prevent unauthorized use of the usercode. System access control is achieved by requiring a user to specify usercode/password before any useful processing is performed. Basic file security is achieved by associating a given user's file with his usercode. By default, a user is restricted to accessing his own files. Mechanisms are available for a user to access a system-global library of files, and files in another user's library, subject to constraints placed by the owners of those files.

Installations not desiring to use these facilities may ignore them, except that file titles with USERCODE as the first identifier are disallowed. However, some system functions require a "privileged" usercode; usercodes must be defined to utilize this function.

Usercodes and Passwords

Usercodes and passwords are identifiers satisfying the same syntax rules as file identifiers; each comprises 1 to 17 EBCDIC characters, normally digits and upper-case letters. Nonalphanumeric characters may be included only if the identifier is enclosed in quotes; the quotes are not counted in the identifier length, and the identifier may not contain a quote.

The usercode is a matter of somewhat public record; it is assigned and controlled by the installation and displayed on various output media. The password should be known only to the usercode holder; it can be assigned or changed by him, and is never displayed.

ASSIGNMENT OF USERCODES

A usercode is defined by its appearance in a file called SYSTEM/USERDATAFILE, which is maintained by installation management to keep information about system users. Facilities to create and maintain this file are described in the USERDATA-MAKEUSER MANUAL. New usercodes may be created through the SYSTEM/MAKEUSER program, or the intrinsics USERDATA and MAKEUSERCODE. At the installation's option, the system operator may create usercodes through the MU and PU system input messages.

CONTROL OF PASSWORDS

Considerable flexibility is available in the use of passwords. In some environments, where security matters are unimportant but it is desired to avoid confusion of one user's files with another's, passwords may be unnecessary. In other environments, it may be desired to allow several users to use the same usercode; therefore, more than one password may be associated. The installation may specify, for each usercode, the minimum and maximum number of passwords allowed. When multiple passwords are assigned, any one of them may be used with the usercode to gain access to the system. No log record is kept of the password used, and no different privileges accrue (to different passwords) (except in the control of other passwords). Passwords may be assigned by the installation. Within the installation imposed restrictions, passwords may be assigned or changed by the users.

ACCESSCODES AND PASSWORDS

An additional level of control may be imposed on the access of a file via an ACCESSCODE and its associated PASSWORD. This feature enables the control of access rights to a file among different users with the same usercode. ACCESSCODES are maintained in the ACCESSCODELIST of the USERDATAFILE together with the optional password(s).

System Access

Initial access to the system is achieved by entering a job through a card reader (or operator console) or by entering information from a terminal via the Data Comm system. In the former case, the access is controlled by the Work Flow Language compiler (WFL); in the latter case, a Message Control System (MCS) program is involved. Once a program is running, it can access other by means of the Inter-Program Communication (IPC) constructs in various programming languages, or by means of a ZIP statement.

JOBS

A usercode may be associated with a job by including a USER statement among the job statements. The job itself and any tasks performed within the job will have the specified usercode. The usercode for the job (and for any tasks yet to be started) may be changed by the appearance of another USER statement. The user statement has the form:

```
USER=usercode/password
```

The equal sign is optional; the /password is omitted if no password is associated with the usercode.

A USER card entered at a card reader remains in effect until the end of the job, or until another USER card is encountered.

The system operator may require all jobs entering by means of a particular card reader to contain USER statements. The requirement is enforced with an SR (secure reader) message; it may be relaxed with an RR (release reader) message.

DATA COMM

Individual MCS programs may accept or require usercode specifications. SYSTEM/CANDE requires all users to log in with usercode. SYSTEM/RJE can accept or require (at the option of the system operator) usercode specification to log in the RJE terminal, and to run individual jobs. The USER statement is used in RJE jobs as it is in WFL jobs. In addition, RJE has the capability of running each job with a default usercode, i.e., the one with which the terminal was logged in.

PROGRAMMATIC

When a task is RUN, PROCESSed or CALLed via the IPC mechanism, a usercode/password may be supplied by means of the pointer-valued task attribute USERCODE. For example, the ALGOL syntax is:

```
REPLACE task.USERCODE BY "usercode/password."

        or

REPLACE task.USERCODE BY pointer
```

where the literal string or the pointer provides a usercode and password (if required), followed by a period. If no usercode is specified for the task, it takes on the usercode of the parent task, if any.

A task may change its usercode by replacing MYSELF.USERCODE by a new usercode/password combination. A task running without a usercode may acquire one by the same mechanism. A nonprivileged task running with a usercode may not remove its usercode.

An error in usercode or password is fatal to the program attempting to supply it (unless that program is privileged).

ZIP

A ZIPped array or file is processed by WFL and may include USER statements. If none is provided, the ZIPped job will be run with the usercode of the task doing the ZIP.

File-Naming Conventions

Disk file TITLEs consist of two parts: the FILENAME and the FAMILYNAME. The FILENAME specifies what usercode the file will be stored under and a name which identifies the file. The FAMILYNAME specifies the physical disk family on which that file is stored.

FILENAME

Disk files created by a program running with a usercode are associated with that usercode by storage in a library (a node of the B 7000/B 6000 hierarchical disk file directory) which is identified by the usercode. In normal situations, the user supplies an arbitrary file name (consisting of 1 to 14 identifiers, including the usercode); the directory structure prevents confusion of that file with any other user's file of the same name. All the files in a given usercode library belong to that user.

A user may not create a new file that does not belong to him but may, however, legitimately require access to general system files and those of other users. To facilitate this access, the following conventions have been adopted:

a. An asterisk as the first character of a file title indicates that the file will be found among the general system files. For example:

```
FILE F (TITLE = *SYSTEM/CARDLINE)
```

b. Parentheses around the first identifier of a file name indicate that the identifier is a usercode and that the file, the title of which consists of the subsequent identifiers, will be found among the files belonging to that user. A <slash> is not required or permitted between the closing parenthesis and the start of the file name. Example:
FILE F(TITLE = "(HISUSERCODE)HISDATAFILE.")

c. In the normal case, when neither of the above conventions is used, then the search will begin with files belonging to the user. If the file is not found, the search will continue among the general system files.

Use of the "*" or (<usercode>) conventions to create a new permanent file under a usercode other than that under which the program is running results in a security violation.

The following examples are syntactically correct.

RUN card example:

```
RUN (HISUSERCODE)HISDIRECTORY/HISPROGRAM
```

Label equation example:

```
FILE ZZZ (TITLE = *SYSTEMFILE)
```

Task code file TITLE example:

```
REPLACE T.NAME BY "(AUSERCODE)PROGRAMNAME.";
```

FAMILYNAME

Currently, the selection of random access mass storage devices varies depending upon whether the KIND attribute has a value of DISK or PACK and whether or not the FAMILYNAME attribute has been assigned a value. If the KIND value is DISK, then the system supplies the FAMILYNAME of "DISK" whether or not the FAMILYNAME attribute has been given another value by declaration, label equation, or direct assignment. If the KIND value is PACK, then the system supplies the FAMILYNAME of "PACK" unless it has been explicitly set to another value. In other words, the FAMILYNAME (even if assigned a value) is ignored if KIND is DISK and has two different default values if one has not been explicitly assigned.

As of the 3.3 release, the values DISK and PACK of the KIND attribute will be synonymous, and the default value of FAMILYNAME will be "DISK" for both disk and disk pack files. This has two implications:

a. The FAMILYNAME attribute will be used when assigning files with KIND equal to DISK so that label-equated FAMILYNAMEs will not be ignored.

b. The assignment of files on the family "PACK" will require the explicit setting of the FAMILYNAME attribute (or the intervention of family substitution).

To aid the transition to a single default value for FAMILYNAME, a run-time warning is given whenever a file from the family named "PACK" is assigned to a logical file that is declared with KIND=PACK and the FAMILYNAME has not been assigned. The warning message is of the form:

```
ATTRIBUTE ERROR: <intname>. FAMILYNAME @ (<line number>)
ON MARK 33, "FAMILYNAME=PACK" MUST BE SPECIFIED
```

The warning and the noted problem can be eliminated by changing the logical file declaration to include FAMILYNAME="PACK".

File Access: Security Attributes

Access to a given disk file is controlled by three attributes of the file: SECURITYGUARD, SECURITYTYPE, and SECURITYUSE.

SECURITYGUARD ATTRIBUTE

If SECURITYTYPE is GUARDED or CONTROLLED, then the title of the guardfile is specified by the SECURITYGUARD attribute. (If SECURITYGUARD is not specified for a GUARDED or CONTROLLED file, or if the specified file is not present, then the file is treated as PRIVATE.) The guardfile may be constructed using the SYSTEM/GUARDFILE program. (Refer to the B 7000/B 6000 System Software Operational Guide, Volume 1.)

SECURITYTYPE ATTRIBUTE

The SECURITYTYPE attribute specifies who, apart from the owner, may access the file. The attribute values are PRIVATE, PUBLIC, GUARDED, or CONTROLLED. PRIVATE means that only the owner may access the file. PUBLIC means that anyone who knows the usercode and title may access it, using the (usercode) title form of file name. GUARDED indicates that the file is "guarded". CONTROLLED is equivalent to GUARDED except the guard file will be involved even for the creating (owning) usercode. The ACCESSCODE is used to govern the access of CONTROLLED files.

Associated with any disk file may be another file title which denotes its "guardfile." If the user opening the former file is neither the owner of the file nor a privileged user, then the guardfile is sought. If the guardfile is found and has the correct structure and format, then access to the other file will be allowed as determined by the information in the guardfile; otherwise, access will be denied.

The default value of SECURITYTYPE is PRIVATE for files created by programs run under a usercode, or PUBLIC otherwise.

SECURITYUSE ATTRIBUTE

The use which may be made of a file is further controlled by the SECURITYUSE attribute, which may have values SECURED, IN, OUT, IO, indicating respectively that the file may be neither read nor written, read only, written only, read or written. The default value is IO.

Attribute Interactions

If a program running under a usercode attempts to access a file in that usercode library and SECURITYTYPE does not equal CONTROLLED, usage is determined by SECURITYUSE. If a file in some other usercode library is accessed, three cases arise: if SECURITYTYPE is PRIVATE, the usage for that program is set to SECURED. If SECURITYTYPE is PUBLIC, usage is determined by SECURITYUSE. If SECURITYTYPE is GUARDED or CONTROLLED, usage is determined from data in the guardfile specified by SECURITYGUARD, and SECURITYUSE is ignored. A security error is detected when some action contrary to the allowed usage is attempted.

Execution of a code file by a nonowner is permitted for a PUBLIC file independent of SECURITYUSE, and is controlled for a GUARDED or CONTROLLED file by the data in the guardfile. In an IPC (Inter-Process Communication) environment, the security tests for the code files are based upon the usercode of the new task, not the parent.

Attribute Specifications

The security attributes for a file may be specified by the SECURITY control statement in WFL, by the SECURITY command in CANDE, or programmatically. Only the owner of the file (a user whose usercode is the same as the usercode specified in the FILENAME of the file) or a privileged user may change the security attributes of the file.

The three security attributes may be specified programmatically, subject to the following constraints:

a. To read the attribute: If the file is not OPEN, then the attribute must have been explicitly set prior to the READ attempt and subsequent to the last CLOSE on the file. If the file is OPEN, the value returned reflects the current attribute setting for the permanent disk file, which may not reflect the privileges afforded to the program examining the attribute (since those privileges were determined at open time).

b. To set the attribute: If the file is OPEN, then the attribute is changed immediately; for a permanent file, the new setting is recorded with the file. If the file is CLOSEd, the attribute will be changed whenever the file is opened. Note that if that file is CLOSED, the title changed, and another disk file OPENed through the same file declaration, it too will have the security attribute changed. Changing a security attribute on an OPEN file causes the permanent copy of the attribute value to be changed, but has no effect on the privileges of any program that already has the file OPEN.

Library Maintenance

A user may remove or change the TITLE of any file that "belongs" to him/her. A user may copy to tape any file that he is allowed to read, with the exception that the "COPY =" function is restricted to his own files.

TEXT DELETED

Privileged Users

In order to perform necessary library maintenance and archiving functions, an installation management requires a means of accessing files that are normally secured. In addition, system security requires that some necessary system functions must be limited to their proper application and denied to general users. These needs are fulfilled by providing the concept of a privileged usercode, and the parallel concept of a privileged task.

A privileged usercode is one marked as privileged in the SYSTEM/USERDATAFILE entry. The privilege may be afforded or withdrawn by the installation management via the SYSTEM/MAKEUSER program, or (at installation option) by the system operator via the MU and PU ODT messages. Refer to the discussion of the SYSTEM/MAKEUSER utility program in the B 7000/B 6000 Series System Software Operational Guide, Volume 2.

A privileged task is, in general, a task running with a privileged usercode or a task whose code file has been marked as privileged by the PP ODT command. MCS programs and certain MCP tasks are automatically privileged. All file security limitations are bypassed for a privileged task. (The default selection of a usercode library remains in effect, however, for a privileged task with a usercode.) A privileged task may change its usercode with the MYSELF. USERCODE replacement. If the new usercode/password combination is not valid, the task's usercode remains unchanged and a task attribute error is noted; it may be tested via the task.TASKATTERR attribute. If the new usercode/password combination is valid, then the task privilege status is set to the privilege status of the new usercode. A privileged task may remove its usercode association; for example

```
REPLACE MYSELF.USERCODE BY "."
```

The privilege status of the task is not affected.

For maximum security, installations should carefully limit the creation of privileged usercodes.

Security Violations

Security in input/output operations are first checked when the file is opened. If the READ or WRITE involved is used as a Boolean primary, the violation is nonfatal and bit 16 is set in the software I/O result descriptor. (See STATE attribute description.)

All violations are recorded in the system summary log, with the mix number, job number and usercode of the violator, along with a violation code and the title of violation subject (typically the file in question). Violation codes are defined with the format of the log messages. (Refer to the B 5000/B 6000/B 7000 Series Operator Display Terminal Reference Manual.)

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

APPENDIX A
FORMAT OF EXTERNAL FILE NAMES

File names are composed of a series of identifiers (file identifiers, volume identifiers, and file directory identifiers) separated by slashes. An identifier may be of any length, although only the first 17 characters of an identifier are used. A file name may be preceded by an asterisk (denoting a system library file) or by a usercode in parentheses (denoting a user library file). As many as 14 identifiers may be used in constructing a file name.

Examples of File Names

A
B/C
D/E/F
G/H/I/J
(U)V

In the preceding examples, A, C, F, J, and V are file identifiers: B, D, E, G, H, and I are file directory identifiers; U is a usercode. Tape files allow only two identifiers to be preserved in the tape label records, and the third and fourth examples would be preserved on the permanent tape file as D/F and G/J.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

APPENDIX B
FORMAT OF LIBRARY TAPES

The basic format of the tape is a USASI multifile file of at least n + 1 files, where n is the number of files dumped. If the tape name is MTA, the files are named MTA/FILE000, MTA/FILE001, etc., with MTA/FILE000 the tape directory.

General tape layout follows:

VOL1

HDR1

HDR2

tapemark

one or more records of 901 words each, the directory

tapemark

EOF1

EOF2

tapemark

HDR1

HDR2

tapemark

first dumped file, beginning with one record which is the disk header and the file's TITLE, followed by the actual file data in 901-word records. The last record may be less than 901 words, i.e., no padding for short records.

tapemark

EOF1

EOF2

tapemark

.
.
.

more dumped files

.
.
.

EOV1 (if go to another reel)

tapemark

tapemark

The first word in all records is a transaction number. In the tape directories, the number in the first record is minus one (-1). In each succeeding record, the number is decremented by 1. The first record in each of the dumped files (the header and file name) has a transaction number of 0. In each succeeding record, the number is incremented by 1.

The directory consists of one or more records, each 901 words long consisting of a 1-word transaction count and one word of link information, followed by up to 899 words of file names in standard form. All records are completely packed; a file name may be split across directory records. The link word is used only for multireel library dumps and is described later.

The external file name is described as follows (STANDARDFORM):

Char. 1 Total number of characters in the whole string (self-inclusive).

Char. 2 Security Byte
0 - For MCP use.
1 - User file.
2 - System file.
3 - Usercode specified.

Char. 3 Number of identifiers in external file name.

Char. 4 Identifiers, each preceded by one character giving the length of that identifier (not self-inclusive). For example:

"SYMBOL/MCP"

becomes: 48 "0E010206" 8 "SYMBOL" 48 "03" 8 "MCP", where "48" means to construct the following string from 4-bit input, but treat the results as 8-bit.

NOTE

The end of the directory is flagged by a name character count field of 0.

In order to optimize reloading of specified files, each new reel contains a partial copy of the tape directory, containing those files not dumped on the preceding reels. Thus, a load of a file dumped to reel 3, for example, needs only look at the third reel, since it can tell from that directory where the desired file is.

The link word consists of three fields, which are as follows:

CHARCNT 47:16,
SKIP 25:10, and
MOREINFO 12:13.

CHARCNT and SKIP are needed because the directories of the succeeding reels are simply copied from the pertinent records of the master directory. That means that the first record in any but the first directory will, in general, begin with some names not pertinent to this reel. CHARCNT tells how many characters to jump over (in general, the user will be in the middle of a standard form external file name) to get to the first complete file name. SKIP tells how many complete file names to skip over in order to get to the first complete file dumped to this reel.

MOREINFO is the relative number of the file which immediately precedes the first complete standard form external file name in this directory record. It is used to determine which directory records must be rewritten at reel switch time. The following is an example of a multireel dump:

Suppose

a. n files were dumped.

b. Two reels were used.

c. The k-1st file was being written at the time of reel change.

d. Two directory records were used.

e. File k is the third complete name in the second directory, but there are six characters ending a previous standard form before the first complete name.

Reel 1

1st directory record Names: 1 through start of k-3

2nd directory record Names: end of k-3 through n

file 1

.

.

.

file k-1 (start)

Reel 2

file k-2 (end)

2nd directory record (Same as reel 1)

file k

.

.

.

file n

Note that the directories of any but the first reel will not be the first item on that reel. Reel switch will be sensed while in the middle of a file. The remainder of that file will be written out before the directory is put on a new reel. The split file is logically considered to be on the reel on which it was begun.

The directory on succeeding reels is again called MTA/FILE000. Furthermore, the subsequent files are numbered sequentially, starting where the preceding reel left off, and have in their label records that this is their first reel. Only files actually split over two reels in their label records will have a "reel 2" designation.

APPENDIX C
FORMAT OF PACK LABEL

Position	Length (bytes)	Contents
000-003	004	"VOL1"
004-009	006	Pack Serial Number
010	001	Reserved
011-027	017	Pack Identification
028-029	002	System-Interchange Code
		Native Mode = 67
		Interchange = 00
030	001	Reserved
031-036	006	Reserved
037-050	014	Owner's Identification
051-078	028	Reserved
079	001	Reserved
080-083	004	"VOL2"
084-088	005	Initialization Date
089-094	006	Initializing System
		Native = 67MC <mark digit> <level no>
		Interchange = <system series>
		<operating system> <version>
095-102	008	Directory Link
		Native Mode Links to Pack Master
		Header, Interchange Links to First
		Directory Block
103-110	008	Master Available Table Link
111-118	008	Available Table Link
		Native Mode Unused

Position	Length (bytes)	Contents
119	001	Integrity Flag
120-125	006	Reserved
126-131	006	Reserved
132-179	048	Reserved
180-359	180	Reserved for Security Information

APPENDIX D
FORMAT OF INTERCHANGE DIRECTORY RECORD

Position	Length (digits)	Contents
000-007	008	Forward Link
008-015	008	Backward Link
016-023	008	Address of this Sector
024	001	Marker
025-027	003	Reserved
028	001	Hexadecimal "F" (1111)
029	001	Reserved
	---DIRECTORY ENTRIES---	
030-037	008	Address of Header
038-041	004	Length of Header
042-057	016	Name
058	001	Validity Flag
059	001	Reserved
060-359	300	10 or More Digit Entries

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

APPENDIX E
FORMAT OF INTERCHANGE FILE HEADER

Position	Length	Contents
000-005	006	Core Address
006-013	008	Self Pointer
014-017	004	Header Size
018-019	002	File Type
020-031	012	Reserved
032-037	006	Record Size
038-040	003	Records-Block
041-049	009	Block Size
050-055	006	Blocks-Area
056-061	006	Sectors-Area
062-065	004	Areas Requested
066-069	004	Area Counter
070-079	010	End-of-File Pointer
080-081	002	Record Format
082-089	008	Link to User Header
090-093	004	Reserved
094-098	005	Creation Date

Position	Length	Contents
099-103	005	Last Access Date
104-108	005	Save Factor (Purge Date)
109-128	020	Reserved
129-161	033	Reserved for Security Information
162-163	002	Open Type and Permanent Flag
164-165	002	Number of Users
166-167	002	Number Open Output
168-199	032	Reserved
200-207	008	First Area Link
208-359	152	19 More Area Links

APPENDIX F
NAME CONVENTIONS FOR DISK PACKS

FAMILYNAME	FILENAME	Pack Name	Resulting TITLE
PN	A	IC	A ON PN
		NP	A ON PN
PN	A/B	IC	B ON PN
		NP	A/B ON PN
PN	A/B/C. . ./N	IC	N ON PN
	(Max. 14 Names)	NP	A/B/C... ON PN
(none)	A	IC	A ON A
		NP	A ON DISK
(none)	A/B	IC	B ON A
		NP	A/B ON DISK
(none)	A/B/C.../N	IC	N ON A
		NP	A/B/C.../N ON DISK

IC = Interchange Pack

NP = Named Pack

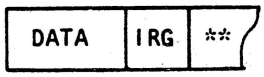
TEXT DELETED

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES.

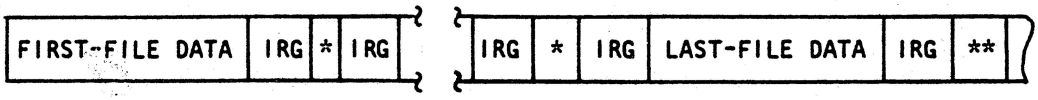
APPENDIX G
STANDARD TAPE LABEL FORMATS

Format of Unlabeled Tapes

SINGLE FILE REEL



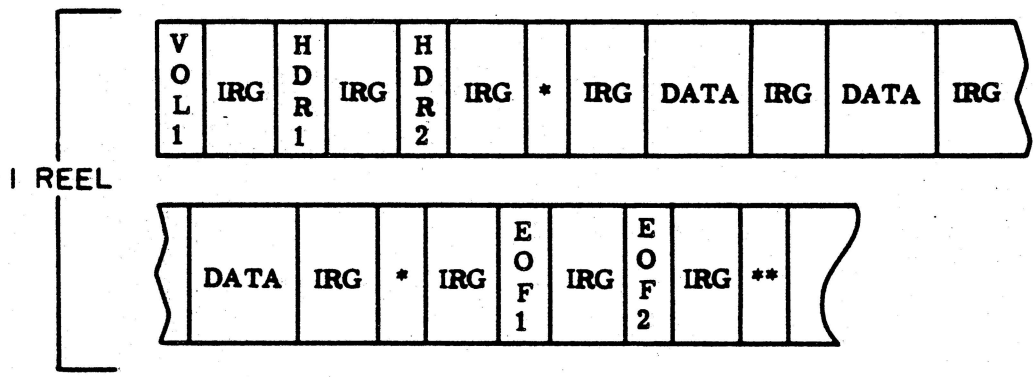
MULTIFILE REEL



Note: Symbol * denotes a tape mark. A double tape mark consists of two tape marks separated by an inter-record gap.

Format of Labeled Tapes

SINGLE FILE, SINGLE REEL



MV1905

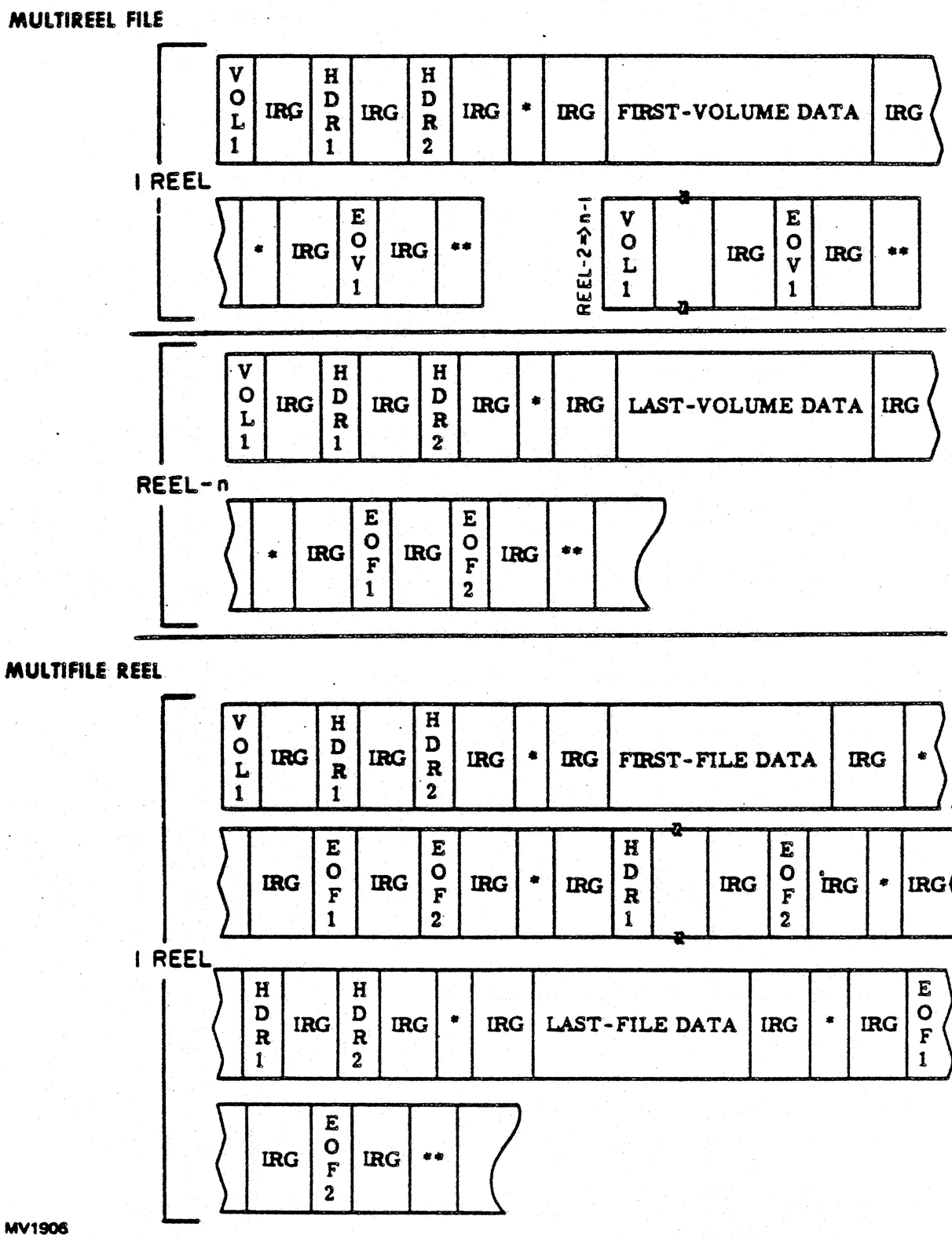
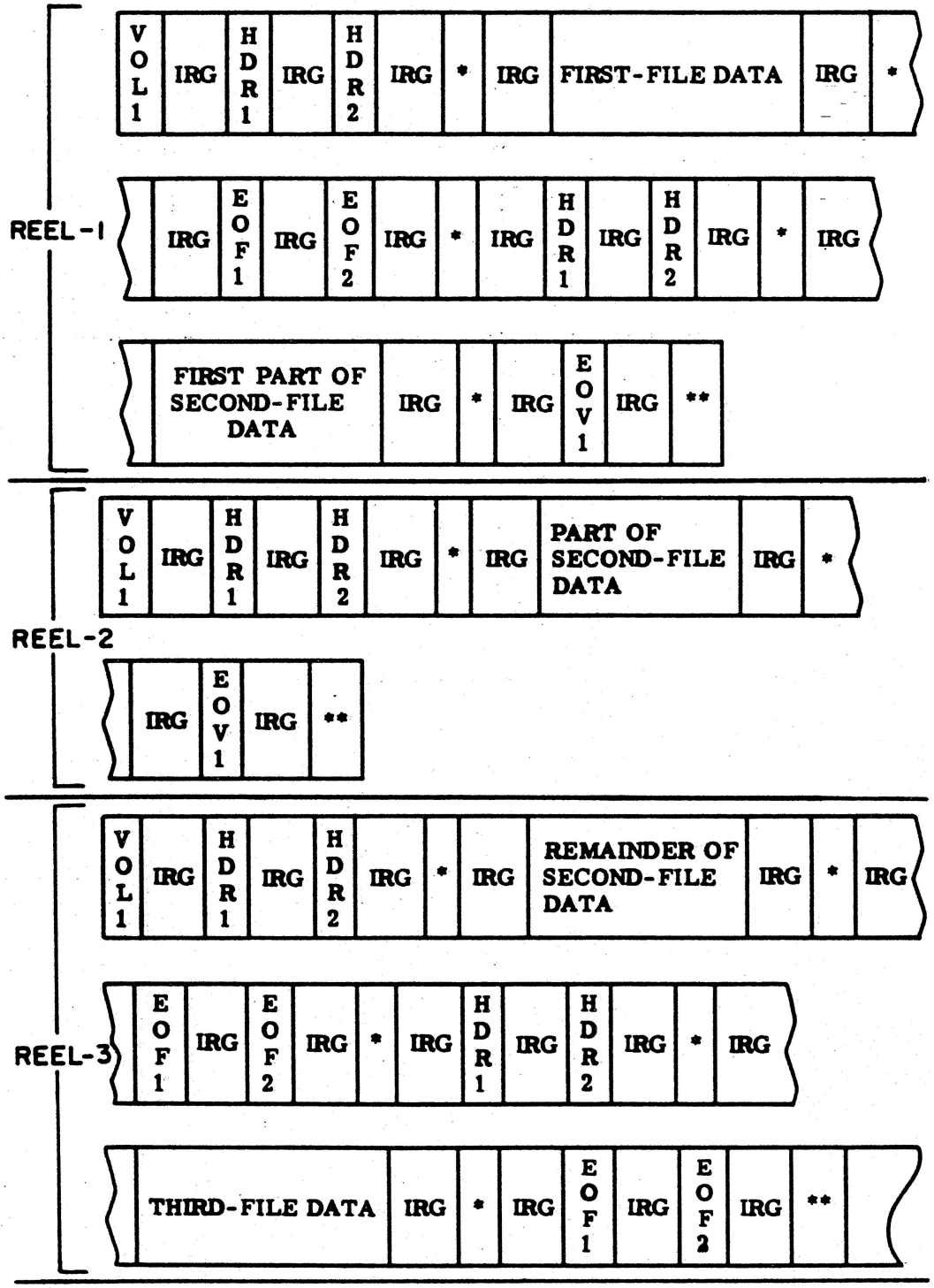


Figure G-2. Multireel File, Multifile Reel

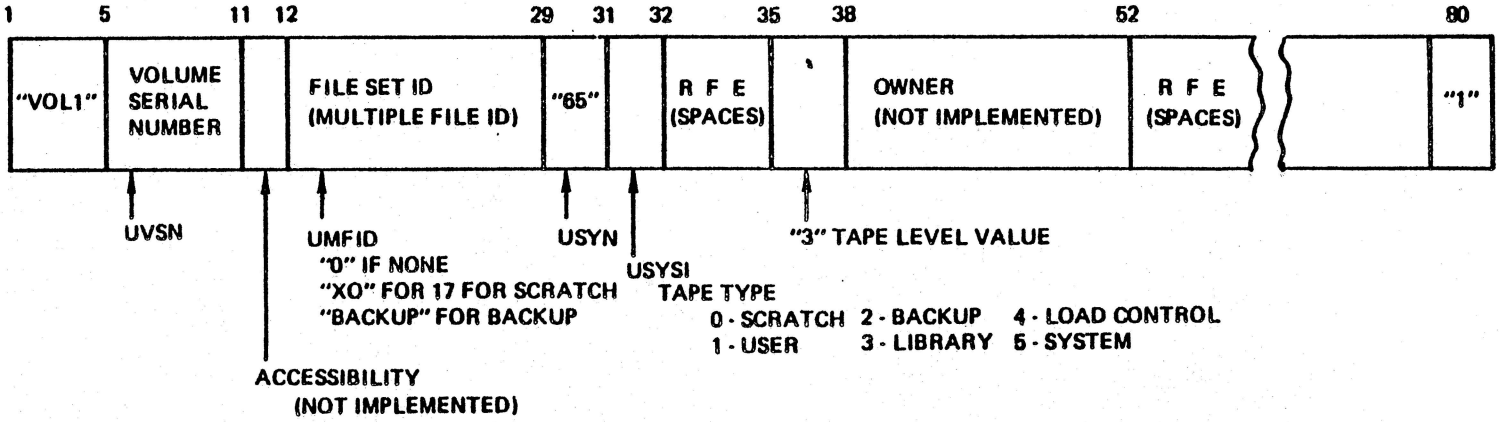
MULTIFILE, MULTIREEL



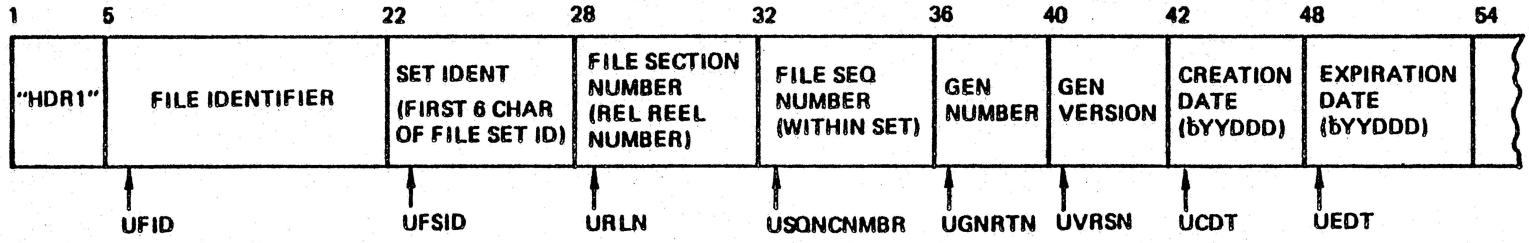
Notes: 1. Symbol * denotes a tape mark.
2. User's header labels may appear immediately after HDR2, and user's trailer labels may appear after either EOF2 or EOV1.

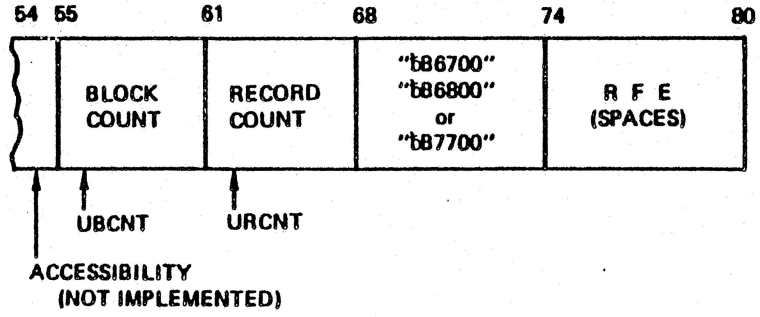
MV1907

VOLUME HEADER



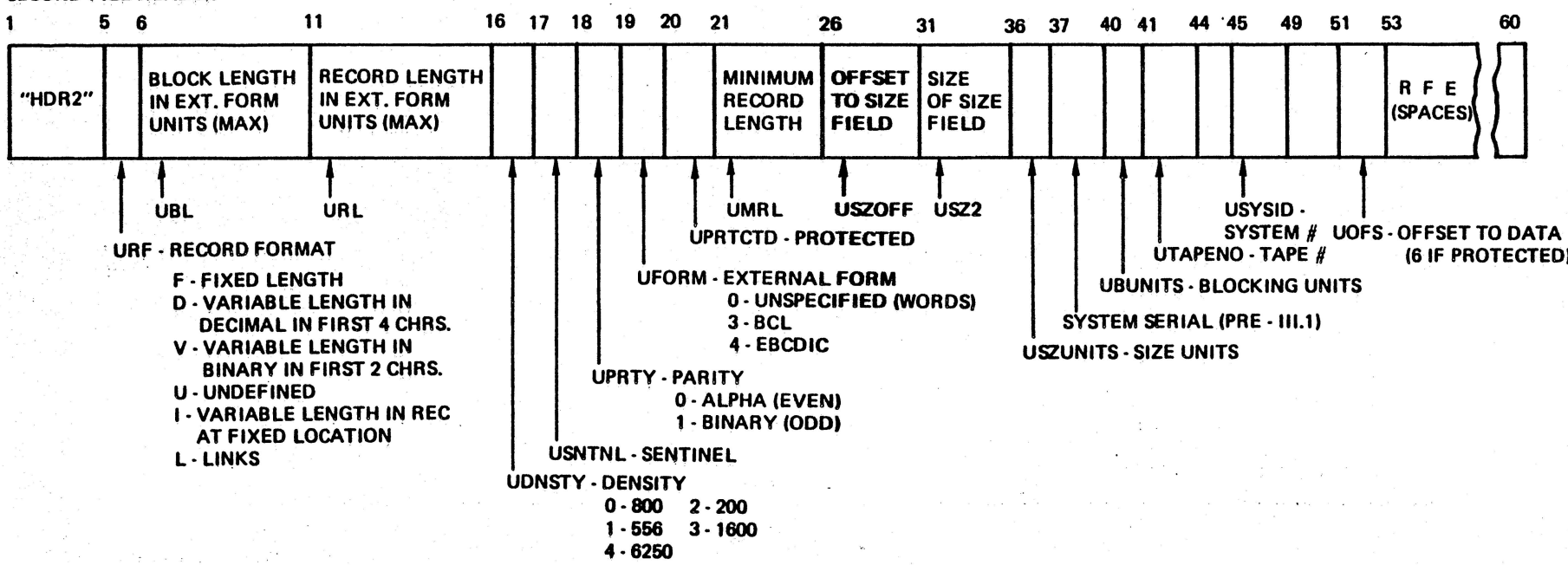
FIRST FILE HEADER





MV1908(SHEET 1 OF 2)

SECOND FILE HEADER



FIRST END-OF-FILE LABEL

SAME AS FIRST FILE HEADER, EXCEPT FOR FIRST FOUR CHARACTERS: "EOF1."

SECOND END-OF-FILE LABEL

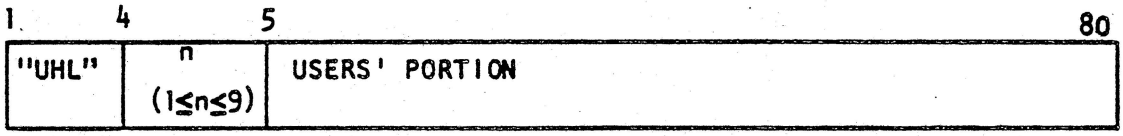
SAME AS SECOND FILE HEADER, EXCEPT FOR FIRST FOUR CHARACTERS: "EOF2."

END-OF-VOLUME LABEL

SAME AS FIRST END-OF-FILE LABEL, EXCEPT FOR FIRST FOUR CHARACTERS: "EOV1."

MV1908(SHEET 2 OF 2)

USERS' HEADER LABEL



USERS' TRAILER LABEL

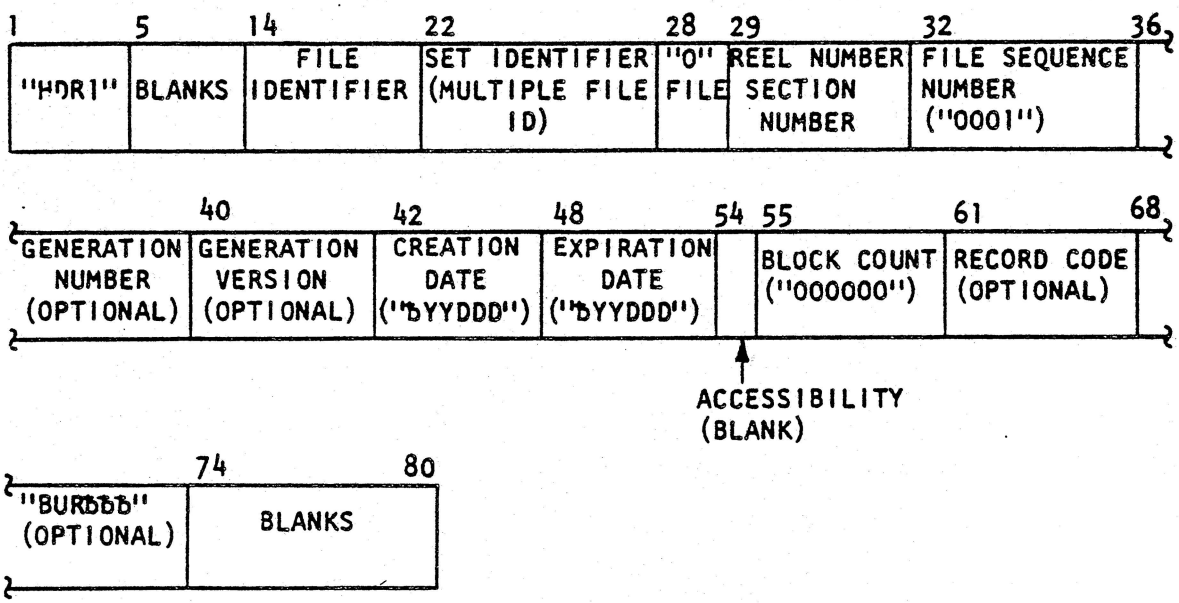
Same as users' header label, except for first three characters: "UTL."

B 3500 USASI Tape Label

VOLUME HEADER



FIRST FILE HEADER



MV1909

B5500 Tape Label

Word	Character (Word)	Character (Record)	Field Description
1	1-8	1-8	Must contain " LABEL ".
2	1	9	Must be 0.
2	2-8	10-16	Multifile id.
3	1	17	Must be 0.
3	2-8	18-24	File id.
4	1-3	25-27	Reel number (within file).
4	4-8	28-32	Date written (creation date).
5	1-2	33-34	Cycle number (to distinguish between identical runs on the same day).
5	3-7	35-39	Purge date (date this file can be destroyed).
5	8	40	Sentinel (1 = end-of-reel, 0 = end-of-file).
6	1-5	41-45	Block count.
6-7	6-8	1-4	Record count.
7	5	53	Memory dump key (1 = memory dump follows label).
7-8	6-8 1-2	54-58	Physical tape number.

The remainder of the information contained in the label record varies for ALGOL and COBOL files as follows:

ALGOL FILES

Word	Character (Word)	Character (Record)	Field Description
8	3	59	Blocking indicator (3 = blocked, 0 = not blocked).
8	4-8	60-64	Buffer size (number of words).
9	1-5	65-69	Maximum record size (number of words).
9	6-8	70-72	Zeros.

COBOL FILES

Word	Character (Word)	Character (Record)	Field Description
8	3-8	59-64	Reserved for file-control routine --- not currently being used.
9-?	1-?	65-?	Users' portion (may be of any format desired by user and may be up to 8,120 characters in length for tape files, up to 16 characters in length for card files, and up to 56 characters in length for printer files.

INDEX

ACCESSCODE 3– 15, 4– 26, 4– 29
ALGOLCODE 3– 30
ALGOLSYMBOL 3– 30
APL file attribute 3– 4, 3– 11
APLDATA 3– 30
APLWORKSOURCE 3– 30
ARCHIVELOG 3– 30
AREAALLOCATED file attribute 3– 4, 3– 11, 4– 10
AREACLASS file attribute 3– 4, 3– 11, 3– 12, 4– 10, 4– 18, 4– 23
AREALENGTH file attribute . . . 2– 5, 3– 4, 3– 12, 3– 13, 3– 22, 3– 36, 3– 70, 4– 20
AREAS file attribute 2– 5, 3– 2, 3– 4, 3– 13
AREASIZE file attribute . . 2– 5, 3– 4, 3– 12, 3– 13, 3– 47, 4– 2, 4– 3, 4– 18
ASCII 3– 26, 3– 39, 3– 60
ASCII72 . . . . . 3– 68
ASSIGNTIME file attribute 2– 3, 3– 4, 3– 13
ASYNC 3– 65
ATTERR file attribute 1– 11, 3– 4, 3– 14
ATTVALUE file attribute 1– 11, 3– 4, 3– 14
ATTYPE file attribute 3– 4, 3– 14
AVAILABLE file attribute 1– 8, 1– 11, 3– 1, 3– 4, 3– 14, 3– 15, 3– 26, 3– 47, 3– 48, 3– 50, 3– 53, 3– 59, 4– 1, 4– 4
AVAILABLE open option 4– 6
AWAITINGHOST 3– 32

BACKUPDISK 3– 30
BACKUPKIND file attribute 2– 4, 2– 7, 3– 4, 3– 15, 3– 16
BACKUPPRINTER 3– 30
BACKUPPUNCH 3– 30
BASICCODE 3– 30
BASICSYMBOL 3– 30
BCL 2– 8, 3– 26, 3– 39, 3– 44, 3– 60, 4– 7, 4– 8
BCL64 . . . . . 3– 68
BDDATA 3– 30
BINARY 3– 26, 3– 44, 3– 69, 4– 8
BINDERSYMBOL 3– 30
BLACKBAND 3– 54
BLOCK file attribute 3– 4, 3– 16
BLOCKED 3– 32
BLOCKSIZE file attribute 1– 11, 2– 1, 2– 5, 2– 6, 3– 4, 3– 10, 3– 12, 3– 14, 3– 16, 3– 17, 3– 21, 3– 22, 3– 23, 3– 34, 3– 35, 3– 36, 3– 44, 3– 53, 3– 70, 4– 7
BLOCKSTRUCTURE file attribute 1– 11, 2– 2, 2– 12, 2– 13, 3– 4, 3– 13, 3– 17, 3– 18, 3– 22, 3– 23, 3– 24, 3– 33, 3– 44, 3– 45, 3– 63, 3– 69, 4– 8
BOUNDCODE 3– 30
BUFFERS file attribute 3– 4, 3– 18

CARDPUNCH 3– 44
CARDREADER 3– 44
CARRIAGECONTROL file attribute 2– 7, 3– 4, 3– 18
CATALOG 3– 30
CDATA 3– 30
CENSUS file attribute 2– 3, 2– 12, 3– 4, 3– 20, 3– 38
CHANGEDSUBFILE file attribute 2– 12, 3– 4, 3– 20
CHANGEEVENT file attribute 2– 12, 3– 4, 3– 20
CHARACTERS 2– 6, 3– 16, 3– 19, 3– 45, 3– 70
CHECKPOINTFILE 3– 30
CLASSA 3– 57
CLASSB 3– 57
CLOSE operation 3– 64, 4– 1, 4– 6, 4– 30
CLOSED 3– 15, 3– 32, 3– 46, 4– 5, 4– 6

CLOSEPENDING 3- 32
COBOLCODE 3- 30
COBOLSYMBOL 3- 30
CODEFILE 3- 29, 3- 30
COMPILERCODEFILE 3- 29, 3- 30, 3- 73
CONTROLLED 1- 12, 3- 56, 3- 57, 4- 29, 4- 30
CONTROLMESSAGE file attribute 3- 4, 3- 20
COPIES file attribute 3- 4, 3- 20, 3- 25, 4- 10
COPYNAME file attribute 3- 4, 3- 21
CP 3- 40
CPJOBFILE 3- 30
CREATIONDATE file attribute 3- 4, 3- 5, 3- 21, 3- 23
CRUNCHED file attribute 3- 5, 3- 21, 3- 42
CSEQDATA 3- 30
CURRENTBLOCK file attribute 3- 5, 3- 22, 3- 36, 3- 70, 4- 7
CURRENTRECORD file attribute . . . 2- 12, 3- 5, 3- 18, 3- 22, 3- 33, 3- 36, 4- 7
CYCLE file attribute . . . 1- 10, 2- 7, 3- 5, 3- 22, 3- 31, 3- 37, 3- 51, 3- 58, 3- 71, 4- 4
CYLINDERMODE file attribute 2- 6, 3- 5, 3- 22

DASDLSYMBOL 3- 30
DATA 3- 30, 3- 42, 4- 16
Data Comm . . . 2- 2, 2- 3, 2- 4, 3- 2, 3- 4, 3- 9, 3- 13, 3- 16, 3- 20, 3- 24, 3- 25, 3- 26, 3- 31, 3- 33, 3- 41, 3- 42, 3- 46, 3- 49, 3- 50, 3- 56, 3- 62, 3- 63, 3- 64, 3- 65, 3- 66, 3- 69, 3- 71, 3- 73, 3- 75, 4- 26, 4- 27
DATALOST 3- 65
DATE file attribute 3- 5, 3- 10, 3- 23
DBDATA 3- 29, 3- 30, 3- 42
DBRESTARTSET 3- 30, 3- 42
DC 3- 40
DCALGOL diskheader attribute 3- 10, 4- 1
DCALGOLCODE 3- 30
DCALGOLSYMBOL 3- 30
Data Communication Processor (DCP) 3- 69
DCPCODE 3- 30
DCPSYMBOL 3- 30
DEACTIVATED 3- 33
DEACTIVATIONPENDING 3- 32
DEFAULTTRANS 3- 68, 4- 8
DENIED 3- 33
DENIEDILLEGALUSE 3- 33
DENSITY file attribute 2- 7, 3- 5, 3- 23
DEPENDENTSPECS file attribute . . 3- 1, 3- 5, 3- 23, 3- 24, 3- 33, 3- 53, 4- 7
DIRECTION file attribute 3- 5, 3- 24
directory . . . . . 1- 3, 1- 8, 1- 10, 2- 5, 2- 6, 3- 21, 3- 47, 3- 51, 3- 58, 3- 71, 4- 16, 4- 20, 4- 27, B- 1, B- 2, B- 3, B- 4
DIRECTORY 3- 30
DISCONNECTED 3- 65
DISK 2- 4, 3- 2, 3- 27, 3- 28, 3- 30, 3- 40, 3- 41, 3- 44, 3- 47, 3- 51, 3- 67, 4- 2, 4- 3, 4- 28, 4- 29
DISKPACK 3- 40
DISPOSITION file attribute 2- 3, 2- 4, 3- 5, 3- 24, 3- 50
DMALGOLCODE 3- 30
DMALGOLSYMBOL 3- 30
DONTCARE 3- 15, 3- 16, 3- 40
DONTWAIT 2- 12, 2- 13, 3- 62, 3- 65, 4- 6
DUPLICATED file attribute . . . 3- 5, 3- 10, 3- 11, 3- 20, 3- 21, 3- 25, 3- 42, 3- 53, 4- 10
duplicated files 2- 5, 3- 21, 3- 25, 3- 62, 4- 10, 4- 12

EBCDIC 3- 26, 3- 39, 3- 60
EBCDIC18 . . . . . 3- 68
EBCDIC48 . . . . . 3- 68
EBCDIC72 . . . . . 3- 68

EBCDIC96 . . . . . 3- 68
ENABLEINPUT file attribute 3- 5, 3- 25
EOF file attribute 3- 5, 3- 25, 3- 41, 3- 62, 3- 63, 3- 74, 4- 12, 4- 20
EOFBITS diskheader attribute 3- 10
EOFSEGMENT diskheader attribute 3- 10
error detection 4- 19, 4- 20, 4- 21, 4- 22, 4- 23, 4- 24
ERRORTYPE file attribute 3- 5, 3- 25, 3- 40, 4- 12, 4- 20, 4- 21
ESPOLCODE 3- 30
ESPOLSYMBOL 3- 30
EUNUMBER diskheader attribute 3- 10
EXCLUSIVE file attribute 3- 5, 3- 26, 3- 50, 3- 53
EXTENDRECOVERY 3- 54
EXTERNAL 2- 2, 2- 13, 3- 17, 3- 18, 3- 22
EXTMODE file attribute . . . 1- 12, 2- 1, 2- 2, 2- 6, 2- 8, 3- 5, 3- 26, 3- 34, 3- 39, 3- 44, 3- 69, 4- 7, 4- 8

FAMILY file attribute 3- 1, 3- 5, 3- 27
FAMILYINDEX file attribute . . . 2- 6, 3- 5, 3- 11, 3- 12, 3- 27, 3- 53, 3- 60, 4- 10, 4- 22, 4- 23
FAMILYNAME file attribute . . . . . 1- 10, 2- 5, 3- 5, 3- 7, 3- 27, 3- 28, 3- 30, 3- 31, 3- 41, 3- 49, 3- 53, 3- 66, 3- 67, 4- 27, 4- 28, 4- 29, F- 1
FAMILYSIZE file attribute 3- 5, 3- 28
FAST 3- 61
FCM file attribute 3- 5, 3- 28, 3- 38, 3- 55
File Information Block (FIB) 2- 3
FILEKIND file attribute 1- 11, 3- 5, 3- 10, 3- 29, 3- 30, 3- 73
FILENAME file attribute 1- 10, 3- 5, 3- 30, 3- 31, 3- 66
file names . . . . . 1- 2, 1- 8, 1- 9, 3- 21, 3- 67, A- 1, B- 2
FILEORGANIZATION file attribute 3- 5, 3- 31, 3- 62
file reconstruction 4- 10, 4- 17, 4- 19, 4- 22, 4- 23, 4- 24
file restoration 4- 10, 4- 14
FILESECTION file attribute 2- 7, 3- 5, 3- 7, 3- 31, 3- 52, 4- 4
FILESTATE file attribute 2- 12, 3- 5, 3- 20, 3- 31, 3- 32, 4- 5, 4- 6
FILETYPE file attribute 1- 11, 3- 5, 3- 10, 3- 13, 3- 17, 3- 18, 3- 22, 3- 23, 3- 24, 3- 33, 3- 34, 3- 35, 3- 44, 3- 45, 3- 53, 3- 60, 3- 63, 3- 69, 4- 7, 4- 8
FILEUSE file attribute 3- 5, 3- 35
FIRMWARE 3- 30
FIXED 2- 13, 3- 13, 3- 17, 3- 18, 3- 44, 3- 45, 3- 63
FLEXIBLE file attribute 2- 5, 3- 5, 3- 13, 3- 35
FLOWSTOPCOMPLETE 3- 54
FORCESOFT 3- 69
FORMID file attribute 2- 6, 3- 6, 3- 36, 4- 4, 4- 7
FORMMESSAGE file attribute 3- 6, 3- 36
FORTRANCODE 3- 30
FORTRANSYMBOL 3- 30
FORWARD 3- 24
FRAMESIZE file attribute 2- 2, 2- 12, 3- 6, 3- 12, 3- 16, 3- 19, 3- 23, 3- 36, 3- 39, 3- 44, 3- 45, 3- 63, 3- 70, 4- 7
FULLTRANS 3- 68, 4- 9

GENERATION file attribute 3- 6, 3- 37
GUARDED 1- 12, 3- 56, 3- 57, 4- 29, 4- 30
GUARDFILE 3- 30, 3- 56, 4- 29

HEX 3- 26, 3- 36, 3- 39, 3- 60, 4- 8
HOSTNAME file attribute 3- 6, 3- 32, 3- 37

IAD file attribute 3- 6, 3- 10, 3- 36, 3- 37, 4- 11
IMAGEFILEID file attribute 3- 6, 3- 37
IMAGEFILEMAP file attribute 3- 6, 3- 37, 3- 38
IN 3- 46, 3- 57
INDEXED 3- 31
INFOFILE 3- 30

003

INITCOMPLETE 3- 54
INPUTEVENT file attribute 2- 12, 3- 6, 3- 38
INPUTTABLE file attribute 3- 6, 3- 38, 3- 69
INTERCHANGE file attribute . . . . . 2- 6, 3- 6, 3- 38, D- 1, E- 1
INTMODE file attribute 1- 11, 1- 12, 2- 8, 3- 6, 3- 16, 3- 17, 3- 19,
3- 26, 3- 33, 3- 34, 3- 36, 3- 39, 3- 44, 3- 45, 3- 53, 3- 63, 3- 69,
3- 70, 4- 7, 4- 8
INTNAME file attribute 1- 10, 2- 12, 3- 1, 3- 6, 3- 30, 3- 39, 3- 66
INTRINSICFILE 3- 30
IO 3- 46, 3- 57
IOADDRESS buffer attribute 3- 9, 3- 73
IOCANCEL buffer attribute 3- 9, 3- 73
IOCHARACTERS buffer attribute 3- 9, 3- 73
IOCLOCKS file attribute 3- 6, 3- 40
IOCOMPLETE buffer attribute 3- 9, 3- 74
IOCW buffer attribute 2- 8, 3- 9, 3- 74
IOEOF buffer attribute 3- 9, 3- 25, 3- 74
IOERRORTYPE buffer attribute 3- 9, 3- 47, 3- 74, 4- 21, 4- 22
IOINERROR file attribute 3- 6, 3- 25, 3- 40, 3- 52
IOMASK buffer attribute 3- 9, 3- 75
IOPENDING buffer attribute 3- 9, 3- 73, 3- 75
IORECORDNUM buffer attribute 3- 9, 3- 42, 3- 75, 4- 21, 4- 22
IORESULT buffer attribute 3- 9, 3- 47, 3- 73, 3- 75
IOTIME buffer attribute 3- 9, 3- 76
IOWORDS buffer attribute 3- 10, 3- 76

JAM 3- 54
JOBCODE 3- 30
JOBDESCFILE 3- 30
JOBSYMBOL 3- 30
JOVIALCODE 3- 30
JOVIALSYMBOL 3- 30

KIND file attribute 1- 10, 1- 11, 2- 1, 2- 2, 2- 4, 2- 6, 2- 7, 2- 12,
3- 2, 3- 6, 3- 13, 3- 15, 3- 16, 3- 18, 3- 26, 3- 27, 3- 28, 3- 30, 3- 31,
3- 39, 3- 40, 3- 41, 3- 44, 3- 46, 3- 47, 3- 48, 3- 50, 3- 51, 3- 55,
3- 58, 3- 66, 3- 67, 3- 70, 4- 1, 4- 2, 4- 3, 4- 4, 4- 5, 4- 7, 4- 8,
4- 22, 4- 28 4- 29

LABEL file attribute 1- 11, 2- 1, 2- 2, 2- 7, 3- 6, 3- 29, 3- 41, 4- 7,
C- 1, G- 1, G- 7
LABELTYPE file attribute 3- 6, 3- 41
LASTACCESSDATE diskheader attribute 3- 10
LASTRECORD file attribute 3- 6, 3- 41, 3- 42
LASTSTATION file attribute 3- 6, 3- 42
LASTSUBFILE file attribute 2- 3, 2- 4, 2- 12, 3- 6, 3- 42, 3- 64, 3- 75
LATEREAD 3- 54
LATESELECT 3- 54
LCOBOLSL3CODE 3- 30
LCOBOLSL5CODE 3- 30
LCOBOLSYMBOL 3- 30
LIBRARYCODE 3- 30
library maintenance 3- 12, 3- 71, 4- 19, 4- 31
library tapes . . . . . B- 1
LINENUM file attribute 2- 7, 3- 6, 3- 42, 3- 43, 3- 49, 3- 50

Master Control Program (MCP) . . . 1- 1, 1- 2, 1- 3, 1- 5, 3- 59, 3- 71, 4- 10,
4- 11, 4- 14, 4- 16, 4- 17, 4- 19, 4- 31, B- 2
MAXCENSUS file attribute 2- 12, 3- 6, 3- 43, 3- 44
MAXRECSIZE file attribute 1- 11, 2- 2, 2- 5, 2- 6, 2- 12, 2- 13, 3- 6,
3- 10, 3- 12, 3- 16, 3- 17, 3- 18, 3- 22, 3- 23, 3- 34, 3- 35, 3- 39,
3- 44, 3- 45, 3- 63, 3- 70, 4- 7, 4- 20
MAXSUBFILES file attribute 2- 12, 3- 6, 3- 15, 3- 45, 3- 65, 4- 5, 4- 6,
4- 7

Message Control System (MCP) . . . 2- 2, 2- 3, 2- 4, 3- 64, 3- 65, 4- 26, 4- 27, 4- 31
MINRECSIZE file attribute . . . 1- 11, 3- 6, 3- 10, 3- 17, 3- 18, 3- 34, 3- 35, 3- 36, 3- 44, 3- 45, 3- 53, 3- 70, 4- 7
MODE diskheader attribute . . . 3- 10
MULTIFLOWSTOP . . . 3- 54
MYNAME file attribute . . . 2- 12, 3- 6, 3- 45, 4- 5
MYUSE file attribute 2- 2, 2- 3, 2- 4, 3- 6, 3- 13, 3- 35, 3- 46, 3- 47, 3- 70, 3- 71, 4- 1, 4- 2, 4- 3, 4- 4

NDLCODE . . . 3- 30
NDLSYMBOL . . . 3- 30
NEWFILE file attribute . . . 3- 7, 3- 13, 3- 46, 3- 47, 4- 1, 4- 2, 4- 3
new file vs. permanent fle . . . 4- 1
NEWPCODE . . . 3- 30
NEWPSYMBOL . . . 3- 30
Network Information File (NIF) . . . 2- 2, 2- 3, 3- 50, 3- 56, 3- 63, 3- 64, 3- 67, 3- 69, 3- 72
NOBUFFER . . . 3- 65
NOERROR . . . 3- 25, 3- 65
NOFILEFOUND . . . 3- 65, 4- 6
NOID . . . 3- 67, 3- 68
NONE . . . 3- 65
NONRESTRICTED . . . 3- 31
NONSTANDARD . . . 3- 50
NORESOURCEWAIT file attribute . . . 3- 47
NORVRSPAPERTAPE . . . 2- 8, 3- 24
NOSOFT . . . 3- 69
NOTRANS . . . 3- 69
NOTREADY . . . 3- 54
Network Support Processor (NSP) . . . 2- 2
NULLFILE . . . 3- 30

OFFER open option . . . 4- 6
OFFERED . . . 3- 32, 4- 5, 4- 6
OMITTED . . . 1- 11, 3- 41, 4- 7
OMITTEDEOF . . . 1- 11, 3- 41, 4- 7
OPEN file attribute . . . 2- 4, 3- 7, 3- 12, 3- 26, 3- 39, 3- 48, 4- 30
OPEN operation . . . 3- 15, 4- 1, 4- 5, 4- 6
OPENED . . . 3- 32
OPTIONAL file attribute . . . 3- 1, 3- 7, 3- 48
OUT . . . 3- 46, 3- 57
OUTPUTEVENT file attribute . . . 2- 12, 3- 7, 3- 48
OUTPUTTABLE file attribute . . . 1- 12, 3- 7, 3- 48, 3- 49, 3- 69

PACK . . . . 2- 4, 3- 2, 3- 16, 3- 27, 3- 28, 3- 30, 3- 40, 3- 41, 3- 44, 3- 47, 3- 51, 3- 67, 4- 28, 4- 29, C- 1
pack label formats . . . C- 1
PACKNAME file attribute . . . 3- 7, 3- 49
PAGE file attribute . . . . 2- 7, 3- 1, 3- 7, 3- 43, 3- 49, 3- 50, A- 1, B- 1, E- 1, G- 1
PAGESIZE file attribute . . . 2- 7, 3- 7, 3- 42, 3- 43, 3- 49, 3- 50
PAPER . . . 3- 40
PAPERPUNCH . . . 2- 8, 3- 40, 3- 44
PAPERREADER . . . 2- 8, 3- 40, 3- 44
PARITY file attribute . . . 3- 7, 3- 50, 4- 14, 4- 17
PASCALCODE . . . 3- 30
PASCALSYMBOL . . . 3- 30
password . . . 4- 25, 4- 26, 4- 27, 4- 31
PETAPE . . . 3- 23, 3- 40
PLICODE . . . 3- 30
PLISYMBOL . . . 3- 30
POPULATION file attribute . . . 2- 3, 3- 7, 3- 42, 3- 50, 3- 64
PORT . . . 2- 12, 3- 18, 3- 40, 3- 62, 4- 5

port subfile assignment 4- 1
POSTPONED 3- 33
PRESENT file attribute 1- 11, 3- 7, 3- 26, 3- 48, 3- 50, 3- 51, 3- 53, 3- 59
PRINTER 2- 6, 3- 40, 3- 44, 3- 47, 4- 2, 4- 3, 4- 4, 4- 7
PRIVATE 1- 12, 3- 57, 4- 5, 4- 29, 4- 30
PROTECTED 3- 42, 3- 51
PROTECTION file attribute 2- 5, 3- 7, 3- 22, 3- 51, 3- 58
PTP 3- 40
PTR 3- 40
PUBLIC 1- 12, 3- 57, 4- 5, 4- 29, 4- 30
PUNCH 2- 1, 3- 26, 3- 40, 3- 45, 4- 2, 4- 3, 4- 4, 4- 7

READ operation 3- 35, 3- 46, 3- 52, 3- 73, 4- 10, 4- 11, 4- 12, 4- 13, 4- 14, 4- 17, 4- 19, 4- 20, 4- 21, 4- 22, 4- 30, 4- 31
READCHECKFAILURE 3- 25
READER 2- 1, 3- 26, 3- 40, 3- 45, 3- 47, 4- 2, 4- 3
READERSORTER 3- 40, 3- 62, 3- 63
READPARITYERROR 3- 25
RECEPTIONS file attribute 3- 7, 3- 51
RECONSTRUCTIONFILE 3- 30
RECORD file attribute . . . . . 1- 11, 3- 7, 3- 52, 4- 20, 4- 21, D- 1
RECORDINERROR file attribute 1- 11, 3- 7
recovery 4- 14, 4- 16, 4- 17, 4- 18, 4- 19, 4- 20, 4- 21, 4- 22, 4- 23, 4- 24
RECOVERYFILE 3- 30
RECOVERYFILEID file attribute 3- 7, 3- 52
RECOVERYINDEX file attribute 3- 7, 3- 52
REEL file attribute 3- 7, 3- 52, 3- 66
RELATIVE 3- 31, 3- 62
REMAPREQUEST 3- 54
REMOTE 2- 2, 3- 16, 3- 26, 3- 40, 3- 44, 3- 46, 3- 50
REMOTEAUDIT 3- 30
REMOTEBACKUP 3- 30
REMOTEID file attribute 3- 7, 3- 52
RESIDENT file attribute 1- 11, 3- 7, 3- 26, 3- 51, 3- 53, 3- 59
result descriptors . . . 1- 11, 2- 2, 2- 12, 3- 55, 3- 74, 3- 75, 4- 11, 4- 12, 4- 21, 4- 31
REVERSE 2- 7, 3- 24
ROWADDRESS 3- 7, 3- 10, 3- 53, 4- 10
ROWCLASS diskheader attribute 3- 10
ROWS diskheader attribute 3- 10
ROWSINUSE file attribute 3- 7, 3- 53
ROWSIZE diskheader attribute 3- 10
RPGCODE 3- 30
RPGSYMBOL 3- 30
RS 3- 40
RSERROR file attribute 3- 7, 3- 54, 3- 63
RSEXTENDIBLE file attribute 3- 7, 3- 54
Relative Station Number (RSN) 2- 3, 2- 4, 3- 13, 3- 24, 3- 25, 3- 42, 3- 50, 3- 51, 3- 56, 3- 63, 3- 64, 3- 67, 3- 69, 3- 71, 3- 72
RSNETFILE 3- 30
RSPCODE 3- 30
RSPERROR 3- 54
RSRESTART file attribute 3- 7, 3- 54
RSRESULT file attribute 3- 7, 3- 55
RSSORTTABLE 3- 30
RSUNITNO file attribute 3- 7, 3- 55

SANSCODE 3- 30
SANSSYMBOL 3- 30
SAVE 3- 51, 3- 58
SAVEFACTOR file attribute 2- 7, 3- 7, 3- 10, 3- 55, 3- 56
SCHEDULEFILE 3- 30

SCREEN file attribute 2- 3, 3- 8, 3- 56
SCREENSIZE file attribute 3- 56
SECURED 3- 57, 4- 30
security 4- 4, 4- 29
SECURITYGUARD attribute 1- 12, 3- 8, 3- 56, 3- 57, 4- 4, 4- 29, 4- 30
SECURITYTYPE attribute 2- 12, 3- 8, 3- 57, 4- 4, 4- 5, 4- 29, 4- 30
SECURITYUSE attribute 1- 12, 3- 8, 3- 57, 4- 29, 4- 30
SENSITIVEDATA file attribute 3- 8, 3- 58, 3- 71
SEQDATA 3- 30
SERIAL 3- 40
SERIALNO file attribute 2- 7, 3- 8, 3- 51, 3- 58, 3- 59, 4- 3, 4- 4
SFORTRANCODE 3- 30
SFORTRANSYMBOL 3- 30
SHUTTINGDOWN 3- 32
SINGLE 3- 26, 3- 33, 3- 34, 3- 39, 3- 60, 3- 69, 3- 70, 4- 8
SINGLEPACK file attribute 3- 8, 3- 59
SINGLEUNIT file attribute 2- 6, 3- 8, 3- 59
SIZEMODE file attribute 1- 11, 3- 8, 3- 10, 3- 23, 3- 33, 3- 34, 3- 53, 3- 60
SIZEOFFSET file attribute . . . 1- 11, 3- 8, 3- 10, 3- 23, 3- 33, 3- 34, 3- 53, 3- 60
SIZE2 file attribute . . . 1- 11, 3- 8, 3- 10, 3- 23, 3- 33, 3- 34, 3- 53, 3- 60
SLINKEDCODE 3- 30
SLOW 3- 61
SMCPCODE 3- 30
SOFTONLY 3- 68
SORTFILEID file attribute 3- 8, 3- 60
SORTTABLECODE file attribute 3- 8, 3- 61
SPEED file attribute 3- 8, 3- 61
SPO 2- 2, 3- 26, 3- 40, 3- 44
STANDARD . . . . . 2- 1, 3- 18, 3- 29, 3- 41, 3- 50, 4- 21, G- 1
STATE file attribute . . . 1- 11, 2- 12, 2- 13, 3- 2, 3- 8, 3- 22, 3- 33, 3- 40, 3- 43, 3- 61, 3- 66
STATIONCOUNT file attribute . . . 2- 3, 3- 5, 3- 8, 3- 28, 3- 50, 3- 63, 3- 64
STATIONLIST file attribute 2- 2, 2- 3, 3- 5, 3- 8, 3- 27, 3- 63, 3- 64
STATIONNAME file attribute 3- 8, 3- 64
STATIONSDENIED file attribute 3- 8, 3- 50, 3- 64
SUBFILEERROR file attribute 2- 12, 3- 8, 3- 64, 3- 65, 4- 6
SUNOTREADY 3- 25, 4- 21, 4- 22
SYNC 3- 65
SYSTEMDIRECTORY 3- 30
SYSTEMDIRFILE 3- 30

TANKING file attribute 3- 8, 3- 65
TAPE 2- 7, 3- 16, 3- 40
tape label formats . . . . . 2- 5, G- 1
TAPEPE 2- 7, 3- 16, 3- 40, 3- 50
TAPEREELRECORD file attribute 2- 7, 3- 8, 3- 65, 3- 66
TAPE7 . . . . . 2- 7, 3- 16, 3- 40, 3- 50, 4- 8
TAPE9 . . . . . 2- 7, 3- 16, 3- 23, 3- 40, 3- 50
TEMPORARY 3- 51, 3- 71
TIMELIMIT file attribute 3- 8, 3- 66
TITLE file attribute . . . . 1- 10, 2- 2, 2- 3, 2- 6, 2- 7, 2- 12, 3- 8, 3- 30, 3- 31, 3- 37, 3- 48, 3- 52, 3- 56, 3- 57, 3- 58, 3- 61, 3- 63, 3- 66, 3- 67, 3- 69, 4- 4, 4- 5, 4- 28, 4- 31, B- 1, F- 1
TRAINID file attribute 2- 6, 3- 8, 3- 67, 3- 68
TRANSLATE file attribute 1- 12, 3- 8, 3- 68, 3- 69, 4- 8, 4- 9
TRANSLATING file attribute 3- 8, 3- 38, 3- 48, 3- 69
translation 1- 12, 2- 8, 3- 17, 3- 38, 3- 48, 3- 68, 3- 69, 4- 7, 4- 8, 4- 9
TRANSMISSIONNO file attribute 3- 8, 3- 9, 3- 69
TRANSMISSIONO file attribute 3- 8, 3- 69
TRANSMISSIONS file attribute 3- 9, 3- 69

UCRABORT 3- 54
UCRFILE 3- 30
UCRID file attribute 3- 9, 3- 69
UCRSTOPFLOW 3- 54
UNITNO file attribute 3- 9, 3- 55, 3- 70
UNITS file attribute 1- 11, 3- 9, 3- 10, 3- 16, 3- 19, 3- 23, 3- 34, 3- 36, 3- 39, 3- 44, 3- 45, 3- 63, 3- 70, 4- 7
UNREACHABLE 3- 65
UNSPECIFIED 3- 65
UPDATEFILE file attribute 3- 9, 3- 18, 3- 35, 3- 46, 3- 70, 3- 71
USECATALOG file attribute 3- 9, 3- 71
USEDATE file attribute 3- 9, 3- 71
usercode . . . . . 4- 25, 4- 26, 4- 27, 4- 28, 4- 29, 4- 30, 4- 31, A- 1
USER statement 3- 29, 4- 26, 4- 27, 4- 31

VARIABLE 3- 17, 3- 18
VERSION file attribute . . . 1- 10, 2- 7, 3- 9, 3- 22, 3- 37, 3- 58, 3- 71, 4- 4
VERSIONDIRECTORY 3- 30
VFORTRANCODE 3- 30
VFORTRANSYMBOL 3- 30
VLINKEDCODE 3- 30
VMCPCODE 3- 30

WAIT open option 4- 6
Work Flow Language (WFL) 3- 1, 3- 39, 3- 59, 4- 26, 4- 27, 4- 30
WIDTH file attribute 2- 3, 3- 9, 3- 71, 3- 72
WORDS 3- 70
WRITE operation 3- 35, 3- 42, 3- 46, 3- 73, 4- 3, 4- 10, 4- 11, 4- 12, 4- 13, 4- 14, 4- 19, 4- 20, 4- 21, 4- 31

XALGOLCODE 3- 30
XALGOLSYMBOL 3- 30
XDISKFILE 3- 30
XFORTRANCODE 3- 30
XFORTRANSYMBOL 3- 30

YOURNAME file attribute 2- 12, 3- 9, 3- 45, 3- 72, 4- 5
YOURUSERCODE file attribute 2- 12, 3- 9, 3- 72, 4- 5

Documentation Evaluation Form

Title: B 7000/B 6000 Series I/O Subsystem Reference Manual

Form No: 5001779

Date: July 1981

Burroughs Corporation is interested in receiving your comments and suggestions regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

From:

Name

Title

Company

Address

Phone Number Date

Remove form and mail to:

Burroughs Corporation
Documentation Dept., TIO - West
P.O. Box 4040
El Monte, CA 91734