Burroughs

Reference Manual

A Series I/O Subsystem

(Relative to the Mark 3.6 System Software Release)

Distribution Code SE

Priced Item
Printed in U.S.A.
December 1985

1169984

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies 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 to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the Class specified as "2" (System Software), the Type specified as "1" (F.T.R.), and the Product specified as the 7-digit form number of the manual (for example, "1169984"). The FCF should be sent to the following address:

Burroughs Corporation
PA&S/Orange County
19 Morgan
Irvine, CA 92718

CONTENTS

1 INTRODUCTION

1.1 SCOPE OF MANUAL

The I/O Subsystem Reference Manual describes the Input/Output Subsystem of the Burroughs 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 logical input/output operations. In particular, this manual is concerned with structured collections of information called "files" whose structure is determined by means of "file attributes" and which can be manipulated by means of "I/O statements".

This manual serves both as an introduction to system conventions that control input/output and as a reference guide to attributes and formats that are occasionally needed in controlling and tuning I/O operations. Given this two-fold approach, parts of the manual are useful to newcomers approaching the system for the first time; other parts are useful to experienced systems programmers who are tuning the system for some special use.

Burroughs A Series and B 5000/B 6000/B 7000 Series systems are sophisticated, providing a range of capabilities, but sophistication is not demanded of their users. These systems are designed for ease of use, making minimal demands on the user. The sophistication of I/O operations is internal to the logic of the integrated hardware and software that make up Burroughs systems; the user interface itself is quite simple.

It is assumed that the reader has introductory knowledge of at least one programming language, some minimal experience in using that language, and a broad acquaintance with the major issues involved in the effective and efficient use of a computer.

The information in this manual includes the following sections:

1. An introduction to A Series and B 5000/B 6000/B 7000 Series systems I/O capabilities and conventions

2. A description of the interface between programs and file attributes, which includes discussions of file attribute assignment, file equation, file attribute inquiry, I/O statements, and Logical I/O across a BNA network

3. Descriptions of physical files and related attributes, organized according to kind of device

4. Detailed descriptions of general file attributes

5. An overview of Direct I/O and individual descriptions of buffer attributes

6. A discussion of duplicated files

7. A description of hardware and software translation

8. A description of the format of external file names

9. A discussion of disk file and system access security

In addition, appendixes describe library maintenance tapes, pack labels, interchange directory records, interchange file headers, and standard tape label formats. An appendix describing the handling of attributes by FORTRAN77 programs is also provided.

All of the material in this manual is information that is visible to the user. There is no internal description of MCP procedures.

1.2 RELATED DOCUMENTS

Document	Form No.
Burroughs Network Architecture (BNA) Version 2 Installation Guide	1169778
ALGOL Reference Manual	1169844
CANDE Reference Manual	1169869
COBOL ANSI-74 Reference Manual	1169877
COBOL Reference Manual	1169786
DCALGOL Reference Manual	5014574
DiagnosticMCS Reference Manual	1169596
DMSII User Language Interface Software Operation Guide	1180536
Generalized Message Control System (GEMCOS) Capabilities Manual	1100211
Operator Display Terminal (ODT) Reference Manual	1169612
Physical I/O Overview	1169943
Print System (PrintS/ReprintS) User's Guide	1169919
Remote Job Entry (RJE) Reference Manual	1169828
System Software Site Management Reference Manual	1170008
System Software Support Reference Manual	1170016
System Software Utilities Reference Manual	1170024
Work Flow Language (WFL) Reference Manual	1169802

1.3 OBJECTIVES OF THE INPUT/OUTPUT SUBSYSTEM

The following section describes some important objectives of the Input/Output Subsystem. The system can best be understood and used when these underlying objectives are kept in mind. Some of these objectives are general to the MCP and the system, but they apply in particular to the I/O Subsystem.

The I/O Subsystem is intended to be transparent to the user program. The user sees a simple, familiar, and convenient user interface. In most cases, the user need not be concerned about the detailed internal operations of the I/O Subsystem. Therefore, little of that detail is included in this manual.

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

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.

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 minimal impact on the program.

The I/O 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.

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

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

The system is designed to operate efficiently with minimal tuning. However, there are some areas in which a programmer who knows the characteristics of his particular program, or an installation manager who knows the characteristics of his task environment, can tune the system better than the system can tune itself. The ability to tune in important areas, with minimal burden on the user, is intended. The system is tuned by using simple parameters.

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

1.4 CONCEPTS OF FILE HANDLING

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

RECORDS

A record is a set of data containing 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 I/O Subsystem, which deals with complete records. The I/O Subsystem gives the program one record at a time and takes from it one record at a time.

FILES

A file is a group of related records. Files are of central importance in the I/O Subsystem, for most of the communication between programs and the I/O 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 prototypical 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 records that exist apart from the program. The program needs to interact with these records from time to time; the program arranges with the I/O Subsystem to have those records made available or sent back to external storage when ready. The I/O Subsystem moves the records into and out of the user working areas.

File Structure

A file has a particular structure that is seen by the program. The records of the file 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 I/O Subsystem. If the length is variable, the I/O 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, they 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.

Physical File

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 that are packaged together so that they can be transferred to or from the physical file as a group. Furthermore, 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.

A physical file can be either temporary or permanent. A temporary file is one that exists only at the time of its original creation; it is of no further interest to any program. Examples of temporary files include 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 visibility to the general system, and exist for the I/O Subsystem only while the logical files that created them remain assigned.

Logical File

The file, and its structure as seen by the program, is called a "logical file". A logical file is a file variable declared within a 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 within blocks to which 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 that 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.

A logical file exists in one of four states: open-assigned, closed-unassigned, closed-assigned (also known as closed-retained), and open-unassigned. Before data can be transferred between a logical and a physical file, the logical file must be open and the physical file must be assigned to the logical file. This assignment can be accomplished explicitly by opening the logical file or by means of the AVAILABLE attribute. If the OPTIONAL attribute is TRUE for a logical file, opening the file can leave it unassigned to a physical file. (For complete descriptions of the AVAILABLE and OPTIONAL attributes, refer to "General File Attributes".) 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 it can be closed with release, which severs the connection between the logical and physical files.

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

FILE NAMES

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 the physical file is stored on tape, the external file name is recorded in one of the tape labels. If the physical file is stored on cards, the external file name is punched on a control card preceding the data cards. The I/O Subsystem maintains the disk directory; it also reads and keeps track of all label information on physical files that have been loaded by the operator. When it creates a new file, as on tape, it automatically writes the necessary labels, including the external file name and other label information.

PHYSICAL/LOGICAL FILE INTERACTION AND ASSIGNMENT

The I/O Subsystem acts as an intermediary between the logical file and the physical file associated with it. The I/O 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 I/O 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 physical I/O operations only when necessary.

One task of the I/O Subsystem is to make the file assignment, namely, 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).

USE OF BUFFERS

To smoothly expedite the flow of I/O operations, the I/O Subsystem makes use of buffers. A buffer is an intermediate storage area, under control of the I/O Subsystem, that is used to store data in transit between the physical file and the user work area. Typically, two buffers are 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.

1.5 THE CONCEPT OF 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.

FILES AS SYSTEM COMPONENTS

Each programming language provides some means of describing files and manipulating input/output operations. Some languages have been enhanced by Burroughs extensions.

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

In a multilanguage, multiprogram environment with applications of varying complexity, a file cannot be viewed as the simple property of a single program. In systems oriented toward data communications, time-sharing, or database 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, language-general way.

This does not mean that the programmer writing in COBOL, for example, must learn a new language and rewrite 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 Work Flow Language (WFL).

DEFINITION OF A FILE ATTRIBUTE

File attributes are control parameters that contain all the information the I/O Subsystem needs when it connects a physical file to the logical file of a program. They can also be used to indicate file structure, to control file access, and to obtain the status of a file.

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 information specifying whether the file is to be used for input, output, or both.

Besides the file attributes used directly by the I/O Subsystem for file and input/output control, there is another class of attributes that is used for synchronization and intercommunication between running programs or between running programs and the I/O Subsystem. These attributes are maintained by the I/O Subsystem and typically are not modifiable by user programs.

SPECIFYING ATTRIBUTE VALUES

Various programming languages provide ways of stating file attribute information. The programmer provides it in the way defined for the language he is using. If the information is not provided (where its omission is allowed in the language used), the system provides the most reasonable defaults. Attribute values can be explicitly specified in file declarations in the various languages. The syntax for performing this equation varies depending on the style of the individual language. The appropriate syntax is given in the manual that describes that particular language.

The I/O Subsystem allows dynamic file definition; that is, the precise nature of all attributes need not be known at the time the program is written. Furthermore, some attributes, even those defined in the program, can be changed at a later time without having to modify the program.

File attributes can be changed when the program is compiled or executed by the use of file equation statements in the Work Flow Language. The specific syntax for file equation statements is given in the Work Flow Language Reference Manual under "File and Database Equations".

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

ATTRIBUTE CONSISTENCY

Some combinations of attributes are reasonable; others are not. The I/O Subsystem checks attributes for consistency. Although it generally disallows combinations that are self-contradictory or that logically indicate errors, the I/O Subsystem is permissive in most cases. That is, 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/or TITLE attributes.

Every logical file also has an internal name (INTNAME). The default value for the internal name is the identifier used to declare the file variable. The I/O Subsystem uses the value of the INTNAME attribute as default values for FILENAME and TITLE when the values of FILENAME and TITLE are unspecified.

The FILENAME and TITLE attributes are used to identify files on peripheral devices. When the file is on disk, the name of the family (as distinguished from the name of the file) can also be given by assigning the name to the FAMILYNAME attribute.

In a file maintenance situation, the name alone is not always sufficient 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 that 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.

One of the most important attributes is KIND, which describes the class of peripheral device(s) associated with the logical file.

I/O SUBSYSTEM

The I/O 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 the LABEL attribute to either OMITTED or OMITTEDEOF. This lack of label records makes it impossible for the system to automatically associate a physical file with the logical file. As a consequence, the system operator must assign the physical file by means of a system message dialog with the I/O Subsystem at the time the logical file is opened.

See also

CYCLE 159
FAMILYNAME. 183
FILENAME. 188
INTNAME 215
KIND. 218
LABEL 220
TITLE 314
VERSION 335

ATTRIBUTES USED FOR FILE STRUCTURING

The BLOCKSIZE, MAXRECSIZE, and MINRECSIZE attributes determine the lengths of the records and blocks of a file. The values for BLOCKSIZE, MAXRECSIZE, and MINRECSIZE are specified in units called frames. The FRAMESIZE attribute (or, alternatively, the INTMODE and UNITS attributes) determines the number of bits in each frame. For more specific information, refer to the descriptions of these attributes under "General File Attributes".

Records can be either fixed length or variable length. If variable, the allowable limits of record size are given in two attributes, MAXRECSIZE and MINRECSIZE. For fixed-length records, both attributes 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 SIZEMODE, SIZEOFFSET, and SIZE2 attributes are also needed.

The FILEKIND attribute allows the identification of the internal structure of the file and its records.

The FILEORGANIZATION attribute affects the internal structure of a file and can be used to restrict the use of files to uses consistent with the file's creation.

See also
BLOCKSIZE 133
FILEORGANIZATION. 191
FILETYPE. 198
INTMODE 213
MAXRECSIZE. 232
MINRECSIZE. 236
SIZEMODE. 294
SIZEOFFSET. 295
SIZE2 298
UNITS 328

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, PRESENT, and RESIDENT, 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.

In actuality, these attributes perform a full or partial conditional OPEN action. AVAILABLE, the most useful of these attributes, can be used as a form of the OPEN statement in some languages.

See also
AVAILABLE 122
PRESENT 259
RESIDENT. 275

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 guard file if GUARDED or CONTROLLED security is required. The security system is described in more detail under "Disk File and System-Access Security".

See also
SECURITYTYPE. 285
SECURITYUSE 287
Disk File And System-Access Security. 391

ATTRIBUTES USED FOR PRINT/PUNCH FILES

Print attributes describe the characteristics applicable to a file when it is assigned or routed to a printing device. These attributes can be read or assigned values from a program and can be file-equated upon task initiation. Print attributes are useful only when a given file has the attribute KIND = PRINTER or KIND = PUNCH.

See also
AFTER 109
BANNER. 129
DESTINATION 165
FORMID. 204
NOTE. 248
PRINTCHARGE 261
PRINTCOPIES 262
PRINTDISPOSITION. 263
PRINTERCONTROL. 264
PRINTERKIND 266
SAVEBACKUPFILE. 279
TRANSFORM 318
TRAINID 317
USERBACKUPNAME. 332

ATTRIBUTES USED FOR TRANSLATION

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

Translation is discussed in more detail in the description of the TRANSLATE attribute under "General File Attributes" and in the section entitled "Hardware and Software Translation".

See also
EXTMODE 178
INTMODE 213
INPUTTABLE. 211
OUTPUTTABLE 253
TRANSLATE 319
TRANSLATING 322
Hardware and Software Translation 377

DIAGNOSTIC ATTRIBUTES

As already stated, 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 I/O Subsystem makes consistency checks on attributes. When there are errors, the ATTERR, ATTVALUE, and ATTYPE attributes return information that may be helpful in program debugging.

The I/O Subsystem does extensive checking on physical I/O operations and translates error and exception information into a result descriptor that is meaningful to the user. This result descriptor is returned for each logical I/O operation and is directly accessible in some of the languages. The STATE attribute can also be used to access this information from the last I/O operation. The I/O Subsystem relates the error information as closely as possible to a logical record and uses the RECORDINERROR attribute to return the record number of that logical record.

See also
ATTERR. 119
ATTVALUE. 120
ATTYPE. 121
KIND. 218
RECORDINERROR 272
STATE 299

2 OVERVIEW OF PROGRAMMATIC INTERFACES

The following sets of of programmatic constructs are relevant to files: file declaration, file equation, file assignment, file attribute inquiry, and the file operations OPEN, WRITE, READ, SEEK, and CLOSE.

FILE DECLARATION

A file declaration is used to associate an internal name (INTNAME) with a logical file (a specific instance of the internal system structure) used by the user program. A file declaration can also be used to assign values to the attributes associated with the file. Sample file declarations in ALGOL and COBOL74 follow.

Examples (ALGOL)

```
FILE F(KIND=DISK,NEWFILE=FALSE,DEPENDENTSPECS=TRUE);

FILE OUTPUT_FILE;
```

Examples (COBOL74)

```
FD  IN-FILE.

FD  UPDATE-FILE;
        VALUE OF DEPENDENTSPECS IS TRUE,
                 FILENAME IS "MASTER/UPDATE.".
```

See also
INTNAME 215

FILE EQUATION

File equation is a general mechanism for specifying the values of attributes when the program is initiated or compiled. For example, a program can be initiated using a KIND of file different from that already specified in the file declaration by file equating KIND to some other value. As another example, a program could be written to handle any file of a specific type, allowing the user to indicate a particular file by file equating the FILENAME attribute appropriately.

For a syntactical description of file equation, refer to the Work Flow Language Reference Manual under "File and Database Equations". For a description of how file equation affects the program, refer to the introduction to "General File Attributes".

See also

General File Attributes 95
FILENAME. 188
KIND. 218

FILE OPERATIONS

Run-time statements are used to perform operations on the logical file, the physical file, or both. The following run-time statements or file operations can modify the state of a file in various ways:

a. The OPEN statement

b. The WRITE statement

c. The READ statement

d. The SEEK statement

e. The CLOSE statement

f. Attribute assignment

g. Attribute interrogation

OPEN and CLOSE are complementary statements, as are READ and WRITE, and attribute assignment and attribute interrogation. SEEK is a statement that is associated with both reading and writing, and can modify a read or write that follows it. READ, WRITE, and SEEK are all input/output statements.

The OPEN Statement

The purpose of the OPEN statement is to assign the file to a physical file and mark the logical file as open. The input/output statements require the logical file to be open before they can perform their functions. (An implicit open is performed by default if an input/output statement is attempted while the logical file is closed, and then the input/output statement is performed; however, some languages do not allow implicit opens.) In most cases, a physical file is assigned to the logical file when the open statement is performed.

Some versions of the OPEN statement are conditional, in that they open the logical file only if a physical file satisfying specified matching criteria is found. In addition, the initial positioning of the logical file can be specified.

If requested, an OPEN statement can return a result indicating success or the reason for failure. The value is returned in the same format as the value of the AVAILABLE attribute. If a result is not requested, then the system takes default actions for each result that would have been returned.

As part of the open process in BNA Version 2 (BNA V2) port files, the value of the ACTUALMAXRECSIZE attribute to be used in the conversation between the two subfiles is negotiated. The negotiated value is never greater than the smaller of the two REQUESTEDMAXRECSIZE values. In addition, the TRANSLATE attribute is negotiated to determine the extent of translation for this dialog.

For languages that allow implicit file opens, port files can be implicitly opened by an I/O operation on the file. If a nonzero subfile-index is specified, then only that subfile will be opened. If a subfile-index of zero is specified, then no implicit open action will take place. If a subfile-index is not specified and MAXSUBFILES is equal to 1, then the subfile will be implicitly opened. Any port file implicitly opened by an I/O operation should be opened just as though the user had, at that point, specified an OPEN using the default option.

Note that values normally associated with closing a file may be returned during an open operation and values normally associated with opening a file may be returned during a close operation if the KIND is TAPE, DISKETTE, PAPERPUNCH, or PAPERREADER (or the KIND is PRINTER or PUNCH and the BACKUPKIND is TAPE) and the open or close causes a volume switch to occur.

Examples (ALGOL)

```
OPEN(F);

I := OPEN(SOURCE_FILE,AVAILABLE);
```

Examples (COBOL74)

```
OPEN INPUT IN-FILE.

OPEN EXTEND UPDATE-FILE.
```

See also
File Assignment 28
AVAILABLE 122

The WRITE Statement

The WRITE statement transfers data from the user program to the file. (Normally, data is actually transferred to the physical file only if a buffer is filled; otherwise, the data is stored in buffers in the logical file.) There are two major types of writes: serial and random.

For a serial write, the output is written to the next record in the file. For a random write, the output is written to a specified record in the file (which is not usually the next sequential record) and the normal sequence of counting of next records is modified. If requested, a WRITE statement can return a result indicating success or the reason for failure. The value is returned in the same format as that returned by the STATE attribute. (Refer to the description of the STATE attribute under "General File Attributes".) If a result is not requested, then the system takes default actions for each result that would have been returned.

Examples (ALGOL)

```
WRITE(F,90,A);

B := WRITE(OUT_FILE[REC_NUM],OUT_RECORD_SIZE,OUTPUT_POINTER);
```

Examples (COBOL74)

```
WRITE OUT-RECORD.

MOVE EMPLOYEE-NUMBER TO UPDATE-KEY.
WRITE UPDATE-DATA;
    INVALID KEY PERFORM HANDLE-INVALID-KEY.
```

See also
STATE 299

The READ Statement

The READ statement transfers data to the user's program from the file. (Normally, data is actually transferred from the physical file only if a buffer is emptied; otherwise, the data is transferred from buffers in the logical file.) There are two major types of reads: serial and random.

For a serial read, the input is read from the next record in the file. For a random read, the input is read from a user-specified record in the file (which is not usually the next sequential record) and the normal sequence of counting of next records is modified. If requested, a READ statement can return a result indicating success or the reason for failure. The value is returned in the same format as that returned by the STATE attribute. (Refer to the description of the STATE attribute under "General File Attributes".) If a result is not requested, then the system takes default actions for each result that would have been returned.

Examples (ALGOL)

```
READ(F,90,A);

B := READ(IN_FILE[REC_NUM],IN_RECORD_SIZE,INPUT_POINTER);
```

Examples (COBOL74)

```
READ IN-FILE.

MOVE NEXT-FILE-INDEX TO RANDOM-KEY.
READ RANDOM-FILE;
    INVALID KEY GO TO DISPLAY-KEY-ERROR.
```

See also
STATE 299

The SEEK Statement

The SEEK statement modifies the pointer that specifies the next record in the file at which a subsequent serial read or write will be performed. The SEEK statement does not transfer any data to or from the user's array. If requested, a SEEK statement can return a result indicating success or the reason for failure. The value is returned in the same format as that returned by the STATE attribute. (Refer to the description of the STATE attribute under "General File Attributes".) If a result is not requested, then the system takes default actions for each result that would have been returned.

Examples (ALGOL)

```
SEEK(F[0]);

SEEK(F[SAVED_RECORD_NUMBER]);

SPACE(F,-1);
```

Example (COBOL74)

```
MOVE NEXT-KEY TO UPDATE-KEY.
SEEK UPDATE-FILE.
```

See also
STATE 299

The CLOSE Statement

The CLOSE statement marks a logical file as closed (closed is the opposite of open). In many cases, the physical file is disassociated from the logical file when the close statement is performed. If desired, the disposition of the physical file can be specified. Some languages require the file to be closed before the program is exited. If requested, a CLOSE statement can return a result indicating success or the reason for failure. The value is returned in the same format as the value of the AVAILABLE attribute. If a result is not requested, then the system takes default actions for each result that would have been returned.

Examples (ALGOL)

```
CLOSE(F);

CLOSE(F,CRUNCH);

CLOSE(F,REWIND);
```

Examples (COBOL74)

```
CLOSE UPDATE-FILE.

CLOSE OUT-FILE WITH SAVE.
```

See also
AVAILABLE 122

Attribute Assignment

Attribute assignment modifies the value of a file attribute. Some attributes are not modifiable (in other words, they are read-only). Some attributes are modifiable only while the file is closed, and some are modifiable only while the file is unassigned. The attributes that are modifiable only while the file is closed are associated with the structure of the logical file. The attributes that are modifiable only while the file is unassigned are associated with the structure of the physical file or are used to specify the matching criteria for assigning the physical file to the logical file.

Examples (ALGOL)

```
F.NEWFILE := TRUE;

REPLACE TEST_FILE.FILENAME BY "TEST/PROGRAM.";

TERM.BLOCKSTRUCTURE := VALUE(EXTERNAL);
```

Examples (COBOL74)

```
CHANGE ATTRIBUTE DEPENDENTSPECS OF
    IN-FILE TO VALUE TRUE.

CHANGE ATTRIBUTE FILENAME OF
    OUT-FILE TO "OUTPUT/MASTER.".

CHANGE ATTRIBUTE BLOCKSTRUCTURE OF
    TERMINAL-FILE TO VALUE EXTERNAL.
```

Attribute Interrogation

Attribute interrogation provides the user program with the current value of a file attribute. For a few attributes, interrogation is not allowed (in other words, the attributes are write-only). Interrogation is feasible for some attributes only while the file is open and for others only while the file is assigned. Those attributes for which interrogation is feasible only while the file is open report part of the current state of the logical file. Those attributes for which interrogation is feasible only while the file is assigned report part of the current state of the physical file.

Examples (ALGOL)

```
B := F.ATTERR;

REPLACE POINTER(TEST_FILENAME) BY
     TEST_FILE.FILENAME;

IF IN_FILE.BLOCKSTRUCTURE NEQ VALUE(FIXED) THEN
     HANDLE_VARIABLE_FILE;
```

Examples (COBOL74)

```
MOVE ATTRIBUTE CRUNCHED OF IN-FILE TO SAVE-CRUNCH.

MOVE ATTRIBUTE FILENAME OF OUT-FILE TO
    FILENAME-TEMP.

IF ATTRIBUTE BLOCKSTRUCTURE OF IN-FILE IS
        NOT EQUAL TO VALUE FIXED
    PERFORM HANDLE-VARIABLE-FILE.
```

FILE ASSIGNMENT

Before a program can create data or access data contained in a physical file, an association between the physical file and the program must be made. This association is made 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 are 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 I/O Subsystem attempts to associate the logical and physical files without requiring intervention by the system operator.

Most languages provide an OPEN statement or the equivalent. The user should consult the appropriate language manual for a description of the open statement, or equivalent, for that language. In 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 the same values as those returned by the AVAILABLE attribute. Refer to the description of the AVAILABLE attribute under "General File Attributes".

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 terminated with an error message in one of the following forms:

FILE <title> OPEN ERROR: <error message>

FILE <title> CLOSE ERROR: <error message>

If an error occurs on an I/O operation that discards the I/O result descriptor, the program is terminated 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 program accepts responsibility for handling the error by referencing the value returned by the open, close, or I/O operation.

See also

AVAILABLE 122

New Files Versus Permanent Files

The KIND, MYUSE, and NEWFILE attributes determine whether a new file is created or a permanent file is assigned when the file is opened. Table 2-1 shows the decisions made by the I/O Subsystem based on the values of these attributes.

Table 2-1. Attributes Affecting New File/Permanent File Decision

KIND	NEWFILE	MYUSE	File Found	Open Action
Output-only devices (for example, PRINTER or PUNCH)	--(1)	--	--	New file (2)
Input-only devices (for example, READER or PAPER-READER)	--	--	yes	Existing file (3)
			no	No file msg (4)
Input/Output device (for example, 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
DISK	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

NOTES

1. A dash indicates that, given the values of other attributes, this attribute does not 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.

- As of the Mark 3.5 release, the value of AREASIZE no longer affects the new file/permanent file decision.

When the KIND of a logical file is an output-only peripheral (for example, PRINTER or PUNCH), a new file is created. When the KIND is 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, a KIND of TAPE or DISK), the MYUSE and NEWFILE attributes determine whether a new file is created or an existing file is assigned.

As of the Mark 3.5 release, AREASIZE does not affect file assignment. A new file is always created when NEWFILE is unspecified and MYUSE is OUT. Prior to the Mark 3.5 release, a run-time warning message was reported whenever a permanent disk file was assigned to a logical file with MYUSE specified as OUT and AREASIZE either unspecified or 0.

Creating a new file on a labeled tape causes header records to be written at the time the file is opened and trailer records to be written at the time the file is closed. Creating a new file on an unlabeled tape file causes tape marks to be written out when the file is closed.

See also
Device Dependencies 45
KIND. 218
MYUSE 241
NEWFILE 243

Selecting a Peripheral for a New File

When a 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 non-disk file, a peripheral is assigned on the basis of the availability of a scratch or unassigned peripheral. If a logical file is used to create a new tape file and the SERIALNO attribute is specified, the tape does not have to be a scratch tape. A tape with a matching serial number that is not locked, not saved, not not-ready, and not in use and that has a write ring is selected. If it has not been rewound, the tape is rewound and new labels are 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 a backup file is determined by the interaction of system options and file attributes.

If a logical file is used to create a 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 is selected.

See also
FAMILYNAME. 183
FORMID. 204
KIND. 218
SERIALNO. 289

Finding a Permanent File

When a logical file is to be assigned to a permanent file, a number of attributes (KIND, FILENAME, FAMILYNAME, FILESECTION, CYCLE, VERSION, SERIALNO) are used in an attempt to uniquely describe the physical file. The KIND attribute is used to narrow the search to certain peripherals. The FILENAME attribute gives the external file name of the permanent file and, where appropriate, the family name.

Once the permanent file with the proper FILENAME 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 that 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 specified for a tape file, then the serial number of the physical tape must match the value of the SERIALNO attribute.

See also
CYCLE 159
FAMILYNAME. 183
FILENAME. 188
FILESECTION 193
KIND. 218
SERIALNO. 289
VERSION 335

Interactions of the FILEUSE 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 FILEUSE 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 (that is, if not opened via the AVAILABLE attribute) and the logical file is not optional, the program is terminated instead for a security violation.

If the value of the SECURITYUSE attribute for a physical file is incompatible with the value of FILEUSE (if FILEUSE is specified) or with the value of MYUSE (if FILEUSE is not specified) of the logical file, the following occur:

1. An attempt to open the file results in a security error.

2. When interrogated, the RESIDENT and PRESENT attributes return FALSE.

3. When interrogated, the AVAILABLE attribute returns "2" (no file).

See also

Restrictions Imposed by Peripheral Association

Under certain conditions imposed by the management of an installation, the actual peripheral devices that a program can see are limited. For example, if an installation manager has a large number of small jobs that require fast turnaround, he might want to designate one particular card reader and one particular printer to handle these jobs. He might install these two devices in a room for people with this class of job to use. He would then want some way of restricting the search of a program entered through that card reader to those peripheral devices specified in this peripheral association group. (Refer to the PA (Peripheral Association) ODT command described in the Operator Display Terminal (ODT) Reference Manual.) In this case, only those devices within the peripheral association group are searched for file assignment for KINDs in the peripheral association as described above.

Circumstances Requiring Operator Intervention

Tape, paper tape, and card files can be declared to be unlabeled files (where LABEL is either 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 multiple files meet the assignment criteria. In such cases, a "NO FILE" condition or a "DUPLICATE FILE" condition exists. In the former case, the condition is signaled by a "NO FILE" message when searching for an existing file or a "REQUIRES" message when attempting to create a new file. Under either of these conditions, the system periodically searches for the permanent file. If the search succeeds due to changed conditions, the file assignment proceeds automatically.

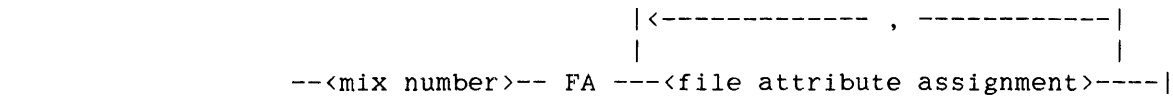
Given the condition of an unlabeled file, a "NO FILE", or a "DUPLICATE FILE", the system operator may assign a physical file to the logical file with either an IL (Ignore Label) or UL (UnLabeled) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.) If the program is requesting a labeled file, but the system operator enters a UL command, then the file is treated as unlabeled (whether or not the physical file is created as labeled). In this case, the program will read the labels as the first data records.

If the operator intervenes in the assignment of a labeled permanent file and DEPENDENTSPECS is TRUE (or FILETYPE is 7 or 8), the logical file assumes the structure of 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 by entering the OU (Output Unit) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.) The operator may not use the OU command to assign a peripheral with a different general KIND than that requested by the logical file, but a different KIND for backup or a different specific KIND within the same general KIND may be assigned.

If the KIND is PRINTER or PUNCH, there may be no peripheral that matches the FORMID attribute. In this case, the operator may use the FORM (Assign FORM ID) ODT command to FORM an available peripheral that otherwise meets the requirements of the logical file or may respond with an FM (Form Message) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

The operator may also change the requirements for file assignment by changing some of the file attributes with the FA (File Attribute) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of the FA command.) This command has the following format:



Refer to the Work Flow Language Reference Manual for a description of <file attribute assignment>.

The following is an example of the FA (File Attribute) command:

```
1234 FA VERSION=12, FILENAME=X/Y, KIND=TAPE
```

If the user program has set the OPTIONAL attribute to TRUE, then the operator may additionally respond with an OF (Optional File) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.)

See also

LOGICAL I/O ACROSS A BNA NETWORK

The BNA Logical I/O Host Service allows programs running on one host system to access and create files located on a remote host system. The HOSTNAME attribute is used to indicate the name of the host on which the file resides. The value of HOSTNAME may be specified either in the program or by file equation.

Examples

```
?BEGIN JOB FOREIGN/COMPILE;
  COMPILE PROG COBOL;
  COBOL FILE CARD(KIND=DISK,FILENAME=S/PROG,HOSTNAME=D);
?END JOB

BEGIN
  FILE F(KIND=DISK,DEPENDENTSPECS=TRUE,FILENAME="THE/FILE.",
         HOSTNAME="E.");
  ARRAY A[0:12];
  LABEL EOF;
  WHILE TRUE DO
    BEGIN
      READ(F,12,A)[EOF];
      .
      .
    END;
  EOF:
    CLOSE(F);
END.

BEGIN
  FILE F(KIND=PACK,MAXRECSIZE=14,FILENAME="OVER/THERE.",
         NEWFILE=TRUE,
         HOSTNAME="E.");
  ARRAY A[0:12];
  BOOLEAN DONE;
  DO
    BEGIN
      .
      .
      WRITE(F,12,A);
      .
      .
    END
  UNTIL DONE;
  LOCK(F);
END.
```

The host on which the program is running is known as the "process host". The host on which the file resides is known as the "file host".

When the file is opened, a handler task named FILE/HANDLER/<process-hostname> is initiated by the BNA Logical I/O Host Service on the file host. This task performs all I/O Subsystem functions on the file. Messages and RSVPs pertaining to the file include the file hostname and the handler task's mix number when displayed at the process host. A user at the process host may reply to RSVPs by using the AT (AT Remote Host) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.) The command should be of the following form:

AT <file-hostname> <handler mix number> <ODT command>

Example

AT BLUE 6543 FA FILENAME = MYFILE

See also
HOSTNAME. 208

Restrictions

The BNA Logical I/O Host Service supports a subset of the capabilities provided by the I/O Subsystem. The following restrictions are imposed by the BNA Logical I/O Host Service when both the process host and the file host are A Series or B 5000/B 6000/B 7000 Series systems:

1. A program accessing or creating a file at a remote host must be running under a usercode.
2. Direct I/O is not supported.
3. Keyed I/O is not supported.
4. Relative I/O is not supported.
5. BASIC programs may not access or create files on remote hosts.
6. Compilers may not create code files on remote hosts.
7. USE routines for tape labels are not supported.

8. Update I/O action is not supported. COBOL programs may not specify the I-O and O-I options in an OPEN statement. COBOL74 programs may not specify the I-O option in an OPEN statement. The value of the UPDATEFILE attribute must be FALSE.

9. In order to create a file on a host that is not an A Series or B 5000/B 6000/B 7000 Series system, the value of the NEWFILE attribute must be TRUE. If the value of the NEWFILE attribute is modified to FALSE, the BNA Logical I/O Host Service searches for an existing file. Note that NEWFILE cannot be changed by the FA (File Attribute) ODT command if the program hangs on a "NO FILE" condition.

   For COBOL or COBOL74 programs, the compiler modifies the value of NEWFILE attribute appropriately on an OPEN statement, so the program does not have to consider this situation.

10. Error results for WRITE statements are reported one WRITE after the WRITE statement that reuses the buffer that originally had the error, instead of exactly at the WRITE that reuses the buffer that had the error. (That is, in this case, the error is reported one buffer later than normal.)

11. The KIND attribute must be specified. (Refer to the description of the KIND attribute under "General File Attributes".)

12. A privileged status, whether from a privileged usercode or from a privileged program, is not carried across the network.

13. BLOCKSTRUCTURE = VARIABLE is not supported for files on a remote BNA host. Because of this, any file declared using the COBOL74 construct "integer-1 TO" cannot be opened if the file resides on a remote BNA host. An attempt to open such a file results in an open error.

14. Binary I/O is not supported. For example, the ALGOL statement "WRITE (FILE1,*,X)" is not supported.

15. Files with partial last records and files created by PLISUPPORT ISAM intrinsics are not supported.

See also
The OPEN Statement. 21
The WRITE Statement 23
KIND. 218
NEWFILE 243

Attributes Supported by BNA Logical I/O

FULL SUPPORT

The following attributes are fully supported by the BNA Logical I/O Host Service:

AREAALLOCATED
AREALENGTH
AREAS
ATTERR
AVAILABLE
BACKUPKIND
BLOCK
BUFFERS
CARRIAGECONTROL
CENSUS
COPIES
COPYINERROR
COPYNAME
CREATIONDATE
CRUNCHED
CURRENTBLOCK
CURRENTRECORD
CYCLE
DENSITY
DEPENDENTSPECS
DIRECTION
DISPOSITION
DUPLICATED
FAMILYINDEX
FAMILYNAME
FILESECTION
FILESTATE
FILEUSE
FLEXIBLE
FORMID
GENERATION
HOSTNAME
INPUTTABLE
INTNAME
LABEL
LABELKIND
LASTRECORD
LASTSUBFILE
LINENUM
MINRECSIZE
NEWFILE
NEXTRECORD
OPEN
OPTIONAL
OUTPUTTABLE
PAGE
PARITY
PRINTERCONTROL
PRINTERKIND
SAVEFACTOR
SCREEN
SCREENSIZE
SECURITYUSE
SENSITIVEDATA
SINGLEUNIT
STATIONCOUNT
STATIONLIST
STATIONNAME
STATIONSALLOWED
STATIONSDENIED
TAPEREELRECORD
TIMELIMIT
TRAINID
TRANSLATING
TRANSMISSIONNO
USECATALOG
USEDATE
VERSION
WIDTH
YOURHOST

For more information, refer to the relevant attribute descriptions under "General File Attributes".

RESTRICTED VALUES

The following attributes are supported by the BNA Logical I/O Host Service have restrictions on the values supported:

BLOCKSIZE	KIND
BLOCKSTRUCTURE	MAXRECSIZE
EXTMODE	SECURITYGUARD
FILEKIND	SECURITYTYPE
FILENAME	TIMELIMIT
FILEORGANIZATION	TITLE
FILETYPE	TRANSLATE
FRAMESIZE	UPDATEFILE
INTMODE	

For more information, refer to the relevant attribute descriptions under "General File Attributes".

RESTRICTED USAGE

The following attributes are supported by the BNA Logical I/O Host Service but have restrictions on their usage:

AREASIZE	PROTECTION
FILEKIND	SERIALNO
FILETYPE	STATE
MYUSE	TITLE
PAGESIZE	UNITS

For more information, refer to the relevant attribute descriptions under "General File Attributes".

Attributes Not Supported by BNA Logical I/O

The following attributes are not supported by the BNA Logical I/O Host Service:

ACTUALMAXRECSIZE
AFTER
ALTERDATE
ALTERTIME
APL
APPLICATIONGROUP
AREACLASS
ASSIGNTIME
ATTVALUE
ATTYPE
AVAILABLEONLY
BANNER
BLANK
BLOCKEDTIMEOUT
CHANGEDSUBFILE
CHANGEEVENT
COMPRESSING
COMPRESSION
COMPRESSIONCONTROL
COMPRESSIONREQUESTED
CREATIONTIME
CURRENTEXTENT
CYLINDERMODE
DESTINATION
DIALOGCHECKINTERVAL
DIALOGPRIORITY
ENABLEINPUT
EOF
ERRORTYPE
EXCLUSIVE
EXTENT
FAMILY
FAMILYSIZE
IAD
INPUTEVENT
INTERCHANGE
IOCLOCKS
IOINERROR
MAXCENSUS
MAXSUBFILES
MYHOST
MYHOSTGROUP
MYHOSTNAME
MYNAME
NORESOURCEWAIT
NOTE
OLDYOURUSERCODE
OUTPUTEVENT
POPULATION
PRESENT
PRINT
PRINTCHARGE
PRINTCOPIES
PRINTDISPOSITION
PRINTERCONTROL
RECEPTIONS
RECORD
RECORDINERROR
REQUESTEDMAXRECSIZE
RESIDENT
ROWADDRESS
ROWSINUSE
SAVEBACKUPFILE
SIZEMODE
SIZEOFFSET
SIZEVISIBLE
SIZE2
SUBFILEERROR
TANKING
TRANSFORM
TRANSMISSIONS
TRIMBLANKS
UNITNO
USERBACKUPNAME
USERINFO
USETIME
WRITECHECK
YOURHOSTGROUP
YOURNAME
YOURUSERCODE

In addition, Direct I/O buffer attributes are not supported by BNA Host Services.

For more information, refer to the relevant attribute descriptions under "General File Attributes".

See also

General File Attributes 95

3 DEVICE DEPENDENCIES

This section describes various physical files and the attributes that are relevant to particular physical devices.

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

ALL DEVICES

The following attributes generally apply to all devices:

ATTERR	KIND
ATTVALUE	LABEL
ATTYPE	MAXRECSIZE
AVAILABLE	MINRECSIZE
AVAILABLEONLY	MYUSE
BLANK	NEWFILE
BLOCKSIZE	NEXTRECORD
BLOCKSTRUCTURE	OPEN
BUFFERS	OPTIONAL
CURRENTBLOCK	OUTPUTTABLE
CURRENTRECORD	PRESENT
DEPENDENTSPECS	RECORD
EXTMODE	RESIDENT
FILENAME	SIZEMODE
FILESTATE	SIZEOFFSET
FILETYPE	SIZEVISIBLE
FILEUSE	SIZE2
FRAMESIZE	STATE
HOSTNAME	TITLE
INPUTTABLE	TRANSLATE
INTMODE	TRANSLATING
INTNAME	UNITS
IOCLOCKS	YOURHOST

In some cases, these general attributes do not apply to a specific device type. Refer to the relevant attribute descriptions under "General File Attributes" and to the device headings in this section.

BACKUP FILES

In general, there are two kinds of backup files, disk and tape. Either kind can be used for printer, punch or image printer files.

When the SERIALNO attribute is specified for a printer, punch, or image printer backup file, the system selects the unit with the specified serial number or waits for an RSVP at backup file creation time if the disk or pack with the proper serial number is not available.

The FAMILYNAME attribute is also used in unit selection for printer, punch, or image printer backup disk files.

CARD FILES

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

READER	Card reader
PUNCH	Card punch

For card punch files, the BACKUPKIND and FORMID attributes are meaningful. For card punch files directed to backup disk or tape, the TRIMBLANKS and SERIALNO attributes are meaningful. For card punch files directed to backup disk, the SECURITYUSE and SECURITYTYPE attributes are meaningful. For card punch files directed to backup tape, the DENSITY and PARITY attributes are meaningful.

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

Card files have the 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, 12 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.

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 are reduced, and a run-time attribute error is generated when the file is opened.

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

<I> <mode> <title option>

where <I> represents an invalid character punch; <mode> is BCL, EBCDIC, DATA, or BINARY; <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 is 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 be opened only 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 (UnLabeled) ODT command to assign the physical unit to the logical file, before cards (without the beginning label record) are put into the card reader. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of the UL command.)

See also

All Devices 45
General File Attributes 95

DISK FILES

The KIND values associated with disk are DISK and PACK, which are logically synonymous values that refer to removable or nonremovable mass storage, head-per-track storage, or movable-head disk or disk pack.

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

```
BACKUPKIND = DISK
```

The following attributes, in addition to those listed under "All Devices", are meaningful for disk files:

ALTERDATE
ALTERTIME
APL
AREAALLOCATED
AREALENGTH
AREAS
AREASIZE
BLOCK
COPIES
COPYINERROR
COPYNAME
CREATIONDATE
CREATIONTIME
CRUNCHED
CYCLE
DUPLICATED
ERRORTYPE
EXCLUSIVE
FAMILYINDEX
FAMILYNAME
FILEKIND
FILEORGANIZATION
FLEXIBLE
GENERATION
IAD
IOINERROR
LASTRECORD
LASTRECORD
NORESOURCEWAIT
POPULATION
PROTECTION
RECORDINERROR
ROWADDRESS
ROWSINUSE
SAVEFACTOR
SECURITYGUARD
*SECURITYTYPE
*SECURITYUSE
SENSITIVEDATA
*SERIALNO
SINGLEUNIT
UPDATEFILE
USECATALOG
USEDATE
USERINFO
USETIME
VERSION

Those attributes marked with an asterisk (*) are also meaningful for backup disk.

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

In addition to attributes useful for disk, the CYLINDERMODE and INTERCHANGE attributes are meaningful for KIND=PACK.

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

The smallest addressable space of a head-per-track disk unit is a 30-word segment (180 EBCDIC bytes). The smallest contiguous space addressable on a disk pack is a 30-word sector (180 EBCDIC bytes).

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

Another element of storage structure imposed upon 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 using 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 that 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 AREAS attribute, which specifies the maximum number of areas to be assigned. If the value of AREAS 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 default value), this limit can be exceeded.

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

One technique provided by the system to enhance the integrity of disk files is that of duplicated files. For more information, refer to "Duplicated Files".

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, changing the physical disk file into a permanent file is performed when the file is closed after creation by a statement in the program such as CLOSE (F, LOCK) or CLOSE WITH CRUNCH. A file can also be made a permanent file from the time it is originally opened by means of the PROTECTION attribute. 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 when their sectors are of the same length and are addressed in the same manner.

When a disk unit is initialized, reconfigured, or labeled through the RC (ReConfigure Disk) or LB (ReLaBel Pack) ODT commands, label records in the ANSI (USASI) volume and header tape label formats, which contain the family name, are written on the unit. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.) The family name is a single identifier of up to 17 characters. It may be used as an identification check by means of the FAMILYNAME attribute. 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 new unit has the same serial number as another unit currently on-line, the new unit is marked "DUP SN" and is not made available for use.

The directory contains a list of all of the files on a family and the information describing them. A multiunit family has a base unit, which is either the member of the family on which the directory resides or, if the family has duplicate directories, the first member of the family that the MCP sees when the family becomes available. (Refer to the descriptions of the RC (ReConfigure Disk) and DD (Directory Duplicate) ODT commands in the Operator Display Terminal (ODT) Reference Manual.)

Space for files on multiunit families is allocated on each member in a rotational manner. The CYLINDERMODE and/or SINGLEUNIT attributes can be used to modify this mode of allocation.

The logic used for initiating physical I/O operations to disk 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.

Interchange Packs

Files are organized upon interchange disk packs differently than they are organized 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. The interchange mode may be selected when pack files are created by using the INTERCHANGE attribute; otherwise, the native format is used. The interchange compatibility includes some restrictions on the files residing on interchange packs. The external file name (FILENAME) of a file residing on an interchange pack must be a single identifier, up to eight characters in length (if the FILENAME is longer, it is truncated). If the FILENAME 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 multiple file identifier (MFID) is used to identify the name of the interchange pack unless the FAMILYNAME attribute has also been initialized. In this case, the MFID is completely ignored.

See also
All Devices 45
General File Attributes 95
Duplicated Files. 373

DISKETTE FILES

The KIND value associated with diskette files is DISKETTE.

The following attributes, in addition to those listed under "All Devices", are meaningful for diskette files:

CURRENTEXTENT	FILESECTION
EXTENT	WRITECHECK

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

The following paragraphs describe the use of files stored on diskette, also known as B9489-16 Industry-Compatible Mini-Disks (ICMD). The characteristics of physical files and the methods of defining and accessing logical files are described.

Unless otherwise noted, diskette is compatible with accepted industry standards for labels, physical file directories, available space accounting, and error recording.

A diskette unit is a single-user device that may contain from 0 to 15 physical files. Only one physical file on a diskette may be open at a time. Selection of a diskette unit and assignment of a physical file on that unit depend on the settings of the SERIALNO, NEWFILE, FILESECTION, KIND, and FILENAME file attributes. When the logical file is closed, the type of close (for example, LOCK, CRUNCH, or PURGE) determines subsequent selection of a physical file as well as the assignment status of the unit.

The data on a diskette can have physical holes (deleted sectors) while the logical data is sequentially contiguous. Because of this physical structure, access to the logical file is by sequential access only.

The record format is always fixed-length (where BLOCKSTRUCTURE is FIXED or FILETYPE is 0) that can be blocked or unblocked. A diskette file is read and written with one logical record per physical sector. The number of characters per record is defined by the MAXRECSIZE file attribute, which must be greater than or equal to 2 bytes and less than or equal to 128 bytes.

Direct I/O and the update form of logical I/O are not allowed on diskette files, nor does family substitution apply to these files.

The FILENAME attribute contains the physical file name, which is a single-level identifier of eight or less characters, beginning with a letter. The FILENAME attribute is not truncated when searching for a file title on a diskette; if the FILENAME is too long, the file is not found. If a FILENAME that is too long is specified when creating a file, a fatal error occurs.

A permanent physical file can be expanded if all three of the following conditions are met:

a. The diskette is not read-only.

b. The write protect field in the HDR1 label is set to blank.

c. Adequate space is available as defined by the EXTENT value of the physical file.

A new file may be created on a nonscratch diskette, provided that sufficient space is available to allocate the physical file extent. The defined extent of a physical file is established by the EXTENT attribute when the physical file is created. The entire extent is preallocated at this time, but a last record indicator is used to specify the actual end of information in the physical file. If the EXTENT attribute has a value of 0, then all the available space on the diskette is taken as the extent of this physical file. If the EXTENT value is greater than 0, only the specified amount of space is allocated. The EXTENT attribute is defined in units of sectors. For more information, refer to the description of the EXTENT attribute under "General File Attributes".

The diskette has 73 tracks available for user physical files, each containing 26 sectors. A physical sector contains 128 bytes. EXTENT values greater than 26*73 (1898) sectors produce multivolume physical files. The maximum number of volumes in such a physical file is 99.

When a new diskette file is opened and its EXTENT is allocated, the physical file label is updated to include the physical file name, the beginning and end of its extent, the creation date, and the expiration date. If the value of the file's SAVEFACTOR attribute is 0, a temporary physical file is created, which is purged if the file is closed with release. If the SAVEFACTOR value is nonzero, a permanent physical file is created; it is removed only if the file is closed with purge.

Once a diskette has been assigned to a task, the file close type determines subsequent assignment of the unit as follows:

File Close Type	Subsequent Assignment
LOCK or CRUNCH	Temporary files are made permanent. If SAVEFACTOR is 0, the value is changed to 7. The unit is released.
PURGE	The physical file is marked as null and is renamed DATAxx (the null physical file name for that HDR1 position). The unit is released.
REWIND	The file is positioned at the beginning of the physical file.

If the value of SAVEFACTOR is 0 and the close type is not LOCK or CRUNCH, the physical file is removed when the logical file is closed.

See also
All Devices 45
General File Attributes 95

HOST CONTROL FILES

A BNA I/O-station link provides a powerful and versatile connection between two systems linked by an Inter-System Control (ISC). At the level of the user program, port files provide a simple and reliable way of achieving program-to-program communication between BNA ISC-linked hosts.

For those installations where the convenience and extra features of a BNA link are outweighed by efficiency considerations, Direct HC files provide an alternative means of using an ISC link between two or more large systems for simple, high-speed data transfers. Use of Direct HC files is a user program alternative to BNA I/O-station control of a ISC connection to a host. By means of either two Direct files (one with MYUSE specified IN and one with MYUSE specified OUT) or one bidirectional Direct file (with MYUSE specified IO, allowed only on DLP-based systems), a user program dedicates an ISC connection to its own exclusive use. Two or more such programs, controlling HCs attached to the same hub, communicate directly across the ISC link.

The KIND mnemonic associated with the Host Control portion of an ISC is HC. Only Direct files are permitted to have a KIND specification of HC.

The user of an HC file must assume full responsibility for flow control of the data being transferred across the ISC link and for recovery from I/O errors occurring on Direct reads or writes. Direct HC file use of ISCs is possible regardless of the configuration of the hosts. Changes to the named HCs and ISCs of a system can be made without a reconfiguration Halt/Load.

Opening HC Files

When a Direct I/O HC file is opened, the MCP attempts to find an available, labeled HC whose label matches the file's FILENAME and whose mode matches the file's MYUSE. On B 6800, B 7700, and B 7800 systems, MYUSE must be specified, at least implicitly, because the default value of MYUSE for Direct I/O HC Files is IO. (Refer to the description of the MYUSE attribute under "General File Attributes".) If exactly one match is found, the HC is assigned to the file; otherwise, a "DUP FILE" or "NO FILE" RSVP results. Assuming that bidirectional communication between programs is desired, a user program opens an IN/OUT pair of HC files. (In the case of an HC-2 whose MODE is IO, the "pair" just happens to be one file.) The file OPEN request initializes the Access Mask Register (AMR) for the selected Host Control. The AMR controls directionality of data transfer (input only, output only, or input and output). The AMR is set to "CLOSED" when the file is closed.

I/O Operations

Data transfer across an ISC is accomplished when the hub hardware "pairs up" an HC write (always directed to a specified target hubindex) with an HC read at the target HC. An HC read is indiscriminate, accepting input from any writing HC on its hub. HC I/O operations must always specify an even number of characters; any even length between 2 and the hardware maximum of 65534 is allowed.

For HC writes, the directionality requirement is satisfied by requiring that the WRITEPARTNER buffer attribute always be valid. If the WRITEPARTNER for the assigned HC unit is not specified in the HUBMAP, an HC write attempt for which the direct buffer attribute WRITEPARTNER is not specified will fail with a "NO WRITE PARTNER SPECIFICATION" error. For HC reads, the sending hubindex must be obtained from the IORESULT buffer attribute if input from more than one sender is allowed. If the READPARTNER buffer attribute is valid for an HC read, input from other than the desired writer hubindex will be indicated in the IOERRORTYPE result. Refer to the description of the IOERRORTYPE buffer attribute under "Direct I/O" for details.

In cases in which a pair of HC file programs is used simply to communicate between the same two systems on a regular basis, an installation will normally specify WRITEPARTNERs and READPARTNERs at the HUBMAP level. If an installation preselects the physical paths in this way, the programs using HC files themselves should not reference READPARTNER or WRITEPARTNER at the direct buffer level; they need only check the results of I/O operations by interrogating IOERRORTYPE. The only other direct buffer attribute normally used for HC I/O is IOCHARACTERS, which has the count of data characters received by an HC read completion. The IOMASK and IOCW attributes are ignored for HC operations.

Error Reporting

The IORESULT Direct I/O buffer attribute returns the hardware logical result descriptor (LRD) in its entirety after an HC operation. (For details, refer to the description of the IORESULT attribute under "Direct I/O".)

An HC read operation may complete successfully (IORESULT returns FALSE) yet return "INPUT FROM WRONG HUBINDEX" via the IOERRORTYPE buffer attribute. When an HC read has an intended READPARTNER specified, an "INPUT FROM WRONG HUBINDEX" result is reported by IOERRORTYPE even if "PARITY" or "LONG BLOCK" errors were also detected. The occurrence of such other errors can be detected by checking IORESULT.

The IOERRORTYPE values "WRITE TIMEOUT" (18) and "WRITE ACCESS DENIED" (6) may be returned when an HC write is performed. The IOERRORTYPE value "LONG BLOCK" (9) may be returned from an HC read. Refer to the description of IOERRORTYPE under "Direct I/O" for explanations of these error values.

A program waiting for an HC write to complete waits a maximum of 30 seconds for the operation to complete. A program that waits on an HC read completion may be suspended indefinitely; a multiple wait on time and the buffer event is usually more appropriate.

On B 6800, B 7700, and B 7800 systems, patience is required regarding active HC read operations. When an HC-1 has a read request active but is awaiting a write with which it can be paired, it can take up to 30 seconds before any of the following actions completes:

1. the termination of the program using the HC

2. the clearing of the unit

3. the programmatic cancelation of the read by setting IOCANCEL to TRUE

See also

FILENAME. 188
MYUSE 241
IOCANCEL. 350
IOCHARACTERS. 351
IOERRORTYPE 355
IORESULT. 361
READPARTNER 364
WRITEPARTNER. 365
Direct I/O. 343

IMAGE PRINTER FILES

The KIND value associated with line printer is VSID.

In addition to the attributes listed under "All Devices", BACKUPKIND, FORMID, and PRINTDISPOSITION are meaningful for image printer files. For those files backed up to disk the following attributes are also meaningful:

AFTER	SAVEBACKUPFILE
BANNER	SECURITYTYPE
DESTINATION	SECURITYUSE
NOTE	SERIALNO
PRINTCHARGE	TRANSFORM
PRINTCOPIES	USERBACKUPNAME

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

A file of KIND = VSID is never opened, closed, read, or written to by a user program. It is used only as a parameter by various library procedures when they create pages for an image printer. File attributes can be set by the user to specify whether the file is spooled (using the PRINTDISPOSITION attribute), and if so, when and how to print the resulting backup file.

See also

All Devices 45
Backup Files. 46
General File Attributes 95

LINE PRINTER FILES

The KIND value associated with line printer is PRINTER.

The following attributes, in addition to those listed under "All Devices" are meaningful for line printer files:

BACKUPKIND	PAGESIZE
CARRIAGECONTROL	PRINTDISPOSITION
FORMID	PRINTERCONTROL
LINENUM	PRINTERKIND
PAGE	TRAINID

For those files backed up to disk the following attributes are meaningful:

AFTER	SAVEBACKUPFILE
BANNER	SERIALNO
DESTINATION	SECURITYUSE
FORMID	SECURITYTYPE
NOTE	TRAINID
PRINTCHARGE	TRANSFORM
PRINTCOPIES	USERBACKUPNAME
PRINTDISPOSITION	TRIMBLANKS

For those files backed up to tape, the DENSITY, PARITY, PROTECTION, SERIALNO, and TRIMBLANKS attributes are also meaningful.

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

The I/O Subsystem recognizes both EBCDIC and BCL drum printers and various models of train printers. The EXTMODE attribute can be used to specify a desired type of drum printer, and the TRAINID attribute can be used to select a train printer. The I/O Subsystem adjusts to whichever type is assigned and takes 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 for special forms or other information related to the file.

Printer files can be declared as blocked files. The most extreme case of blocking (where MAXRECSIZE is 1, BLOCKSIZE is 132, and UNITS is 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 LINENUM, PAGE, and PAGESIZE attributes allow the description of a logical printer page, and the I/O Subsystem performs the line and page counting.

Line printer carriage control is an integral part of the I/O statements of the various languages. The CARRIAGECONTROL attribute allows the use of data formats that include the carriage control in the first character of the record.

See also
All Devices 45
General File Attributes 95

MAGNETIC TAPE FILES

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

Mnemonic	Meaning
TAPE	Any tape
TAPE7	7-track NRZ tape
TAPE9	9-track NRZ tape
TAPEPE	9-track phase-encoded (PE) or group-coded recording (GCR) tape

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

```
BACKUPKIND = TAPEPE
```

The following attributes, in addition to those listed under "All Devices", are meaningful for tape files:

CREATIONDATE	GENERATION	SAVEFACTOR
CYCLE	IOINERROR	*SERIALNO
*DENSITY	LABELKIND	TAPEREELRECORD
DIRECTION	*PARITY	USECATALOG
EOF	PROTECTION	VERSION
FILESECTION	RECORDINERROR	

Those attributes marked with an asterisk (*) are also meaningful for backup tape.

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

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. (If a user program does not select a density, a default that is compatible with the unit is selected by the I/O Subsystem.)

Magnetic tape files can be either labeled or unlabeled, as indicated by the LABEL attribute. Both ANSI (USASI) standard tape labels and B 5500 and B 3500 system labels are recognized by the I/O 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 FILENAME (or TITLE) are used in the external name of a physical tape file. ANSI standards require that labeled multifile tapes have two identifiers in each file's external name and that the MFID be the same for all the files on the tape. For example, if the file name is "A/B/C", then the MFID is "A" and the FID is "C". Consequently, the name stored in the tape label is "A/C".

Labeled tapes have a serial number that is initially assigned to the tape using the SN (Serial Number) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of the SN command.) This serial number can be interrogated later by means of the SERIALNO attribute. The label information also contains the CYCLE and VERSION attributes that can be used as part of the file identification for file assignment.

The SAVEFACTOR attribute can be used to save a tape file and to determine how long it is to remain saved before being purged.

NOTE

Because SAVEFACTOR is 0 by default, tapes will appear to come up scratch by default when creating them. When an expired tape is mounted with a write ring, it is identified as scratch but not purged. Thus, removing the write ring will make the data visible.

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

When a tape file that is 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 block size (that is, the physical record size) for tape files is six words (36 EBCDIC bytes). Blocks are padded with zeros, if necessary. Users are cautioned that this padding will occur every time the last block in a tape file is shorter than six words. Writing such a block may cause binary zeros to be returned to the program when the file

is re-read and then may cause a record count error to be displayed. This error does not occur if all all records are at least six words (36 EBCDIC bytes) long (for example, if MAXRECSIZE is at least this long and the file has fixed-length records).

Multifile Tapes

Multifile tapes require two-level tape titles. The first-level name must match for all files written to a logical tape, regardless of the number of physical volumes used. No two files on such a tape can have the same second-level name. Multiple files with the same name will be found only if their genealogy (that is, the values of the CYCLE and VERSION attributes) are different and the CYCLE and VERSION attributes are explicitly specified for the file that is to be opened. In other cases, it is not possible to predict which file will be found.

The file search routines make extensive use of the requirements described above in order to reduce the time taken to find a tape file. Files on tapes that do not meet these criteria might not be found without operator intervention.

The following algorithm is used to search for files on multifile tapes.

First, all units associated with the process attempting to open the file are searched. Then, if the file is on one of these units, it is assigned. If not, all other tapes whose first-level names match with that of the required file are searched. When a unit is searched unsuccessfully, it is not locked, and the serial number of the searched tape is kept by the search routines to indicate that the tape has already been searched. Operator intervention (using the ODT command IL) is required only when the operator wishes the search routines to check a tape whose serial number is identical to that of a tape already searched.

Closing Tape Files

The CLOSE WITH LOCK version of the CLOSE statement causes the unit to which the file is assigned to be made unavailable for assignment to another file. On tape drives, the normal method used to enforce this restriction is to unload the tape.

On multiplexor (MPX) machines, some Phase Encoded (PE) tape drives do not unload tapes when the software requests it. The software marks as LOCKED all drives that might not be unloaded (that is, all non-GCR drives on multiplexor machines).

ODT FILES

The value of the KIND attribute associated with ODT files is SPO.

A logical file can be assigned to an operator display terminal (ODT), thus allowing terminal communications to a program without using data communications. The external mode (EXTMODE) of an ODT file is EBCDIC. An output ODT file (with MYUSE equal to OUT) is assigned to a scratch unit, preferably to the originating unit, if the program is entered from an ODT. An input or input/output ODT file (with MYUSE equal to IN or IO) is assigned to a labeled unit. An ODT can be labeled by entering the following:

```
LABEL <file name> <etx>
```

where <etx> is the ETX character represented by the hexadecimal value 03. When the ODT file is opened, all input to the file must be preceded by the Group Separator (<GS> or <delta>) character represented by hexadecimal 1D. An input record contains all the characters between the <GS> and the <etx>. If the record length is less than the value of MAXRECSIZE, the record is padded on the right with blanks.

The end of the input file is indicated by entering "<GS>?END<etx>". When the record is received, the ODT file takes end-of-file action. An ODT again becomes scratch when a labeled file assigned to it is closed. The CL (CLear) ODT command can also be used to unlabel an ODT. (Refer to the Operator Display Terminal (ODT) Manual for a description of this command.) Because of the nature of the input records, the most natural way to describe the logical file is with a FRAMESIZE of 8 and a BLOCKSTRUCTURE of EXTERNAL. (Refer to the descriptions of FRAMESIZE and BLOCKSTRUCTURE under "General File Attributes"). This file is a character-oriented, variable-length ODT 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 the exact number of characters in the record in the logical I/O result descriptor's size field ([47:20]), as well as in the CURRENTRECORD attribute. Likewise, on output, only the number of characters in the record (as described in the write statement's size parameter) is transferred to the ODT.

See also
All Devices 45
General File Attributes 95

PAPER TAPE FILES

The values of the KIND attribute for paper tape files are PAPERPUNCH and PAPERREADER.

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

Paper tape files have a form of volume switch that is similar to tape files. When the trailer label is sensed, an input volume switch is performed for paper tape reader files that the system operator can respond to through the UL (UnLabeled), FR (Final Reel), or DS (DiScontinue) ODT commands. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.) When "low tape" is sensed for a paper tape punch file, an output volume switch is performed and the program continues when a scratch paper tape punch is made available.

See also
All Devices 45
General File Attributes 95

Reverse and Backspace Operations

With a paper tape punch, there is no reverse positioning, just as there is none for a card punch. Several forms of paper tape readers are used with the B 7000/B 6000 Series systems. 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 that is not equipped with the reverse read option, the MCP is provided with a run-time option that 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. (For further information about the NORVRSPAPERTAPE run-time option, refer to the description of the OP (OPtions) ODT command in the Operator Display Terminal (ODT) Reference Manual.)

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.

Several models have special wiring blocks for placement of channels on the tape. Plugboard jumper connections are shown in Figures 3-1 and 3-2. Use of various INTMODE and EXTMODE combinations on equipment without this wiring block results, in some cases, in track transposition.

Input Operations

Figure 3-3, "File Attribute Settings - Input Operations", shows 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 3-1 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 Figure 3-4, "File Attribute Settings - Output Operations". Figure 3-2 gives plugboard configurations for paper tape punch output.

Figure 3-1. Plugboard Configurations for Paper Tape Read Operations

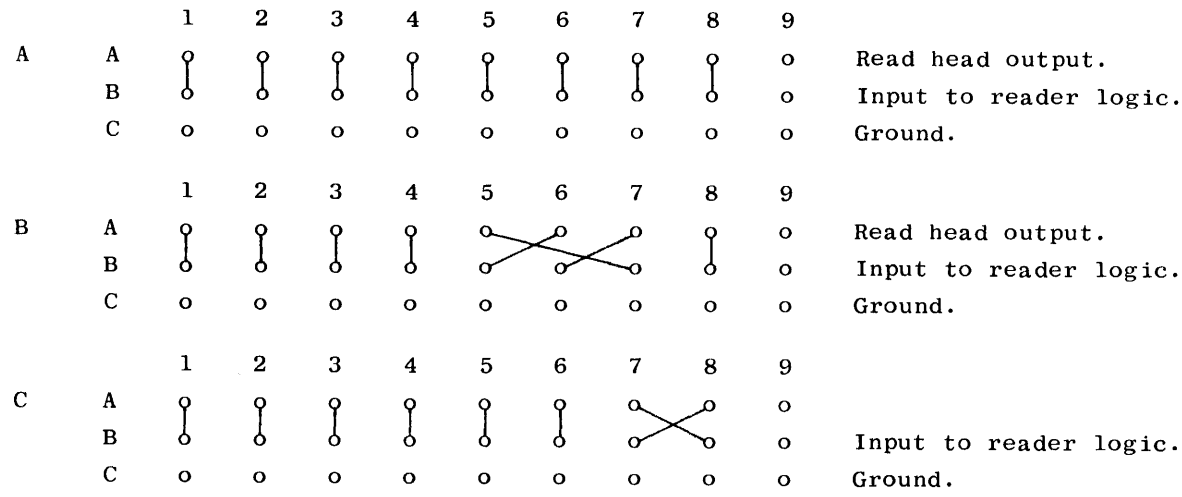


Figure 3-2. Plugboard Configurations for Paper Tape Punch Operations

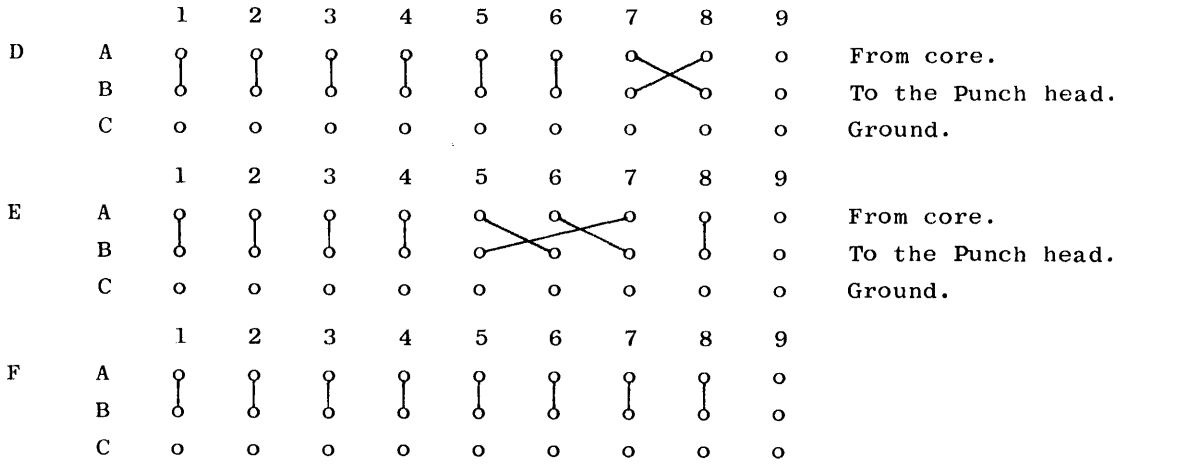


Figure 3-3. File Attribute Settings - Input Operations

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. (Refer to the B 6700 Hardware Handbook for INTERNAL and EXTERNAL BCL tape codes.) Parity errors are sensed. The leader and trailer contain 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

Figure 3-3. (Continued) File Attribute Settings - Input Operations

INTMODE	EXTMODE	PARITY	ACTION	PLUGBOARD
BCL (DISPLAY-1)	BCL (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 TRANSLATE attribute. If TRANSLATE is set to value DEFAULTTRANS (0) at the time the file is opened, then a 7-bit binary read is performed, with the 8-bit being taken as parity. This parity is odd. If TRANSLATE set to value FULLTRANS (1) at the time the file is opened, 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

Figure 3-4. 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

Figure 3-4. (Continued) File Attribute Settings - Output Operations

INTMODE	EXTMODE	PARITY	ACTION	PLUGBOARD
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 TRANSLATE at file open, one of the following two occurs: If TRANSLATE is set to DEFAULTTRANS (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 set to FULLTRANS (1), then the EBCDIC core image is first translated by means of the software EBCDICTOASCII table, and then punched using an 8-bit binary data transfer. The leader and trailer contain all 1's.	D

See also

PORT FILES

Port files differ from other files in that I/O operations and file assignment do not occur between a logical file and a physical device but between correspondent port-subfiles of two or more logical files. A port-subfile is the endpoint of a two-way, point-to-point, logical communication path between two associated programs. In order to establish this association, each program must describe the properties desired for that association. The system compares association descriptions, matches complementary descriptions, and marks the port-subfiles OPENED. This process is called "matching".

The mnemonic value of the KIND attribute for port files is PORT.

Attributes that are meaningful for port files are listed below. Next to each attribute name is one or more of the following notes:

1. Attributes that apply to port files are marked "f" below.

2. Attributes that apply to subfiles are marked "s" below.

3. Attributes that apply only to BNA V2 port files are marked "V2" below.

ACTUALMAXRECORDSIZE (s,V2)
APPLICATIONGROUP (f,V2)
ATTERR (f)
AVAILABLE (f)
AVAILABLEONLY (s,V2)
BLOCKSTRUCTURE (f)
BLOCKEDTIMEOUT (s,V2)
CENSUS (f,s)
CHANGEDSUBFILE (f)
CHANGEEVENT (f,s)
COMPRESSING (s,V2)
COMPRESSION (s)
COMPRESSIONCONTROL (s,V2)
COMPRESSIONREQUESTED (s,V2)
CURRENTRECORD (f)
DIALOGCHECKINTERVAL (s,V2)
DIALOGPRIORITY (s,V2)
DONOTSEARCHNETWORK (s,V2)
EXTMODE (s,V2)
FILENAME (f)
FILESTATE (s)
FRAMESIZE (f)
HOSTNAME (s)
INPUTEVENT (f,s)
INTMODE (f,V2)
INTNAME (f)
LASTSUBFILE (f)
MAXCENSUS (s)
MAXRECSIZE (f,s)
MAXSUBFILES (f)
MYHOST (f)
MYHOSTGROUP (f,V2)
MYHOSTNAME (f)
MYNAME (f)
MYUSE (f)
OLDYOURUSERCODE (f)
OPEN (f)
OUTPUTEVENT (s)
REQUESTEDMAXRECSIZE (f,V2)
SECURITYTYPE (f)
STATE (f)
SUBFILEERROR (s)
TRANSLATE (f,V2)
TRANSLATING (s,V2)
UNITS (f)
YOURHOST (s)
YOURHOSTGROUP (s,V2)
YOURNAME (s)
YOURUSERCODE (s)

In addition, all the attributes listed under "All Devices" are meaningful for port files (with certain exceptions) but not for port-subfiles, unless the list above indicates otherwise. Attributes listed under "All Devices" that are not meaningful for port files are:

BLOCKSIZE	OPTIONAL
CURRENTBLOCK	OUTPUTTABLE
DEPENDENTSPECS	PRESENT
FILEUSE	RECORD
INPUTTABLE	RESIDENT
LABEL	SIZEMODE
MYUSE	SIZEOFFSET
NEWFILE	SIZE2
NEXTRECORD	

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

Attributes that apply only to the port file as a whole can be interrogated or assigned by not specifying a subfile index. If a subfile index is specified for interrogation or assignment to one of these attributes, an attribute error is generated.

Attributes that apply only to port-subfiles can be assigned a value for a particular port-subfile by providing a subfile index. If a subfile index of 0 is specified, the attribute assignment applies to all port-subfiles in the file. If MAXSUBFILES is 1, the subfile index may be omitted.

BNA V2 allows the value of MAXSUBFILES to be increased while the file is open. When this is done, the newly available subfiles will have attribute values initialized from the subfile attribute-value template maintained by the system for this purpose. The template is initialized to contain the specified default values for subfile attributes. Attribute assignments to all subfiles (that is, assignments with a subfile index value of 0) affect the value in the attribute-value template, but assignments to specific subfiles do not affect template values. However, subfile attribute assignments made within a port file declaration do affect the template values.

For attributes that apply to both the file as a whole and to individual port-subfiles, the port-subfile index has the following significance:

a. When a port-subfile index is specified, attributes can be interrogated for, or assigned to, particular port-subfiles.

b. When a port-subfile index is not specified, attributes can be interrogated for, or assigned to, the file as a whole.

For specific information about each attribute, refer to the relevant attribute descriptions under "General File Attributes".

See also

Opening Port Files

In general, when an OPEN statement is executed for a port-subfile (or an implicit open is done on an I/O operation), the FILESTATE value of the port-subfile changes to OFFERED, and the matching algorithm for the port-subfile searches for a complementary port-subfile. (For more information about complementary and offered subfiles, refer to the description of the FILESTATE attribute under "General File Attributes".)

If a complementary port-subfile is not found, the OPEN attempt is abandoned (if AVAILABLEONLY is TRUE or the OPEN option is AVAILABLE), or the port-subfile remains OFFERED until a complementary port-subfile is offered or the port-subfile is closed. If a complementary port-subfile is found, the port-subfile's FILESTATE changes to OPENED. (For additional information, refer to the description of the FILESTATE attribute under "General File Attributes".)

When a port-subfile is offered, the matching algorithm uses the APPLICATIONGROUP, AVAILABLEONLY, DONOTSEARCHNETWORK, FILENAME, HOSTNAME, INTMODE, KIND, MYHOST, MYHOSTGROUP, MYHOSTNAME, MYNAME, SECURITYTYPE, YOURHOST, YOURHOSTGROUP, YOURNAME, and YOURUSERCODE attributes to find a complementary port-subfile. The values of these attributes are used by the matching algorithm as follows:

APPLICATIONGROUP

The value of the APPLICATIONGROUP attribute must be expressed in the form of a <simple identifier> (1 to 17 characters long) except that the first must be an asterisk (*), a digit, or a letter.

The value of the APPLICATIONGROUP attribute for the offered port-subfile must equal the value of the APPLICATIONGROUP attribute for the complementary port-subfile. A null value for the APPLICATIONGROUP attribute of the offered port-subfile matches only a null value for the APPLICATIONGROUP attribute of the complementary port-subfile. The default value of the APPLICATIONGROUP attribute is null.

This attribute is valid in BNA V2 port files only.

AVAILABLEONLY

Setting AVAILABLEONLY to TRUE instructs the BNA software to attempt to establish a dialog only with those subfiles that are available at the time the open process is begun. If no matching subfile is found immediately, the open process fails. If AVAILABLEONLY is set to FALSE and if no matching subfile is found immediately, the open process is suspended until a matching subfile is found.

DONOTSEARCHNETWORK

If DONOTSEARCHNETWORK = TRUE but the value of YOURHOST is a null string, the matching algorithm for the port-subfile searches the local host for a complementary port-subfile but does not search any other host. If a complementary port-subfile is not found and AVAILABLEONLY is FALSE, the port-subfile remains OFFERED until a complementary port-subfile is offered or the port-subfile is closed. If a complementary port-subfile is not found and AVAILABLEONLY is TRUE, the port-subfile remains closed. If YOURHOST is supplied DONOTSEARCHNETWORK is not applicable.

In order to match, the complementary port-subfile must also be on the local host, or must set YOURHOST to the local host or set DONOTSEARCHNETWORK to FALSE. This attribute is valid on BNA V2 port files only.

FILENAME

The value of the FILENAME attribute must be expressed in the form of a <simple identifier> (1 to 17 characters long; case is not significant) and must not be null. The value of the FILENAME attribute for the offered port-subfile must equal the value of the FILENAME attribute for the complementary port-subfile.

The default value of the FILENAME attribute is the value of the INTNAME attribute.

HOSTNAME

HOSTNAME is the nonpreferred synonym for YOURHOST.

INTMODE

The value of the INTMODE attribute for the offered port-subfile must equal the value of the INTMODE attribute for the complementary port-subfile unless the port-subfile TRANSLATE attribute value allows translation for either of the port-subfiles. (See the explanation of translation negotiation below.)

The default value of the INTMODE attribute is EBCDIC. For BNA Version 1 (BNA V1) INTMODE values are restricted to EBCDIC and SINGLE.

KIND

The value of the KIND attribute for both the offered port-subfile and the complementary port-subfile must be PORT.

MYHOST

The MYHOST attribute contains the name of the host system on which the local process is running. The value of the MYHOST attribute for the offered port-subfile must match the value of the YOURHOST attribute for the complementary port-subfile. The value of the MYHOST attribute is always the name of the local host.

MYHOSTGROUP

The MYHOSTGROUP attribute is a name that identifies the group of hosts to which the local host belongs. The value of the MYHOSTGROUP attribute for the offered port-subfile must match the value of the YOURHOSTGROUP attribute for the complementary port-subfile. The value of the MYHOSTGROUP attribute is always the name of the local host group.

This attribute is valid in BNA V2 port files only.

MYHOSTNAME

MYHOSTNAME is the nonpreferred synonym for MYHOST.

MYNAME

The MYNAME attribute is a string of up to 100 characters that is used for matching complementary port-subfile descriptions. The value of the MYNAME attribute for the complementary port-subfile must equal the value of the YOURNAME attribute for the offered port-subfile. A null value for the MYNAME attribute of the offered port-subfile matches only a null value for the YOURNAME attribute of the complementary port-subfile. The default value is a null string.

SECURITYTYPE

Security checking is performed for each port-subfile as follows:

a. If the SECURITYTYPE attribute values for both the offered port-subfile and the complementary port-subfile are PUBLIC, then security checking is immediately successful.

b. If the SECURITYTYPE attribute value for the offered port-subfile is PRIVATE, the YOURUSERCODE attribute value for the offered port-subfile must equal the usercode of the program offering the complementary file.

c. If the SECURITYTYPE attribute value for the complementary port-subfile is PRIVATE, then the YOURUSERCODE attribute value for the complementary file must equal the usercode of the program offering the offered port-subfile.

The default value for the SECURITYTYPE attribute is PRIVATE.

YOURHOST

The YOURHOST attribute contains the name of the host system on which the complementary program is running. The value of the YOURHOST attribute for the offered port-subfile must be null or equal to the value of the MYHOST attribute for the complementary port-subfile. A null value for the YOURHOST attribute of the offered port-subfile matches any value for the MYHOST attribute of the complementary port-subfile. The default value for the YOURHOST attribute is a null string.

On BNA V1, two port-subfiles that have a null YOURHOST value and reside on separate host systems will not match.

On BNA V2, two port-subfiles will not match if DONOTSEARCHNETWORK is TRUE for both port-subfiles and they are not on the same host.

YOURHOSTGROUP

The YOURHOSTGROUP attribute is a name identifying a group of hosts. The complementary program must be running on one of these hosts. The value of the YOURHOSTGROUP attribute for the offered port-subfile must match the value of the MYHOSTGROUP attribute for the complementary port-subfile. A null value for the YOURHOSTGROUP attribute of the offered port-subfile matches any value for the MYHOSTGROUP attribute of the complementary port-subfile. The default value of the YOURHOSTGROUP attribute is a null string. This attribute is valid in BNA V2 port files only.

YOURNAME

The YOURNAME attribute is a string of up to 100 characters that is used for matching complementary port-subfile descriptions. The value of the YOURNAME attribute for the offered port-subfile must be null or equal to the value of the MYNAME attribute for the complementary port-subfile. A null value for the YOURNAME attribute of the offered port-subfile matches any value for the MYNAME attribute of the complementary port-subfile. The default value for the YOURNAME attribute is null.

YOURUSERCODE

Use of the YOURUSERCODE attribute during security checking is explained under SECURITYTYPE above. The default value for the YOURUSERCODE attribute is the usercode of the process opening the port-subfile.

The OPEN statement allows two optional parameters: the port-subfile that is to be opened and an open option (discussed below). In addition, on BNA V2 port files, two port-subfile attributes (DONOTSEARCHNETWORK and AVAILABLEONLY) directly affect the open process. For details on these attributes see their descriptions under "General File Attributes".

The processing of the OPEN statement causes the matching algorithm to be invoked and causes a result to be returned indicating the success or failure of the open attempt. (Possible values for the OPEN statement result are described in the section under "The Open Statement".)

The port-subfile to be opened is specified using a file identifier and a port-subfile index. If the port-subfile index is 0, all port-subfiles in the port file with a FILESTATE value of CLOSED are opened. If the port-subfile index is greater than 0 and less than or equal to the value of the MAXSUBFILES attribute, then only the specified port-subfile is opened. If no port-subfile index is specified and MAXSUBFILES is greater than 1, the BADSUBFILEINDEX (42) error result is returned. If no port-subfile index is specified and the value of the MAXSUBFILES attribute is 1, then the only port-subfile is opened.

In BNA V1, attempting to open a port file with INTMODE or EXTMODE set to a value other than EBCDIC (the default value) or SINGLE results in the open error message "INVALID TRANSLATION". For BNA V2 port files, ASCII is also a supported value of the INTMODE and EXTMODE attributes.

As part of the open process in BNA V2 port files, the value of the ACTUALMAXRECSIZE attribute to be used in the conversation between the two subfiles is negotiated. The negotiated value is never greater than the smaller of the two REQUESTEDMAXRECSIZE values.

For port files, EXTMODE and TRANSLATING are negotiated during the matching process, based upon the values of INTMODE and TRANSLATE for the two complementary port-subfiles. If the two values of INTMODE are the same, EXTMODE takes the value of INTMODE and TRANSLATING becomes FALSE. If the two values of INTMODE are different, the result depends upon the value of TRANSLATE.

If both port-subfiles specify NOTRANS, no match is made and the OPEN process fails. If either port-subfile specifies USERTRANS or FULLRANS, the match is made. The responsibility for translation is assigned to the subfile with the greatest translation capability. FULLTRANS is assumed to be more capable than USERTRANS, which in turn is assumed to be more capable than NOTRANS.

If one of the two subfiles specifies FULLTRANS, TRANSLATING becomes TRUE for that subfile and FALSE for the complementary subfile, and the value of EXTMODE for that subfile becomes the value of the INTMODE attribute for the complementary subfile. In this case, the system where TRANSLATING is TRUE becomes responsible for the translation. If both subfiles specify FULLTRANS, the assignment of translation responsibility is arbitrary.

If neither subfile specifies FULLTRANS, but one of the subfiles specifies USERTRANS, TRANSLATING becomes FALSE, and the value of EXTMODE for that subfile becomes the value of the INTMODE attribute for the complementary subfile. In this case, the subfile in which INTMODE is not equal to EXTMODE (that is, the subfile that specifies USERTRANS), is responsible for performing user-level translation for any data read or written. If both subfiles specify USERTRANS, the assignment of translation responsibility is arbitrary.

During the open process, a choice must be made between the two MAXRECSIZE values for the two port-subfiles that are involved, if the values are different. In this case, the smaller of the two MAXRECSIZE values is always chosen for use in the conversation between the two port-subfiles.

For languages that allow implicit file opens, port files can be implicitly opened by an I/O operation on the file. If a nonzero subfile-index is specified, then only that subfile will be opened. If a subfile-index of zero is specified, then no implicit open action takes place, and an error is reported. If a subfile-index is not specified and the value of the MAXSUBFILES attribute is 1, then the port-subfile is implicitly opened. If a port-subfile index is not specified and the value of the MAXSUBFILES attribute is greater than 1, and error is reported. Any port file implicitly opened by an I/O operation should be opened just as though the user had, at that point, specified an OPEN using the default option.

The following OPEN options apply to BNA V1 port files:

AVAILABLE

AVAILABLE causes the port-subfile to be matched only to a complementary port-subfile that has already been offered. If an available complementary port-subfile is found, the port-subfile is opened; if not, a NOFILEFOUND (2) result is returned. The port-subfile is not left offered for subsequent matching.

DONTWAIT

The DONTWAIT option indicates that control is returned immediately to the using program, while the open process continues in parallel.

The result of the open can be determined via changes in the FILESTATE attribute and the SUBFILEERROR attribute for the subfile(s) being opened.

In some cases, such as a bad subfile index, an error causes immediate failure of the OPEN operation. In these cases, the error result is returned when the program is resumed.

The DONTWAIT option is the same as the OFFER option.

OFFER

OFFER causes the port-subfile to be offered for matching. The program is resumed without waiting for the port-subfile to be matched.

WAIT

WAIT is the default value, and indicates that the user process will be suspended until the open operation is completed. The subfile is offered for matching, and the program is suspended on a "NO MATCHING PORT" RSVP until a matching subfile is found or the open attempt fails.

If the host specified for YOURSHOST becomes unreachable before the open operation is complete, the program is resumed and the UNREACHABLEHOST (38) result is returned. For more information on these result values, see "AVAILABLE".

The following OPEN options apply to BNA V2 port files:

AVAILABLE

AVAILABLE causes the port-subfile to be matched only to a complementary port-subfile that has already been offered. If an available complementary port-subfile is found, the port-subfile is opened; if not, a NOFILEFOUND (2) result is returned. The port-subfile is not left offered for subsequent matching.

Use of the AVAILABLE option is nonpreferred. This option causes an OPEN WAIT (the default option) to be performed with the AVAILABLEONLY attribute treated as if it were TRUE.

DONTWAIT

The DONTWAIT option indicates that control is returned immediately to the using program, while the open process continues in parallel. The result of the open can be determined via changes in the FILESTATE attribute and the SUBFILEERROR attribute for the subfile(s) being opened.

In some cases, such a bad subfile index, an error causes immediate failure of the OPEN operation. In these cases, the error result is returned when the program is resumed.

If AVAILABLEONLY is FALSE, and the subfile YOURHOST names an unknown or unreachable host, then the FILESTATE attribute will be set to AWAITINGHOST. The subfile will remain in this state until either it is closed by the user or the host becomes reachable (the open then proceeds normally).

If AVAILABLEONLY is TRUE and the host specified by YOURHOST is unreachable, then the OPEN operation fails, the program resumes, and an UNREACHABLEHOST (38) result is returned. For BNA V2 port files, if AVAILABLEONLY is TRUE and no match is found, a NOFILEFOUND result is returned. In both cases, the subfile is not left offered for subsequent matching.

OFFER

OFFER causes the port-subfile to be offered for matching. The program is resumed without waiting for the port-subfile to be matched.

Use of the OFFER option is nonpreferred. This option causes an OPEN DONTWAIT to be performed with the AVAILABLEONLY attribute treated as if it were FALSE.

WAIT

WAIT is the default value, and indicates that the user process will be suspended until the open is complete. The subfile is offered for matching, and the program is suspended on a "NO MATCHING PORT" RSVP until a matching subfile is found or the open attempt fails.

If AVAILABLEONLY is FALSE and the host specified by YOURHOST is unreachable, then the program is suspended on a "NO HOST" RSVP. If AVAILABLEONLY is FALSE and the host becomes unreachable before the open is complete, then the program is resumed and UNREACHABLEHOST (38) is returned. For more information on these result values, see "AVAILABLE".

If AVAILABLEONLY is TRUE and the host specified by YOURHOST is unreachable, then the OPEN operation fails, the program resumes, and an UNREACHABLEHOST (38) result is returned. If AVAILABLEONLY is TRUE and no match is found, the OPEN operation fails, the program resumes, and a NOFILEFOUND (2) result is returned. In both cases, the subfile is not left offered for subsequent matching.

If WAIT, DONTWAIT, or OFFER is specified as the open option and the port-subfile's YOURHOST attribute names an unknown or unreachable host system, the FILESTATE attribute becomes AWAITINGHOST. The port-subfile remains in this state until it is either closed by the user or the host becomes reachable; in this latter case, the open function then proceeds normally. If AVAILABLE is specified as the open option and the host is (or becomes) unknown or unreachable, the result UNREACHABLEHOST (38) is returned.

When the open function is used to open port-subfiles, one of several values is returned. See the "OPEN Statement" for details. When a subfile index of 0 is specified, open results for individual port-subfiles must be obtained through use of the SUBFILEERROR port-subfile attribute.

See also

AVAILABLE 122
FILENAME. 188
FILESTATE 194
HOSTNAME. 208
INTNAME 215
KIND. 218
MAXRECSIZE. 232
MAXSUBFILES 235
MYHOSTNAME. 239
MYNAME. 240
SECURITYTYPE. 285
SUBFILEERROR. 308
The OPEN Statement. 21
YOURNAME. 340
YOURUSERCODE. 341

Port File I/O Operations

I/O operations for port files are done using port-subfiles. By specifying a subfile index, the read or write will apply to a particular port-subfile. If there is only one port-subfile, the subfile index is optional.

SEEK statements are not allowed for port files.

A subfile index of 0 has special meanings for I/O operations. On reads, a subfile index of 0 indicates a nonselective read, which inputs the next message from any port-subfile of the port.

A subfile index of 0 on a write specifies a broadcast write. This means that the write is directed to every port-subfile that can be written to.

If an error occurs on a nonselective read or broadcast write, the error result is returned in the usual manner through the result descriptor and the STATE attribute. (Refer to the description of the STATE attribute under "General File Attributes".)

I/O statements may include a DONTWAIT specification (for example, "READ (F[DONTWAIT]...)" in ALGOL), which allows a program to continue executing if the I/O operation cannot be completed immediately. Normally, if a read operation is performed on a port-subfile with CENSUS equal to 0, the program is suspended until data is available. If a write operation is performed on a port-subfile with no buffers available, the program is suspended until a buffer is available. However, when DONTWAIT has been specified and a read or write operation 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 result descriptor and the STATE attribute are set to indicate this occurrence.

A program attempting an I/O operation is terminated if either of the following conditions is true:

a. A port-subfile index less than 0 or greater than MAXSUBFILES is specified.

b. No port-subfile index is specified, and MAXSUBFILES is greater than 1.

Some languages can return a result from an I/O operation. Instead of terminating the user program, these languages may return a BADSUBFILEINDEX (42) error result.

When the value of the FILESTATE attribute is DEACTIVATIONPENDING, all subsequent write operations return an end-of-file indication. Read operations continue to operate normally as long as messages are queued for input. When no more messages are enqueued for input, the value of the FILESTATE attribute changes from DEACTIVATIONPENDING to DEACTIVATED, and all subsequent read operations return an end-of-file indication. If a program is suspended to wait for an I/O operation to complete and the corresponding port-subfile closes (in other words, the FILESTATE value changes to DEACTIVATIONPENDING or DEACTIVATED), the program resumes and end-of-file is returned, if appropriate according to the rules above.

I/O statement options such as the following have no significance for port files:

a. Skip-to-channel

b. Skip lines

c. Stacker selection

When the value of the FILESTATE attribute changes to BLOCKED, I/O operations continue to function normally. This action enables programs to maintain dialogs through temporary conditions in which the host is not reachable. However, the system may quickly run out of buffer space, and, if it does, write operations will cause the program to be suspended unless DONTWAIT is specified. Because there will be no incoming messages during this time period, read operations cause the program to be suspended after all queued input is exhausted, unless DONTWAIT is specified.

When the value of the FILESTATE attribute is AWAITINGHOST or OFFERED, all I/O operations return an end-of-file indication.

The quantity of data (the length of the records) that can be transferred during an I/O operation on a port file depends on some or all of the following:

a. The quantity specified in the I/O statement

b. The MAXRECSIZE of the port-subfile (ACTUALMAXRECSIZE for BNA V2 port files)

c. For read operations, the quantity of the data actually received (This is a factor only if the value of the BLOCKSTRUCTURE attribute is EXTERNAL.)

d. The size of the user program's buffer

For write operations, the quantity of data sent is determined by the minimum of the following:

a. The value of the MAXRECSIZE attribute for the port-subfile (ACTUALMAXRECSIZE for BNA V2 port files)

b. The quantity indicated in the WRITE statement

c. The size of the user program's buffer

During write operations, truncation occurs and the user program receives no indication that it has occurred, under the following conditions:

a. The user attempts to write data that is larger than MAXRECSIZE. In this case, the message is truncated to MAXRECSIZE.

b. The user attempts to write data that is larger than ACTUALMAXRECSIZE. In this case, the message is truncated to ACTUALMAXRECSIZE.

c. The quantity specified in the write statement is smaller than MAXRECSIZE (or ACTUALMAXRECSIZE for BNA V2 port files) but larger than the user program's buffer. In this case, the message is truncated to the size of the user program's buffer.

The BLOCKSTRUCTURE attribute has no significance for WRITE operations.

For read operations when the value of the BLOCKSTRUCTURE attribute is FIXED, the quantity of data delivered to the user's buffer is determined by the minimum of the following:

a. The quantity indicated in the read statement

b. The value of the MAXRECSIZE (or ACTUALMAXRECSIZE) attribute for the port-subfile

c. The size of the user program's buffer

If BLOCKSTRUCTURE is FIXED and the message received is smaller than the size of the user program's buffer, the buffer is blank-filled. If the message received is larger than this size, the message is truncated and the user receives no indication of the truncation. In either case, the value of the CURRENTRECORD attribute always equals the value of the MAXRECSIZE (or ACTUALMAXRECSIZE) attribute.

For read operations when the value of the BLOCKSTRUCTURE attribute is EXTERNAL, the length of the data delivered to the user's buffer is determined by the minimum of the following:

a. The length indicated in the read statement

b. The value of the MAXRECSIZE (or ACTUALMAXRECSIZE) attribute

c. The size of the user's buffer

d. The size of the actual message

If the size of the actual message is larger than the length indicated in the read statement, larger than the value of MAXRECSIZE (or ACTUALMAXRECSIZE), or larger than the size of the user's buffer, then truncation takes place and the user receives no indication of the truncation. If BLOCKSTRUCTURE is FIXED, buffers are not blank-filled. Interrogating the CURRENTRECORD attribute reveals the length of the actual message most recently read or written (excluding blanks added as fill).

BLOCKSTRUCTURE = VARIABLE is invalid for port files if SIZEVISIBLE is FALSE. When BLOCKSTRUCTURE = VARIABLE and SIZEVISIBLE = FALSE, the buffers are not blank-filled.

The compression attributes (COMPRESSING, COMPRESSION, COMPRESSIONCONTROL, and COMPRESSIONREQUESTED) allow the user to request that the system compress data passed between subfiles on different BNA hosts. Compression is generally useful only when the link between the two hosts is relatively slow. This decision can be left to BNA by setting COMPRESSIONCONTROL to SYSTEM.

See also
BLOCKSTRUCTURE. 135
CURRENTRECORD 158
FILESTATE 194
MAXRECSIZE. 232
STATE 299

Closing Port Files

Port-subfiles are closed explicitly by using the CLOSE statement or closed implicitly when the block in which the file was declared is exited. When closing port-subfiles, the CLOSE statement requires one parameter: the port-subfile to be closed. In addition, the CLOSE statement allows a second, optional parameter called a "close option".

If no close option is specified, the program is suspended until the specified port-subfile is closed. For time-critical programs, the close option DONTWAIT is provided. If DONTWAIT is not specified, the program is suspended while the actual close takes place, which could take a significant amount of time. If DONTWAIT is specified, control returns immediately to the program and the process of actually closing the file takes place in parallel with the execution of the program. The program can detect when the close is complete by monitoring changes in the value of the port-subfile's FILESTATE attribute.

All other close options (PURGE and LOCK, for example) are ignored.

If the port-subfile index specified in a close statement is 0, all open port-subfiles are closed. When this "close all" facility is used, results of the close operation can be obtained by interrogating the SUBFILEERROR port-subfile attribute. If the port-subfile index is greater than 0 but not greater than MAXSUBFILES, only the specified port-subfile is closed. If no port-subfile index is specified and the value of the MAXSUBFILES attribute is greater than 1, a BADSUBFILEINDEX error result is returned. If the value of the MAXSUBFILES attribute is 1, then the only port-subfile is closed.

The close statement can also be used as a function. If the CLOSE statement is used as a function, one of the following values is returned when port-subfiles are closed.

OK (1)
: The close was successful or (if the CLOSE option is DONTWAIT) successfully initiated.

FILENOTOPEN (30)
: The FILESTATE value of the port-subfile was already CLOSED or CLOSEPENDING when the close operation was attempted.

DATALOST (32)
: Not all messages were acknowledged by the remote host system. (Some data may not have been successfully delivered to the destination port-subfile.)

BADSUBFILEINDEX (42)
: The port-subfile index specified as a parameter was less than 0 or greater than the value of the MAXSUBFILES attribute, or no port-subfile index was specified and MAXSUBFILES was greater than 1.

CLOSEALLERROR (46)
: The port-subfile index was 0, and an error occurred on one or more port-subfiles during close operations. Refer to SUBFILEERROR for more information on errors for specific subfiles.

See also

FILESTATE 194
MAXSUBFILES 235
SUBFILEERROR. 308

REMOTE FILES

The KIND value of REMOTE is associated with any remote device attached to the system by way of a Network Support Processor (NSP) or Data Communications Processor (DCP) or CP 2000.

The following attributes, in addition to those listed under "All Devices", are meaningful for remote files. Those items marked "f" are valid only for the entire remote file. Those items marked "s" are valid only for a specific station. Those items marked "f,s" can be used either way.

ASSIGNTIME (s)
CENSUS (f)
DISPOSITION (s)
ENABLEINPUT (f,s)
INPUTEVENT (f)
LASTSUBFILE (f)
POPULATION (f)
RECEPTIONS (f,s)
SCREEN (s)
SCREENSIZE (s)
STATIONCOUNT (f)
STATIONLIST (f)
STATIONNAME (s)
STATIONSALLOWED (f)
STATIONSDENIED (f)
TANKING (f)
TIMELIMIT (f)
TRANSMISSIONNO (s)
TRANSMISSIONS (f,s)
TRIMBLANKS (f)
WIDTH (s)

For more information about these attributes, refer to the relevant attribute descriptions under "General File Attributes".

A 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 I/O Subsystem provides for remote files essentially the same degree of device independence that can be achieved by a program dealing with other peripheral files.

Remote files are permanent files in the sense that they are described in the Network Definition Language (NDL/NDLII) and are assigned an external name (FILENAME) in the Network Information File (NIF). The NDL/NDLII allows a remote 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, and the stations 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.

See also
All Devices 45
General File Attributes 95

Message Control Systems

A remote station is controlled by a Message Control System (MCS) program; the MCS for each station is specified in the NDL. The MCS is responsible for logging the user on, handling error situations, assigning the station to logical files, and performing other functions required or desired by the MCS designers. Examples of MCS programs supported by Burroughs are DIAGNOSTICMCS, RJE (for remote job entry), COMS (Communication Management System), GEMCOS, and CANDE (for file editing and task control); other MCS programs may be implemented at user installations. When a program opens a file involving a remote station, the MCS may elect to participate in I/O or not. By participating, the MCS processes 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 programs must agree on a protocol so that the MCS can perform message switching and the program can identify source and destination stations. Without MCS participation, the messages are routed directly between the Data Communications Subsystem and a program by the I/O Subsystem.

Station List

When a remote file is opened, a station list is created. The station list contains the information from the NIF or DATACOMINFO file that is necessary to distinguish the different stations that have been included in the file. The file attributes that are uniquely related to remote files reference information stored and maintained in a station list.

Relative Station Number

When the station list is created, each station in the file is assigned a Relative Station Number (RSN) in the order in which it was added to the station list. The RSN is, in effect, an index into the station list. A program may perform reads and writes without regard to 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 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 valid RSN may also be larger than the value of the STATIONCOUNT attribute, if some stations have been subtracted from the file.

See also
LASTSUBFILE 226
STATIONCOUNT. 302
STATIONLIST 303

Opening Remote Files

When a program opens a remote file, a "FILE OPEN" 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 Communications Subsystem overrules this assignment. A denial is forced if the MCS did not elect participation and either the station has no line assignment or the assignment would result in more than one concurrent input-capable file for the station.

If the MCS postpones assignment, an end-of-file indication is placed in the input and output queues. The file remains open but is not assigned to a physical device.

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 program may open an unrestricted number of remote files, and a station may be assigned to any number of remote files. However, a station can be assigned concurrently to only one remote file that is input-capable, without the station's controlling MCS participating in the I/O operations. The program file's FILEUSE value is considered when opening a remote file. For a remote file, FILEUSE equal to OUT allows output only, FILEUSE equal to IN allows input only, and FILEUSE equal to IO allows both input and output. Thus, if a remote file to be used for both input and output is opened with a write statement (where the source language permits this), its FILEUSE attribute 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 file equation.

See also
FILEUSE 202
STATE 299

Closing Remote Files

If a remote file is closed with retention, as when using the ALGOL statement "CLOSE (FILEID,REWIND)" or the regular close in COBOL, and then reopened, each station in the file will have the same RSN that it had at the time the file was closed. Any changes in the file's STATIONLIST (additions or subtractions) are also retained. If a remote 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 the NIF or in the DATACOMINFO file.

See also
STATIONLIST 303

Remote File I/O Operations

A program can write to a remote file in two different ways. If the LASTSUBFILE attribute is set to an RSN, the output is directed to that specific station in the file. If the value of the LASTSUBFILE attribute is 0 (the default value), the output is 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.

A program receives input from a remote 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. In this case, any subsequent writes that do not explicitly mention a STATION (and where LASTSUBFILE was not changed) write to the same station from which the last input was received. This feature allows reply output to be sent to a corresponding input, without the need for interrogating which station the input came from. A SEEK statement only changes the value of LASTSUBFILE.

A multistation remote 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 FILESTATE attribute or the qualification field of the STATE attribute can be interrogated to find the reason for the end-of-file.

The FILENAME attribute has special semantics for remote files. Refer to the description of the FILENAME attribute under "General File Attributes".

See also

4 GENERAL FILE ATTRIBUTES

This section provides detailed instructions for using file attributes during input/output operations. For those users who are unfamiliar with the I/O Subsystem and file attributes in general, a discussion of logical I/O is provided in the introduction to this section. In addition, the various states of a logical file (open, closed, assigned, and unassigned) are defined.

4.1 INTRODUCTION

The file attribute interface allows a user program to interrogate or request modification of the I/O Subsystem information that defines his file. The information is accessible to a program only 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 as well.

When the logical file is first accessed in a program (by setting or accessing any attribute, by opening the file, or by any I/O statement involving the file), the following actions occur in the order indicated:

1. The logical file is set up, and all attributes are set to their system default values.

2. The attributes mentioned in the file declaration are set to the values specified in the file declaration. (INTNAME and INTMODE are always implicitly placed in the file declaration by the compilers.)

3. If any compile-time file equation exists for the logical file (that is, the value of the INTNAME attribute matches a name referenced in the file equation), then the file-equated attributes are set to the values specified.

4. If any run-time file equation exists for the logical file, which is determined by matching the INTNAME as described in step 3, then the file-equated attributes are set to the values specified.

5. The action that first accessed the logical file is performed. (For example, this action may be a statement to set an attribute.)

When a program specifies the value of a file attribute, it is specifying the value for a logical file. When a physical file is created, some of the attributes used to create it must be kept with the file so that it can be properly used later. Thus, a physical file also has attributes.

The process of opening a logical file involves searching either for a device or a place on a device on which to create the file or for an existing physical file with file attributes compatible with those of the logical file. When such a search is completed successfully, the logical file is said to be assigned to the physical file. Data transfer can occur only when the logical file is assigned to a physical file.

It is possible for a logical file to be open without being assigned to a physical file. This would be the case if the OPTIONAL attribute is TRUE for the logical file and the operator has responded to a "NO FILE" RSVP with the OF (Optional File) ODT command. (For a description of the OF (Optional File) command, refer to the Operator Display Terminal (ODT) Reference Manual.)

Closing a logical file has several purposes. Close is most often used to unassign a logical file from a physical file. It is also used to specify what is to be done with the physical file after the association is broken. For example, the physical file may be discarded, rewound, unloaded, or entered in a directory. For permanent files, close also updates the end-of-file pointer.

It is possible for a logical file to be closed but still assigned to a physical file. This is useful for creating multiple files on a multifile tape, for example. The ALGOL statements "CLOSE(F,*)" and "CLOSE(F,REWIND)" and the COBOL/COBOL74 statement "CLOSE FILEID" leave the logical file assigned to the physical file.

Thus, a logical file can be in one of four basic states: open and assigned, open and unassigned, closed and assigned, and closed and unassigned.

When a logical file is assigned to a physical file, attributes that are not relevant to the physical file are ignored. If the file is opened and any logical attribute is found to be invalid (even if it is correctly set), then the attribute's value is changed by the I/O Subsystem and a nonfatal, run-time attribute error is given. The assignment of a logical file to the physical file can cause the attributes of the logical file to be changed. Changed attributes 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. (Refer to the

descriptions of the FILETYPE and DEPENDENTSPECS attributes.) 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 in WFL. Whenever a conflict exists between the logical and physical framesizes used, the values returned by the attributes are logical framesizes.

Changing the value of a logical file attribute may result in a change in the corresponding physical file attribute. For details, refer to the discussions of the various attributes.

Users who need to know the numbers that the MCP assigns to attribute names, as well as values for the use of compilers and the MCP, should consult SYMBOL/ATTABLEGEN.

Example

The following example demonstrates the use of file attributes. An ALGOL program symbolic and a WFL job deck are shown, where the WFL deck compiles and runs the ALGOL program and uses file equation in the process as well.

ALGOL program symbolic (whose FILENAME on disk is "FILE/PROGRAM"):

```
BEGIN
FILE
    F(KIND=DISK,MAXRECSIZE=90,BLOCKSIZE=180,NEWFILE=TRUE,
      FILENAME="TEST.",FRAMESIZE=8,AREALENGTH=3600);
ARRAY
    A[0:14];

F.NEWFILE := FALSE;
OPEN(F);
REPLACE POINTER(A) BY " " FOR 90;
REPLACE POINTER(A) BY "THIS FILE'S FILENAME IS:  ", F.FILENAME;
WRITE(F,90,A);
CLOSE(F,CRUNCH);
END.
```

WFL job deck:

```
?BEGIN JOB FILE/EXAMPLE;
COMPILE OBJECT/FILE/PROGRAM ALGOL LIBRARY;
  COMPILER FILE CARD(KIND=DISK,FILENAME=FILE/PROGRAM);
    FILE F(AREALENGTH=90720,BLOCKSIZE=180,NEWFILE=TRUE,
           FILENAME=TEST);
RUN OBJECT/FILE/PROGRAM;
  FILE F(BLOCKSIZE=2520,NEWFILE=TRUE,FILENAME=TEST1);
?END JOB
```

After the attribute assignment statement "F.NEWFILE := FALSE" is executed, the following attributes take on the indicated values:

```
KIND       = DISK          FILENAME   = "TEST1"
MAXRECSIZE = 90            FRAMESIZE  = 8
BLOCKSIZE  = 2520          AREALENGTH = 90720
NEWFILE    = FALSE
```

During execution of the OPEN statement, if the permanent physical file with the FILENAME "TEST" does not exist on disk when the assignment to the physical file is attempted, then the program waits on a "NO FILE" RSVP. In other words, the program displays the message "NO FILE TEST1" and waits for a response from either an operator or a programmer, or for the file with the FILENAME "TEST1" to be created on or copied to the appropriate disk device.

If the user responds to the RSVP with the FA response "?FA FILENAME=TEST/FILE" (through CANDE), then all the file attributes listed above will have the same values indicated above, except that the attribute FILENAME will then equal "TEST/FILE". If at that time the physical file with the FILENAME "TEST/FILE" exists on disk, then the OPEN process proceeds normally, assigning the logical file "F" to the physical file "TEST/FILE" on disk, setting up the logical file, and marking the logical file as open.

See also

4.2 ATTRIBUTE CHARACTERISTICS

Each file attribute described in this section lists all or some of the following items directly beneath the attribute heading:

```
                Kinds:
          Interrogate:
               Modify:
                 Type:
                Range:
              Default:
Stored Permanently:
           Parameters:
   BNA Logical I/O:
```

Each item describes a particular characteristic of the file attribute. These items and their possible values are defined in the following paragraphs.

KINDS

"Kinds" indicates the KIND(s) of the physical file for which the attribute has meaning. The possible values for this item may include one or more of the following:

all devices *	port **
disk (DISK and PACK)	port-subfile
diskette	printer
host controller (HC)	punch
pack only (not DISK)	remote
paper tape	tape

* "all devices" means that an attribute is relevant for all KIND values.

** An attribute that is specified for port files is not a port-subfile attribute unless specifically stated. See "Port Files" for a list of port file and port-subfile attributes.

See also
    Port Files. 74

INTERROGATE

"Interrogate" indicates whether or not an attribute can be interrogated (that is, read) and, if so, under what restrictions. The possible values and associated meanings for this item are as follows:

Value	Meaning
never	The attribute can never be interrogated. In other words, it is a write-only attribute.
when closed	The file must be closed in order for the attribute to be interrogated.
when open	The file must be open in order for the attribute to be interrogated.
when assigned	The logical file must be assigned to a physical file in order for the attribute to be interrogated.
when unassigned	The logical file must not be assigned to any physical file in order for the attribute to be interrogated.
anytime	The attribute may be interrogated under any circumstances.

MODIFY

"Modify" indicates whether or not an attribute can be modified and, if so, under what restrictions. The possible values and associated meanings for this item are as follows:

Value	Meaning
never	The attribute can never be modified. In other words, it is a read-only attribute.
when closed	The file must be closed in order for the attribute to be modified.
when open	The file must be open in order for the attribute to be modified.
when assigned	The logical file must be assigned to a physical file in order for the attribute to be modified.
when unassigned	The logical file must not be assigned to any physical file in order for the attribute to be modified.
anytime	The attribute may be modified under any circumstances.

TYPE

"Type" indicates the data type required when using an attribute. The possible values and associated meanings for this item are as follows:

Value	Meaning
Boolean	Boolean-valued
event	event-valued
integer	integer-valued
mnemonic	integer values corresponding to mnemonics associated with the attribute
pointer	pointer-valued *
real	real-valued
translatetable	translatetable-valued (An ALGOL-style translatetable)
word	48 bits of information as described in the explanation of the attribute

* The syntax for modifying and interrogating pointer-valued attributes is language-dependent. For example, some languages require that the value be terminated by a period, some require that it be enclosed in quotation marks, and some require both. Refer to the appropriate language manual.

A value consisting of no characters (that is, a null value for a pointer-valued attribute) is indicated in this manual by the phrase "null string".

RANGE

"Range" indicates a list of possible values associated with the attribute. These values vary according to the "type" of the given attribute.

DEFAULT

"Default" indicates the value that the system uses when the user does not specify a value for the attribute. Depending on the attribute, the possible entry for this item is usually either a value that is a member of the defined range for that attribute or "not applicable". If no default exists for the attribute, the entry for this item is "no default".

STORED PERMANENTLY

"Stored Permanently" indicates whether or not the value of the attribute is stored with the permanent file. (If stored, the value of an attribute can be used again when the file is reopened.) The possible values and associated meanings for this item are as follows:

Value	Meaning
disk	The value is stored in the header of the permanent file if the file is a disk file or pack file.
diskette	The value is stored in the header of the permanent file only if the file is a diskette file.
tape	The value is stored in the label of the permanent file only if the file is a tape file.
disk/tape	The value is stored in the header of the permanent file if the file is a disk file or a tape file.
tape/diskette	The value is stored in the header of the permanent file if the file is a tape file or a diskette file.
pack only	The value is stored in the header of the permanent file only if the file is a pack file.
no	No value is stored regardless of the kind of file.

PARAMETERS

"Parameters" indicates how many parameters are associated with the attribute and whether they are optional or required. If no parameters are associated with the attribute, the value for this item is "none".

BNA HOST SERVICES

"BNA Logical I/O" indicates whether or not the attribute is supported by the Burroughs Network Architecture (BNA) Logical I/O Host Service. The possible values and associated meanings for this item are as follows:

Value	Meaning
supported	The attribute is supported by BNA Logical I/O.
not supported	The attribute is not supported by BNA Logical I/O.
restricted usage	BNA Logical I/O imposes restrictions on the attribute's use.
restricted values	BNA Logical I/O does not support those values indicated as not supported in the attribute description.
restricted usage/values	BNA Logical I/O imposes restrictions on the attribute's use and does not support those values indicated as not supported in the attribute description.

4.3 INDIVIDUAL ATTRIBUTE DESCRIPTIONS

Each attribute description begins with the formatted list of attribute characteristics just defined, followed by a narrative discussion of the attribute's uses. Where appropriate, this discussion includes a list of the various combinations of values, mnemonics, and meanings associated with that attribute.

The individual attribute descriptions are presented in alphabetical order beginning on the following page. These descriptions make up the remainder of this section.

ACTUALMAXRECSIZE

Interrogate: anytime

Type:	integer
Modify:	never
Type:	integer
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported
Default:	see below
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The ACTUALMAXRECSIZE file attribute applies only to BNA V2 port files.

The ACTUALMAXRECSIZE attribute specifies the maximum text size that was negotiated when subfile dialog was initiated. The value of this attribute is equivalent to the value returned when MAXRECSIZE is referenced with a specified subfile index. Note that ACTUALMAXRECSIZE is guaranteed to be no more than the lesser of the two values of REQUESTEDMAXRECSIZE.

If FRAMESIZE is not 48 (or UNITS is CHARACTERS), ACTUALMAXRECSIZE is expressed in INTMODE units; otherwise, it is expressed in words. The default 1920 INTMODE units.

The ACTUALMAXRECSIZE attribute requires a subfile index as a parameter if MAXSUBFILES is greater than 1.

See also

MAXRECSIZE. 232
Port File I/O Operations. 84
REQUESTEDMAXRECSIZE 274
SIZEVISIBLE 296

AFTER

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	string
Default:	null string
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The AFTER attribute can be used to defer printing or punching of a printer/punch backup file until some later time. When set, the string value must conform to the syntax for <starttime spec>, which is defined in the Work Flow Language (WFL) Reference Manual. The <date> portion of the <starttime spec> is optional and, if not stated, is assumed to be the date the print request for the file is received by the Print System.

The file will not be printed before the time and date indicated. However, the request is immediately assigned a print request number. (See the PRINT Statement in the Print System (PrintS/ReprintS) User's Guide.) At the specified time and date, the request becomes eligible for printing.

The default value for this attribute is a null string, which indicates no time restriction. If PRINTDISPOSITION = DIRECT, the AFTER attribute does not apply and thus is ignored.

See also
PRINTDISPOSITION. 263

ALTERDATE

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 99999
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The ALTERDATE attribute returns the date of the close operation corresponding to the last time that a file was altered. A file is considered altered when it is closed if the closing program has written to it.

The value is returned as a decimal integer in the form YYDDD, where YY and DDD represent the year and day, respectively, in Julian form. ALTERDATE is unaffected by library maintenance and is not implemented for Installation-Allocated Disk (IAD) files. Changing a permanent file attribute (for example, FILENAME or SECURITYTYPE) does not cause the file's ALTERDATE to change.

ALTERTIME

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 86400000000
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The ALTERTIME attribute returns the time of day, in microseconds since midnight, that is associated with the ALTERDATE. ALTERTIME is not implemented for Installation-Allocated Disk (IAD) files. ALTERTIME is changed only when ALTERDATE is changed.

APL

Kinds:	disk
Interrogate:	when open
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

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

The value of APL may not be altered by file equation.

For details about how to create an APL code file, refer to the description of the MA (May Access) ODT command in the Operator Display Terminal (ODT) Reference Manual.

APPLICATIONGROUP

Kinds:	port
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	null string
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The APPLICATIONGROUP attribute applies only to BNA V2 port files.

The APPLICATIONGROUP attribute indicates the user community to which this port belongs. This port will only communicate with other port files that have the same value of APPLICATIONGROUP. The value of APPLICATIONGROUP must be expressed as a <simple identifier> (1 to 17 characters long) except that its first character must be an asterisk (*), a digit, or a letter.

Application group names beginning with an asterisk (*) are reserved for use by Burroughs-written software or, in some cases, for user programs communicating with Burroughs-written software.

BNA allows an installation to control the usage of specific APPLICATIONGROUPs by usercode. APPLICATIONGROUP authorization is checked when the port-subfile is opened. When a non-null APPLICATIONGROUP is specified and the usercode of the process attempting to open the port-subfile is not authorized to use the value specified for APPLICATIONGROUP, the OPEN process will fail with an UNAUTHORIZEDFORAPPLICATIONGROUP (62) condition.

The APPLICATION Authorization Network Operator command can be used to control the use of APPLICATIONGROUPs. For more information, see the Burroughs Network Architecture (BNA) Version 2 Installation Guide.

AREAALLOCATED

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk
Parameters:	1 required, 1 optional
BNA Logical I/O:	supported

The AREAALLOCATED attribute indicates whether or not (TRUE or FALSE) 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, then AREAALLOCATED requires the copy number as a second parameter. Area numbers begin at 0; copy numbers begin at 1.

Examples (ALGOL)

```
BOOL := F(AREANO).AREAALLOCATED;          % FOR NONDUPLICATED FILES

BOOL := F(AREANO,COPYNO).AREAALLOCATED;   % FOR DUPLICATED FILES
```

AREALENGTH

Kinds:	disk
Interrogate:	anytime
Modify:	when unassigned
Type:	integer
Range:	0 through 4294967295
Default:	1000 * MAXRECSIZE (rounded)
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

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

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

AREALENGTH may be modified 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 rounded up so that it is divisible by the value of BLOCKSIZE. If AREALENGTH is 0, the default value of AREALENGTH is used. The default value of AREALENGTH is a value close to the value of MAXRECSIZE multiplied by 1000 that is also divisible by BLOCKSIZE. The maximum length that can be allocated 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. For example, if MAXRECSIZE is 15 and AREALENGTH is 45, each area uses two sectors.

AREAS

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 1000
Default:	20
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The value of the AREAS attribute 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 0, the default value of 20 is used when creating a new disk file. The setting of the AREAS attribute is not used when opening an existing disk file or reopening a file closed with retention. The AREAS attribute can be overridden when expanding the file if the FLEXIBLE attribute has been modified to TRUE.

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

AREASIZE

Kinds:	disk
Interrogate:	when unassigned (BNA Logical I/O) anytime (local host)
Modify:	when unassigned
Type:	integer
Range:	0 through 1048575
Default:	1000 (rounded to a block boundary)
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	restricted usage

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

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 rounded up so that it is also a multiple of the number of records per block. If AREASIZE is unspecified or 0 and AREALENGTH is unspecified, the default value is assumed. If AREALENGTH is specified, then AREASIZE is ignored.

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

Using the AREALENGTH attribute is preferable to using the AREASIZE attribute.

ASSIGNTIME

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Range:	0 through 86400
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	not supported

The ASSIGNTIME attribute returns the time (in seconds since midnight) at which the station was assigned to the file. The ASSIGNTIME attribute requires one parameter, the Relative Station Number (RSN).

Example (ALGOL)

```
X := F(RSN).ASSIGNTIME;
```

See also

Relative Station Number 91

ATTERR

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The ATTERR attribute returns TRUE if an error resulted from the last file attribute action. For example, a file may have been opened with incompatible attribute values, or a certain attribute combination may be in the process of deimplementation.

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

Attribute errors related to event-valued attributes are always fatal.

ATTVALUE

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	real
Range:	(see below)
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

Following an attribute error, the ATTVALUE attribute contains a value that may aid in correcting that error. If the file is open and an attempt is made to set an attribute that requires the file to be closed, then ATTVALUE returns a word with all bits on (that is, REAL(NOT FALSE)). This word is a small negative number and prints out as 0. If the attribute in error is a pointer-valued attribute, then ATTVALUE returns the pointer (string descriptor) as an operand (that is, with its tag equal to 0); otherwise, the value to which the attribute was to be set is returned. Refer to the descriptions of the ATTERR and ATTYPE attributes.

ATTYPE

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	mnemonic
Range:	attribute name mnemonics
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The ATTYPE attribute returns the number of the last file attribute that was incorrectly used.

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 appropriate language manuals for syntax and semantics.)

Example

The following ALGOL and COBOL74 statements use ATTYPE and the VALUE function to determine if BLOCKSIZE was the last attribute to be incorrectly used:

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

IF ATTRIBUTE ATTYPE OF FILEID = VALUE BLOCKSIZE
       PERFORM HANDLE-PROBLEM.
```

AVAILABLE

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through 74
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The AVAILABLE attribute attempts to open a file and, when that file cannot be opened, reports the reason for the failure without suspending the program or requiring operator intervention. However, 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 AVAILABLE option is specified and the subfile index is 0. For more information, refer to "Opening Port Files".

AVAILABLE returns the following values:

Value	Meaning
0	The permanent file exists but is not available (the program is locked out). This occurs when another program has the file open exclusively, or this program is attempting to open the file exclusively and another program has the file open.
1	The operation was successful. If DONTWAIT or OFFER was the option specified, this result indicates only that the process was successfully started.
2	The permanent file does not exist or cannot be opened because the logical file and the permanent file are incompatible. For port files, this error is returned if a matching subfile was not found and either AVAILABLEONLY was TRUE and the OPEN option was WAIT, or the OPEN option was AVAILABLE. This is interpreted as a NOFILEFOUND error.

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 are available for a tape file.
12	A required disk or pack is not on-line.
14	The file is not available due to ACCESSCODE protection in the user's directory.
38	For port files,AVAILABLEONLY is TRUE or the open option AVAILABLE was specified and the YOURHOST attribute names an unreachable or unknown host. This value is also returned if the open option WAIT is specified and the host named by the YOURHOST attribute becomes unreachable during the process of opening the port-subfile. This is interpreted as an UNREACHABLEHOST error.
40	The value of the FILESTATE attribute for the specified port-subfile is not CLOSED.
42	The file was a port file with MAXSUBFILES greater than 1 and an open was attempted that did not explicitly specify a subfile index or that specified a value greater than MAXSUBFILES or less than 0. This is interpreted as a BADSUBFILEINDEX error.
44	An unknown error occurred during an attempt to open the file, or the file was a port file and an error occurred in opening at least one of the port's subfiles while trying to execute an OPEN operation with a subfile index of 0. The latter case is interpreted as an OPENALL error.
46	An unknown error occurred during an attempt to close the file or the file was a port file and an error occurred in opening at least one of the port's subfiles while trying to execute a CLOSE operation. The latter case is interpreted as a CLOSEALL error.
48	The file has one or more attributes set to BCL and the system does not support BCL.
52	The logical file cannot be assigned to a physical file because the FILEORGANIZATION is incompatible.

54 A request was made for the use of an unsupported protocol type. This result applies to BNA V2 port files only.

56 An open request resulted in the detection of an error in the involved protocol. This result applies to BNA V2 port files only.

58 The open request failed for lack of resources. This result applies to BNA V2 port files only.

60 The requested open on this subfile failed because the YOURHOST specification is not a legal member of the specified YOURHOSTGROUP. This result applies to BNA V2 port files only.

62 The requested open on this subfile failed because the user is not authorized to use the specified application group. This result applies to BNA V2 port files only.

64 The requested open on this subfile failed because BNA was not running. This result applies to BNA V2 port files only.

66 The requested open on this subfile failed because attributes required for the open process were specified with either illegal syntax or invalid values. This result applies to BNA V2 port files only.

68 The requested open on this subfile failed because the function sought was not available. This result applies to BNA V2 port files only.

70 The requested open on this subfile failed because the INTMODE specified was not supported on the host identified by YOURHOST. This result applies to BNA V2 port files only.

72 The requested open on this subfile failed because the networking feature was not supported on this host. This result option applies to BNA V2 port files only.

74 An error occurred in opening a port-subfile, because the matching process was aborted. This is interpreted as a DISCONNECTEDDURINGOPEN error. as a DISCONNECTEDDURINGOPEN error.

Note that values normally associated with closing a file may be returned during an open operation and values normally associated with opening a file may be returned during a close operation if the KIND is TAPE, DISKETTE, PAPERPUNCH, or PAPERREADER (or the KIND is PRINTER or PUNCH and the BACKUPKIND is TAPE) and the open or close causes a volume switch to occur.

Using an OPEN statement with the AVAILABLE option (or for BNA V2 port files, with the AVAILABLEONLY attribute equal to TRUE) is preferable to using the AVAILABLE attribute. For example, in ALGOL the following syntax can be used:

```
I := OPEN(F,AVAILABLE)
```

See also

Opening Port Files. 76

AVAILABLEONLY

Kinds:	port-subfile
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The AVAILABLEONLY attribute is applicable only to BNA V2 port files.

Setting AVAILABLEONLY to TRUE instructs the BNA software to attempt to establish a dialog only with those subfiles that are available when the open process begins. If no matching subfile is found immediately, the open process fails. If AVAILABLEONLY is set to FALSE and if no matching subfile is found immediately, the open process is suspended until a matching subfile is found.

If MAXSUBFILES is greater than 1, the AVAILABLEONLY attribute requires a subfile index as a parameter.

See also

AVAILABLE 122
MAXSUBFILES 235
Opening Port Files. 76

BACKUPKIND

```
                Kinds: printer/punch
          Interrogate: anytime
               Modify: when closed
                 Type: mnemonic
                Range: (see below)
              Default: DONTCARE
Stored Permanently: no
           Parameters: none
   BNA Logical I/O: supported
```

The BACKUPKIND attribute, when used in conjunction with the KIND attribute, allows specification of printer and punch backup devices.

The allowable mnemonic values for BACKUPKIND are as follows:

Mnemonic Value	Integer Value	Meaning
DONTCARE	0	Use the system default values
DISK	1	Use disk pack or head-per-track disk
PACK	17	Use disk pack or head-per-track disk
TAPE	45	Use any available tape
TAPE7	13	Use 7-track NRZ tape
TAPE9	14	Use 9-track NRZ tape
TAPEPE	15	Use 9-track PE or GCR tape

The value of BACKUPKIND is used when the file is opened to select an output device for printer and punch backup files. When the value is DONTCARE (which means nothing was specified), other system and task options are used to select the output device, which may cause an on-line printer or punch to be selected. Installation management can control the selection of the backup device through the use of the SB (Substitute Backup) ODT command. (For a description of the SB (Substitute Backup) ODT command, refer to the Operator Display Terminal (ODT) Reference Manual.)

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

BANNER

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

When TRUE, the BANNER attribute causes the file printout to be preceded by a banner page or card. In this way, the file is more easily identified in the printed output. When BANNER is FALSE, the banner page is suppressed.

The content of the banner page depends upon the NOTE attribute. If NOTE is a null string, the banner consists of the file title (not INTNAME) in block characters; otherwise, the banner consists of the NOTE string in block characters.

The default value is FALSE.

If PRINTDISPOSITION = DIRECT, the BANNER attribute does not apply and thus is ignored.

See also
NOTE. 248
TITLE 314

BLANK

Kinds:	all devices
Interrogate:	anytime
Modify:	anytime
Type:	mnemonic
Range:	NULL, ZERO
Default:	NULL
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The BLANK attribute for FORTRAN77 programs can be used dynamically to control the interpretation of blanks in numeric input fields. This attribute can take on either the value NULL or the value ZERO. If NULL is specified, then all blank characters in numerically formatted input fields are ignored, except that a field of all blanks has a value of ZERO. If ZERO is specified, then all blanks except leading blanks are treated as zeros.

Example (FORTRAN77)

```
OPEN(5,   BLANK=NULL);

OPEN(5,   BLANK=ZERO);
```

BLOCK

```
                Kinds: all except port and remote
          Interrogate: anytime
               Modify: never
                 Type: integer
Stored Permanently: no
          Parameters: none
   BNA Logical I/O: supported
```

The BLOCK attribute returns the number of the block referenced in the last I/O statement. When the file is closed, or if it is open but no I/O action has been performed upon the file, BLOCK returns -1. BLOCK also returns -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 0.

The BLOCK attribute is not valid for a Direct file whose KIND is not DISK or PACK.

BLOCKEDTIMEOUT

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 1440 minutes
Default:	0
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The BLOCKEDTIMEOUT attribute is applicable only to BNA V2 port files.

The BLOCKEDTIMEOUT attribute indicates the amount of time, in minutes, that the system will allow a file to remain in a BLOCKED FILESTATE before beginning deactivation procedures. The default value is 0, which means that deactivation procedures will never be invoked.

If MAXSUBFILES is greater than 1, the BLOCKEDTIMEOUT attribute requires a subfile index as a parameter.

See also
FILESTATE 194
MAXSUBFILES 235

BLOCKSIZE

Kinds:	all except port files
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 65535
Default:	MAXRECSIZE
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted values

The value of the BLOCKSIZE attribute is the length of a block. If the FRAMESIZE attribute is not 48 or if the UNITS attribute is CHARACTERS, then BLOCKSIZE is given in INTMODE units; otherwise, the value of BLOCKSIZE is returned in words.

The BNA Logical I/O Host Service requires that BLOCKSIZE be less than 65486 characters.

When the file is opened, if FILEORGANIZATION is not RELATIVE and BLOCKSIZE is less than MAXRECSIZE, then BLOCKSIZE is set to MAXRECSIZE; if FILEORGANIZATION is RELATIVE and BLOCKSIZE is less than or equal to MAXRECSIZE, then BLOCKSIZE is set to MAXRECSIZE + 1. For relative files, if a COBOL74 file description contains a "BLOCK CONTAINS n RECORDS" clause, with n being a constant, then the value used for BLOCKSIZE is ((n + FRAMESIZE - 1) DIV FRAMESIZE) + (n * MAXRECSIZE).

Files with BLOCKSTRUCTURE values other than LINKED are blocked only when BLOCKSIZE is greater than MAXRECSIZE.

The default value of BLOCKSIZE depends on both the physical unit (KIND) assigned to the file and the value of the MAXRECSIZE attribute. Refer to the description of the MAXRECSIZE attribute.

Card files and remote files have restricted upper bounds for BLOCKSIZE (and therefore MAXRECSIZE). Card files have upper bounds of 80 characters if they are character-oriented, 10 words if they are BCL, 14 words if they are EBCDIC, or 20 words if they are BINARY. The upper limit for remote files is approximately 500 words (the size of a GETAREA row minus 8). These restrictions are enforced when the file is opened. If a restriction is violated, a run-time attribute error is issued and the BLOCKSIZE value is reduced appropriately. If a Direct remote file attempts to write a buffer that is too long, then the I/O is canceled. (Refer to the descriptions of IOCANCEL, IOERRORTYPE, and IORESULT in this section.)

BLOCKSIZE is ignored for port files.

The BLOCKSIZE value for a diskette file must be an integral multiple of MAXRECSIZE.

BLOCKSTRUCTURE

Kinds:	all devices
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	EXTERNAL, FIXED, LINKED, VARIABLE
Default:	FIXED
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted values

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

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

Mnemonic	Integer Value	Corresponding Value of FILETYPE	Meaning
EXTERNAL	1	3	Variable-length records. Neither the record itself nor the structure of the file contains information about the length of the record; length information must be specified externally in the I/O statement. For unblocked tape files, ODT files, and port files, the I/O Subsystem will determine the actual length of a read and return it through the CURRENTRECORD and STATE attributes.
FIXED	0	0	Blocked or unblocked, fixed-length records.

Mnemonic	Integer Value	Corresponding Value of FILETYPE	Meaning
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 data. The INTMODE of the file is assumed to be SINGLE, and software translation is never attempted.
VARIABLE	2	1	Variable-length records. The record size is contained (in decimal form) in the first four INTMODE characters of the record. If INTMODE is SINGLE, then the record size is stored in the first word of the record in binary. This size includes the length of the size field (for example, for character files, the size is four more than the length of the actual data). The size is expressed in FRAMESIZE units. Depending on the value of the SIZEVISIBLE attribute, the record length field may or may not be included within the record.

FIXED is supported by the BNA Logical I/O Host Service, EXTERNAL is supported only when the file is not blocked, and VARIABLE and LINKED are not supported.

If the value of BLOCKSTRUCTURE has not been specified, the structure of the file is determined by the value of FILETYPE. If FILETYPE is also unspecified, the default value FIXED is used. If BLOCKSTRUCTURE has been specified, the value of BLOCKSTRUCTURE determines the structure of the file, overriding the value of FILETYPE. 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.

Variable-length, blocked records are processed differently than fixed-length, blocked records. If the record to be written plus the offset into the block is greater than BLOCKSIZE, then the current block is written to the physical file. 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), then 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), then 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 size fields (where BLOCKSTRUCTURE is VARIABLE or LINKED) require a value for MINRECSIZE equal to or greater than the length of these fields.

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

If BLOCKSTRUCTURE is interrogated when the file is open and BLOCKSTRUCTURE was specified, the value currently in use for the physical file is returned. If BLOCKSTRUCTURE was never specified and the structure being used with the file corresponds to some value of BLOCKSTRUCTURE, then that value is returned. If there is no corresponding value of BLOCKSTRUCTURE, an attribute error occurs. Note that the value for the physical file does not override the value specified in the logical file unless DEPENDENTSPECS is TRUE.

Files with BLOCKSTRUCTURE equal to LINKED are always blocked. Files with BLOCKSTRUCTURE values other than LINKED are blocked only when BLOCKSIZE is greater than MAXRECSIZE.

Attempting to space to any record or to seek to a record other than record zero on a file whose BLOCKSTRUCTURE is EXTERNAL causes a fatal "ILLEGAL SEEK" error.

All diskette files have fixed-length records; that is, BLOCKSTRUCTURE is FIXED.

BLOCKSTRUCTURE has a special meaning for port files. The BLOCKSTRUCTURE values FIXED and EXTERNAL are legal for port files, while VARIABLE is only allowed if SIZEVISIBLE is FALSE. When KIND is PORT and BLOCKSTRUCTURE is FIXED, the buffer is blank-filled on the right after each read; when KIND is PORT and BLOCKSTRUCTURE is EXTERNAL, only the data received is placed in the buffer. Such blank-filling does not occur after a write in either case. When KIND is PORT, SIZEVISIBLE is FALSE, and BLOCKSTRUCTURE is VARIABLE, no blank-filling occurs. For portfiles, BLOCKSTRUCTURE = VARIABLE with SIZEVISIBLE = FALSE is equivalent to EXTERNAL. The quantity of data actually transferred can

be ascertained by interrogating the CURRENTRECORD attribute, interrogating the STATE attribute, or examining the result of the WRITE statement. Attempting to open a port with an unsupported FILETYPE or an unsupported BLOCKSTRUCTURE, results in an open error (an unknown error occurred during an attempt to open the file).

The value of the SIZEVISIBLE attribute affects the meanings of the MINRECSIZE, MAXRECSIZE, and BLOCKSIZE attributes. When SIZEVISIBLE is FALSE at file creation time, the values of MINRECSIZE and MAXRECSIZE are considered not to include the length of the system-maintained record size field, and are adjusted upward accordingly before being stored in the physical file header. If this adjustment results in MAXRECSIZE being greater than BLOCKSIZE, BLOCKSIZE is also adjusted upward. The values returned to the user when the attributes MINRECSIZE and MAXRECSIZE are interrogated reflect the logical values originally specified before they were adjusted to include the system overhead fields. BLOCKSIZE always returns the actual size of the block being used.

BUFFERS

Kinds:	all devices (except port)
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 63
Default:	1, 2, or 3 (see below)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The BUFFERS attribute indicates the number of buffers used in processing a file. The default value of BUFFERS for disk files with an UPDATEFILE of TRUE is 3, the value for remote files is 1, and the value for all other files is 2. Using more than two buffers adds to the efficiency of a file's I/O operations only in exceptional cases, such as when printers are used on-line or when duplicated disk files are written.

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

Setting BUFFERS to 0 indicates that the default value is to be used.

CARRIAGECONTROL

Kinds:	printer
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	CTLASA, CTL360, STANDARD
Default:	STANDARD
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

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

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

The mnemonic values and meanings associated with CARRIAGECONTROL are as follows:

Mnemonic Value	Integer Value	Meaning
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

Mnemonic Value	Integer Value	Meaning
CTLASA (continued)		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 eight bits should be 48"FA", which is a logical extension of the EBCDIC number set. For channel 11, use 48"FB" and for channel 12, use 48"FC".
CLT360	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 occurs, only carriage control. [2:1] — Ignored. [6:4] — The value of this field is either the 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].
STANDARD	0	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.

CENSUS

Kinds:	remote/port/port-subfile
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through 65535
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	supported

The CENSUS attribute returns the number of messages currently in the input queue of a port file, port-subfile, or remote file. If the file is a remote file, then the value returned is the total number of messages queued for that file.

If the port file is open and no subfile index is given, the value returned is the sum of the messages queued for all subfiles in that file. If a subfile index is given, then the value returned is the number of messages queued for the specified subfile only.

If the file is closed, the value 0 is returned.

CHANGEDSUBFILE

Kinds:	port
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through MAXSUBFILES
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The CHANGEDSUBFILE attribute returns the index of an arbitrary subfile whose CHANGEEVENT is currently HAPPENED. If no subfile CHANGEEVENT attribute equals HAPPENED for the file, then a value of 0 is returned.

CHANGEEVENT

Kinds:	port/port-subfile
Interrogate:	anytime
Modify:	never
Type:	event
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The CHANGEEVENT attribute for a subfile is caused (that is, set to HAPPENED) whenever the value of FILESTATE for the subfile changes; CHANGEEVENT is reset (that is, set to NOT HAPPENED) when the FILESTATE attribute for the 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.

Because CHANGEEVENT is a read-only attribute, it should not be passed as a parameter to any of the following statements: ATTACH, CAUSE, CAUSEANDRESET, DETACH, FIX, FREE, LIBERATE, PROCURE, RESET, SET, and WAITANDRESET.

Attribute errors on event-valued attributes result in the termination of the program.

COMPRESSING

Kinds:	port-subfile
Interrogate:	anytime
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The COMPRESSING attribute is available only on BNA V2 port files.

The COMPRESSING attribute indicates whether or not compression is currently being performed for the subfile dialog. COMPRESSING is set as described under the COMPRESSIONCONTROL attribute.

If MAXSUBFILES is greater than 1, the COMPRESSING attribute requires a subfile index as a parameter.

See also
COMPRESSION 147
COMPRESSIONCONTROL. 148
COMPRESSIONREQUESTED. 149
MAXSUBFILES 235

COMPRESSION

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

When set to TRUE, the COMPRESSION attribute compresses data passed between subfiles. Support for the compression feature is negotiated when the subfile is opened. Modifying the value of the COMPRESSION attribute to TRUE while the subfile is open has an effect only if both hosts involved in the subfile dialog support the compression and COMPRESSIONCONTROL has the value USER. (If compression is not supported, the value of COMPRESSION is FALSE even if it is explicitly modified to TRUE.) Provided that compression is supported, records may be selectively compressed by changing the value of the COMPRESSION attribute.

If MAXSUBFILES is greater than 1, the COMPRESSION attribute requires a subfile index as a parameter.

For BNA V2 port files, COMPRESSION returns the value of COMPRESSING upon interrogation. When modified, the COMPRESSION attribute modifies the value of the COMPRESSIONREQUESTED attribute. Use of COMPRESSING AND COMPRESSIONREQUESTED is preferred.

COMPRESSIONCONTROL

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	mnemonic
Range:	USER, SYSTEM
Default:	USER
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The COMPRESSIONCONTROL attribute is available only in BNA V2 port files.

If compression is supported for the current dialog, the COMPRESSIONCONTROL attribute indicates who (the system, or the user) has control over whether compression is to be performed. When the value of COMPRESSIONCONTROL is SYSTEM, the system selectively sets COMPRESSING based on the perceived benefit of compressing data in the current network state. If both host support compression and COMPRESSIONCONTROL is USER, COMPRESSING is set to the value of COMPRESSIONREQUESTED.

If MAXSUBFILES is greater than 1, the COMPRESSIONCONTROL attribute requires a subfile index as a parameter.

See also

COMPRESSING 146
COMPRESSION 147
COMPRESSIONREQUESTED. 149
MAXSUBFILES 235

COMPRESSIONREQUESTED

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The COMPRESSIONREQUESTED attribute is available only on BNA V2 port files.

The COMPRESSIONREQUESTED attribute indicates whether the user has specified that compression be performed.

Note that support for compression in a subfile dialog is negotiated at dialog initiation. The value of COMPRESSIONREQUESTED only affects COMPRESSING when COMPRESSIONCONTROL is USER and compression is supported for this dialog. COMPRESSIONREQUESTED is set to the user-requested value in any case.

If MAXSUBFILES is greater than 1, the COMPRESSIONREQUESTED attribute requires a subfile index as a parameter.

See also
COMPRESSING 146
COMPRESSION 147
COMPRESSIONCONTROL. 148
MAXSUBFILES 235

COPIES

Kinds:	duplicated disk
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	1 through 15
Default:	2
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The COPIES attribute indicates the number of copies within a duplicated disk file. 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 interrogated when the file is assigned.

See also
Duplicated Files. 373

COPYINERROR

Kinds:	duplicated disk
Interrogate:	when open
Modify:	never
Type:	integer
Range:	0 through COPIES
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

If an error has occurred during the last I/O operation on a duplicated file, then the COPYINERROR attribute returns the copy number of the copy in error. If no error has occurred, then the COPYINERROR attribute returns 0. If COPYINERROR returns a nonzero value, then successive inquiries of COPYINERROR return the copy numbers of additional copies that contained errors on the last I/O. A final value of 0 indicates that no other copies were in error.

Inquiry of COPYINERROR resets bits in the duplicated disk file error mark field ([27:16]) in the STATE attribute. Refer to the description of the STATE attribute.

See also
Duplicated Files. 373

COPYNAME

Kinds:	duplicated disk
Interrogate:	when assigned
Modify:	never
Type:	pointer
Default:	not applicable
Stored Permanently:	disk
Parameters:	1 required
BNA Logical I/O:	supported

The COPYNAME attribute returns the external file name of the specified copy of a duplicated file. This attribute may be interrogated only when the logical file is assigned to a physical disk file. The COPYNAME attribute requires the copy number as a parameter. The null string is returned if the copy number specified is greater than the number of copies.

If a duplicated disk file is created with the external file name AAA, then 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 appear in the directory. Because the file name of any copy may be changed, or the copy itself may be removed, the index assigned to a copy or the number of copies may change for subsequent openings of the file.

For further information regarding duplicated files, refer to "Duplicated Files".

See also
Duplicated Files. 373

CREATIONDATE

Kinds:	disk/tape/diskette
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 99999
Default:	not applicable
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The CREATIONDATE attribute returns the date when a file was first opened for creation. When a file is created, "today's date" is used as the CREATIONDATE for the file.

Each value returned by CREATIONDATE is an integer in the form YYDDD, where YY and DDD represent the year and day, respectively, in Julian form. The values returned by CREATIONDATE are unaffected by library maintenance.

The CREATIONDATE attribute is used in conjunction with the CREATIONTIME attribute. For more information, refer to the CREATIONTIME attribute in this section.

CREATIONTIME

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 86400000000
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The CREATIONTIME attribute returns the time (in microseconds since midnight) that the file was first opened when it was created. The values returned by CREATIONTIME are unaffected by library maintenance. The attribute is not implemented for Installation-Allocated Disk (IAD) files.

The CREATIONTIME attribute is used in conjunction with the CREATIONDATE attribute. For more information, refer to the CREATIONDATE attribute in this section.

CRUNCHED

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The CRUNCHED attribute indicates whether or not a permanent disk file was closed with CRUNCH. When a disk file has been crunched, any space in the last area beyond the last block boundary is returned to the system. 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 only 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.

Examples (ALGOL and COBOL74)

```
CLOSE(FILEID,CRUNCH);

CLOSE FILEID WITH CRUNCH.
```

CURRENTBLOCK

Kinds:	all except port
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through BLOCKSIZE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The CURRENTBLOCK attribute returns the size, in FRAMESIZE units, of the block currently in use. Normally, this value is the same as that of the BLOCKSIZE attribute. The value of CURRENTBLOCK becomes of interest for a tape file when the system encounters a short block. For further information, refer to the description of the STATE attribute.

CURRENTEXTENT

Kinds:	diskette
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through EXTENT
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The CURRENTEXTENT attribute returns the currently allocated portion of the total EXTENT value. Refer to the description of the EXTENT attribute.

CURRENTRECORD

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through BLOCKSIZE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The CURRENTRECORD attribute returns the size, in FRAMESIZE units, of the last record read or written. For files other than port files, if BLOCKSTRUCTURE is FIXED, then the value of CURRENTRECORD always equals that of MAXRECSIZE. For port files, CURRENTRECORD is always the ACTUALMAXRECSIZE of the LASTSUBFILE.

If BLOCKSTRUCTURE is EXTERNAL and the previous operation was a READ, CURRENTRECORD corresponds to the size of the data not previously read from the current block. If the file is open but no I/O action has occurred, if the file is closed, or if the last operation caused an I/O error, then CURRENTRECORD returns 0. If the previous action was a SEEK, then the value of CURRENTRECORD is undefined.

CYCLE

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	when not in directory (disk) when closed (tape)
Type:	integer
Range:	0 through 9999
Default:	1
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The CYCLE attribute, in conjunction with the VERSION attribute, is used to denote the different generations of a permanent file. The initial, and default, CYCLE value is 1. The value of the CYCLE attribute may be modified only when the file is closed, except in the case of the creation of a disk file, where CYCLE can be changed any time before the file is entered into the directory. (Refer to the description of the PROTECTION attribute.)

Setting the CYCLE attribute to 0 has the special effect of resetting both CYCLE and VERSION to their default values of 1 and 0, respectively, and marks the logical file as not requiring the specific genealogy checking that it normally receives when it is first 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), then 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, then an "UNMATCHED GENEALOGY" notification is given to the operator. The operator can respond by making the file available or by using the FA (File Attribute) or DS (DiScontinue) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

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

CYLINDERMODE

Kinds:	pack only
Interrogate:	anytime
Modify:	when unassigned
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	pack only
Parameters:	none
BNA Logical I/O:	not supported

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

DATE

DATE is a nonpreferred synonym for CREATIONDATE. Refer to the description of the CREATIONDATE attribute.

DENSITY

Kinds:	tape
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	BPI200, BPI556, BPI800, BPI1600, BPI6250
Default:	(see below)
Stored Permanently:	tape
Parameters:	none
BNA Logical I/O:	supported

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

The mnemonic values associated with the DENSITY attribute are as follows:

Mnemonic Value	Integer Value
BPI200	2
BPI556	1
BPI800	0
BPI1600	3
BPI6250	4

BPI556 is not valid for a 9-track tape (TAPE9). BPI1600 is valid only for phase-encoded tapes (PETAPE). The mnemonic BPI6250 is valid only 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 multivolume file, the density setting remains constant from volume to volume 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 files. This value of DENSITY becomes the hardware-required value of the next volume.

The following equivalent mnemonics for the DENSITY attribute are supported as nonpreferred synonyms of the mnemonics in the preceding list:

Mnemonic	Synonym
BPI200	LOW
BPI556	MEDIUM
BPI800	HIGH
BPI1600	SUPER

DEPENDENTSPECS

Kinds:	all except port files
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

If the DEPENDENTSPECS attribute is TRUE when the file is opened, the logical file assumes the structure of the associated physical file. In other words, the BLOCKSIZE, BLOCKSTRUCTURE, FILETYPE, FRAMESIZE, MAXRECSIZE, MINRECSIZE, SIZEMODE, SIZEOFFSET, SIZE2, and UNITS attributes 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, then DEPENDENTSPECS is ignored.

If DEPENDENTSPECS has not been specified, then the structure of the logical file is determined by the default or user-specified values for BLOCKSIZE, BLOCKSTRUCTURE (or FILETYPE), FRAMESIZE, MAXRECSIZE, and MINRECSIZE.

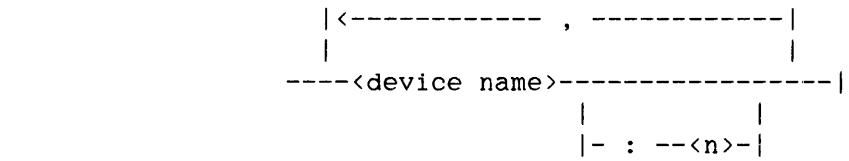
If DEPENDENTSPECS is specified and FILETYPE is 7 or 8, then FILETYPE is assigned a value of 0 when any attempt is made to open the file. An attribute error is reported if DEPENDENTSPECS is specified and an attempt is made to modify FILETYPE to 7 or 8.

If DEPENDENTSPECS is TRUE for the logical file and the FILETYPE of the associated physical file does not correspond to any BLOCKSTRUCTURE value (where FILETYPE is 2 or 4), then an attribute error is reported when the file is opened, FILETYPE for the logical file assumes the FILETYPE value for the physical file, and BLOCKSTRUCTURE reverts to a "not specified" condition.

DESTINATION

```
                    Kinds: printer/punch
              Interrogate: anytime
                   Modify: anytime
                     Type: pointer
                  Default: not applicable
      Stored Permanently: no
               Parameters: none
     BNA Logical I/O: not supported
```

The DESTINATION attribute specifies a list of destinations to which a printer/punch backup file should be routed for printing. When this attribute is set, the string value must conform to the following syntax:



For the definition of <device name>, see "Operator Commands That Control Output" in the Print System (PrintS/ReprintS) User's Guide.

For each destination, the number of copies to be printed can be indicated by ":<n>", where <n> is an integer. If the number of copies is not specified for a particular destination, it defaults to the value of the PRINTCOPIES attribute.

The number of copies specified for each device are printed on that device.

If a non-null value is specified for the DESTINATION file attribute at any point (such as file creation or PRINTDEFAULTS), it overrides the DESTNAME task attribute (see DESTNAME in the CANDE Reference Manual).

The default value of the DESTINATION attribute is a null string, which means print the number of copies specified by PRINTCOPIES at the destination indicated by the DESTNAME task attribute.

If PRINTDISPOSITION = DIRECT, the DESTINATION attribute does not apply and is thus ignored.

DIALOGCHECKINTERVAL

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 1440 minutes
Default:	0
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The DIALOGCHECKINTERVAL attribute is applicable only to BNA V2 port files.

The DIALOGCHECKINTERVAL attribute indicates the minimum amount of time, in minutes, that the system will wait on a silent dialog before automatic invocation of inactivity handshaking procedures. The inactivity handshake attempts to verify that the dialog is still intact. If it fails, the subfile will be deactivated. The default value is 0, which indicates that the inactivity procedures will never be invoked.

If MAXSUBFILES is greater than 1, the DIALOGCHECKINTERVAL attribute requires a subfile index as a parameter.

See also
FILESTATE 194
MAXSUBFILES 235

DIALOGPRIORITY

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 7
Default:	0
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The DIALOGPRIORITY attribute is applicable only to BNA V2 port files.

Currently, the maximum supported value of DIALOGPRIORITY is 2. Any attempt to set it to a larger value will result in a value of 2.

The DIALOGPRIORITY attribute indicates the priority of transmissions from this subfile, relative to other subfiles. Higher values indicate a higher message priority. Accessing DIALOGPRIORITY while the subfile is open returns the current DIALOGPRIORITY value. (This may not be the value requested by the user.) Accessing DIALOGPRIORITY when the subfile is closed returns the value of DIALOGPRIORITY that the user requested.

Note that the value of DIALOGPRIORITY should not be changed in rapid sequence. This introduces inefficiencies and cannot affect the message delivery order. DIALOGPRIORITY does not dictate priority on a message-by-message basis, but rather indicates the long-term priority of transmissions from this dialog.

The default value for this attribute can be changed via the BNA Operations Interface Command PORTDEFAULTDIALOGPRIORITY (PDDP).

If MAXSUBFILES is greater than 1, the DIALOGPRIORITY attribute requires a subfile index as a parameter.

See also
MAXSUBFILES 235

DIRECTION

```
                Kinds: tape/paper tape
          Interrogate: anytime
               Modify: anytime (Direct I/O)
                       when closed (non-Direct I/O)
                 Type: mnemonic
                Range: FORWARD, REVERSE
              Default: FORWARD
Stored Permanently: no
           Parameters: none
   BNA Logical I/O: supported
```

The DIRECTION attribute indicates the direction in which records are to be read from a file.

The mnemonic values associated with the DIRECTION attribute are as follows:

Mnemonic Value	Integer Value
FORWARD	0
REVERSE	1

The DIRECTION attribute is meaningful for paper tape reader files only when the NORVRSPAPERTAPE run-time system option is reset; otherwise, it is ignored. (For further information about the NORVRSPAPERTAPE run-time option, refer to the description of the OP (OPtions) ODT command in the Operator Display Terminal (ODT) Reference Manual.)

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 (IOCW attribute) is also changed. If a change in direction from forward to reverse is concurrent with a change from writing to reading, then system label records are written on the tape before the read is initiated.

For a non-Direct file, DIRECTION may be modified only when the file is closed.

A file cannot be read in a reverse direction unless BLOCKSTRUCTURE is FIXED or the file is unblocked.

DISPOSITION

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Range:	0 through 6
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

The DISPOSITION attribute returns the disposition of a station in the remote file. The DISPOSITION attribute requires one parameter, the Relative Station Number (RSN). The values returned and their meanings are as follows:

Value	Meaning
0	Unknown disposition
1	Assigned
2	Denied assignment
4	Assignment postponed
6	Denied assignment because an illegal use was attempted

Example

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

See also

Relative Station Number 91
Opening Remote Files. 92

DONOTSEARCHNETWORK

Kinds:	port-subfile
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	see below
BNA Logical I/O:	not supported

The DONOTSEARCHNETWORK attribute is applicable only to BNA V2 port files.

The value of DONOTSEARCHNETWORK is significant only if YOURHOST is a null string.

Setting DONOTSEARCHNETWORK to TRUE indicates to BNA that the network is not to be searched for a subport matching the subport that is being opened. If the subport cannot be opened without searching the network, then the OPEN will be suspended (if AVAILABLEONLY = FALSE) or it will be abandoned (if AVAILABLEONLY = TRUE). Note that, for two port-subfiles on different hosts to match, at least one of them must have DONOTSEARCHNETWORK = FALSE, or at least one of them must have a non-null YOURHOST value.

If MAXSUBFILES is greater than 1, the DONOTSEARCHNETWORK attribute requires a subfile index as a parameter.

See also
AVAILABLEONLY 126
MAXSUBFILES 235

DUPLICATED

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The DUPLICATED attribute indicates whether or not a disk file has or is to have area-by-area redundancy maintained by the I/O Subsystem.

If DUPLICATED is TRUE when creating a new file, then the amount of redundancy (that is, the number of copies of the file) is determined by the COPIES attribute. (Refer to the description of the COPIES attribute.)

When accessing an existing duplicated file, DUPLICATED must be TRUE for the logical file.

ENABLEINPUT

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The ENABLEINPUT attribute indicates whether or not a remote file is enabled for input. If no parameters are specified, ENABLEINPUT is TRUE if any station in the file is enabled for input. If a parameter is specified, ENABLEINPUT returns TRUE if the specified relative station is enabled for input.

EOF

Kinds:	tape
Interrogate:	when open
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The EOF attribute indicates whether or not an end-of-file condition has been reached on a tape file. When EOF is TRUE, end-of-file has been reached; when EOF is FALSE, end-of-file has not been reached.

Depending on the situation, it may be possible to reset EOF by accessing a valid record. This action may be performed by reading backward or by backspacing.

The EOF attribute is not valid for Direct I/O files. For the corresponding information concerning Direct I/O files, refer to the description of the IOEOF buffer attribute under "Direct I/O".

See also
IOEOF 354

ERRORTYPE

Kinds: disk
Interrogate: when open
Modify: never
Type: mnemonic
Range: 0 through 3
Default: not applicable
Stored Permanently: no
Parameters: 1 optional
BNA Logical I/O: not supported

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

The mnemonic values and meanings associated with the the ERRORTYPE attribute are as follows:

Mnemonic Value	Integer Value	Meaning
NOERROR	0	No error occurred
READCHECKFAILURE	3	Read-check failure
READPARITYERROR	2	Read parity error
SUNOTREADY	1	Storage unit not ready

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 on the buffer currently in use for data transfer to or from the program.

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

For related physical I/O error information, refer to the description of the IOINERROR attribute.

See also
Duplicated Files. 373

EXCLUSIVE

Kinds:	disk
Interrogate:	anytime
Modify:	when unassigned
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The EXCLUSIVE attribute 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 explicitly, or through the OPEN or PRESENT attribute, causes one of the following situations:

a. If the file does not exist and an explicit or implicit OPEN is attempted, the program waits on a "NO FILE" RSVP. If PRESENT is used, it returns FALSE.

b. If the file exists and is in use by another program, the program waits for the other program to close the file. A "WAITING ON" notification is displayed.

c. If the file exists and is not in use by any other program, it is opened and all future programs attempting to open the file will wait for this program to close the file.

Refer to the descriptions of the OPEN and PRESENT attributes.

The RESIDENT attribute 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 description of the AVAILABLE attribute.

EXTENT

Kinds:	diskette
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 15282
Default:	(see below)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The EXTENT attribute applies to diskette files only. EXTENT defines the maximum number of sectors a file can occupy and is analogous to the AREAS and AREALENGTH attributes for disk files. If EXTENT is not specified or 0, the remaining available space on the physical medium is assigned to this file. A value larger than the maximum space on a diskette (1898 sectors) produces a multivolume file. When a file is created, the specified extent is allocated on the volume. Operator intervention is required when the available space on the volume is smaller than the requested extent. Following an "output volume switch", the EXTENT value is reduced by the amount consumed on the volume that was closed.

When creating a multivolume diskette file, the MCP requests scratch (empty) volumes for each volume except the last. The last volume is allocated when the remaining extent has been reduced to 1898 sectors or less.

EXTMODE

```
                  Kinds: all devices
           Interrogate: anytime
                Modify: when closed
                  Type: mnemonic
                 Range: ASCII, BCL, BINARY, EBCDIC, HEX, SINGLE
               Default: INTMODE
Stored Permanently: disk/tape
           Parameters: 1 optional
   BNA Logical I/O: restricted values
```

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

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

ASCII, BINARY, EBCDIC, and SINGLE are supported by the BNA Logical I/O Host Service; BCL and HEX are not supported by the BNA Logical I/O Host Service.

BNA V1 port files support EBCDIC and SINGLE values of EXTMODE. BNA V2 port files support ASCII, EBCDIC, and SINGLE values of EXTMODE. For all port files, SINGLE is treated as EBCDIC.

For BNA V2 port files, the EXTMODE attribute has one optional parameter: a port-subfile index.

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 remote files or ODT files is unconditionally EBCDIC. Card files (READER/PUNCH) can have only BCL, EBCDIC, or BINARY for their EXTMODE values. The physical mode of all diskette files is EBCDIC.

An open error is given if an attempt is made to open a file with an INTMODE or EXTMODE of BCL on systems that do not support BCL. If a program opens a file with one or more attributes modified to BCL on a non-BCL system, then the program terminates with the message "BCL NOT ALLOWED ON THIS MACHINE".

Whenever the EXTMODE and INTMODE attributes differ, there is a possibility that the data is being translated. Refer to the description of hardware and software translation under the TRANSLATE attribute.

EXTMODE may not be changed for an existing, labeled permanent file. In this case, if a logical file is assigned to this permanent physical file, then the EXTMODE of the permanent file is used as the EXTMODE of the logical file while it is assigned to the permanent file.

See also
INTMODE 213

FAMILY

FAMILY is a nonpreferred synonym for STATIONLIST. Refer to the description of the STATIONLIST attribute.

FAMILYINDEX

Kinds:	disk
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 255
Default:	0
Stored Permanently:	disk
Parameters:	2 optional
BNA Logical I/O:	supported

A FAMILYINDEX is a family-relative number that designates a specific physical unit within a disk family.

When allocating disk space for a file, the value of the FAMILYINDEX attribute is used to identify the family member on which areas are to be allocated. If the value of FAMILYINDEX is 0, then the disk areas are allocated in the system's normal rotational order.

When space is unavailable on the original unit, the system message "REQUIRES FAMILYINDEX(N) SECTORS" is displayed. At this point, the OK (Reactivate) ODT command may be used to override the family index requirement, allowing the file to overflow to another unit in the family, or the DS (DiScontinue) ODT command may be used to terminate the program. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.) The OK command results in a system message indicating the FAMILYINDEX on which space was found:

```
SECTORS FOUND ON FAMILYINDEX (M)
```

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 specified for a specific area after the file is opened, but only before the disk space is allocated for that area.

When the logical file is assigned to a physical file, the FAMILYINDEX attribute requires the area number as a parameter. If the file is duplicated, then the copy number is also required.

The FAMILYINDEX of areas of a disk file that were assigned by the use of the FAMILYINDEX attribute is preserved by library maintenance, but it 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.

Examples (ALGOL)

```
DISKFILE.FAMILYINDEX := 1;

DISKFILE(ROWNUMBER).FAMILYINDEX := 3;

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

See also
Duplicated Files. 373

FAMILYNAME

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	"DISK"
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The FAMILYNAME attribute indicates the name or label of the disk family on which the physical file is located. FAMILYNAME must be a <simple identifier> of 17 or fewer characters.

"TAPE" cannot be specified as a value for the FAMILYNAME attribute.

If family substitution is in effect for the specified or default FAMILYNAME of the file, then the substitute or alternate FAMILYNAME is used when searching for the file. (For specific information regarding family substitution, refer to the <FAMILY specification> in the Work Flow Language Reference Manual.)

FAMILYNAME returns the name of the family on which the physical file resides if interrogated when the file is assigned.

FAMILYSIZE

FAMILYSIZE is a nonpreferred synonym for STATIONCOUNT. Refer to the description of the STATIONCOUNT attribute.

FILEKIND

Kinds: disk
Interrogate: anytime
Modify: anytime
Type: mnemonic
Range: (see below)
Default: DATA
Stored Permanently: disk/tape
Parameters: none
BNA Logical I/O: restricted usage/values

The FILEKIND attribute describes the internal structure and purpose of a record of a disk file. If FILEKIND is interrogated when the file is assigned, it returns the value for the physical file. FILEKIND can be modified before a disk file is opened. It is assigned to the physical file at the time the file is opened. If the physical FILEKIND is inconsistent with the privileges of the program, then a run-time attribute error is given. Compilers are privileged in that they may change a data file to a code file. Non-compilers are limited to changing FILEKIND to non-code values.

For disk files, certain restrictions on FILEKIND changes have been defined in terms of the following FILEKIND value ranges:

CLASS	CONDITION			
System files		FILEKIND <	COMPILERCODEFILE	
Compilers		FILEKIND =	COMPILERCODEFILE	
Code	COMPILERCODEFILE <	FILEKIND <=	CODEFILE	
Data	ALGOLSYMBOL <=	FILEKIND		

For example, CATALOG, DIRECTORY, and XDISKFILE are system FILEKINDs; COMPILERCODEFILE is a compiler FILEKIND; ALGOLCODE, JOBCODE, and COBOL74CODE are code FILEKINDs; DCALGOLSYMBOL, FORTRANSYMBOL, FIRMWARE, and TEXTDATA are data FILEKINDs.

A user program may change one data FILEKIND to another and may change anything but a system file to a data file. In addition, a compiler may change a data file to a code file through the use of file declarations or run-time attribute change statements, or both. However, a compiler may not create any code files across the BNA network. Only the operating system may change or assign a system FILEKIND or assign a compiler FILEKIND.

The integer and mnemonic values associated with FILEKIND for disk files are as follows:

0 NULLFILE	48 RPGCODE
1 DIRECTORY or VERSIONDIRECTORY	50 FORTRAN77CODE
2 SYSTEMDIRECTORY	53 SORTCODE
3 CATALOG	54 COBOL74CODE
4 BACKUPPRINTER or BACKUPDISK	62 BOUNDCODE
5 RECONSTRUCTIONFILE	63 CODEFILE
6 SYSTEMDIRFILE	64 ALGOLSYMBOL
7 JOBDESCFILE	65 COBOLSYMBOL
8 ARCHIVELOG	66 FORTRANSYMBOL
10 BACKUPVSID	67 XALGOLSYMBOL
15 XDISKFILE	68 PLISYMBOL
16 BACKUPPRINTER	69 JOVIALSYMBOL
17 BACKUPPUNCH	71 ESPOLSYMBOL
18 REMOTEBACKUP	72 DCALGOLSYMBOL
19 REMOTEAUDIT	73 BASICSYMBOL
20 COMPILERCODEFILE	74 XFORTRANSYMBOL
21 CHECKPOINTFILE	75 JOBSYMBOL
22 CPJOBFILE	77 VFORTRANSYMBOL
23 DCPCODE	78 SFORTRANSYMBOL
24 NDLCODE	79 NEWPSYMBOL
25 NDLIICODE	80 SANSSYMBOL
26 RECOVERYFILE	82 RPGSYMBOL
27 SCHEDULEFILE	83 NDLIISYMBOL
28 INFOFILE	84 FORTRAN77SYMBOL
29 LIBRARYCODE	85 SORTSYMBOL
30 INTRINSICFILE	86 COBOL74SYMBOL
31 MCPCODEFILE	94 BINDERSYMBOL
32 ALGOLCODE	95 DASDLSYMBOL
33 COBOLCODE	96 DMALGOLSYMBOL
34 FORTRANCODE	97 DCPSYMBOL
35 XALGOLCODE	98 NDLSYMBOL
36 PLICODE	104 MDLCODE
37 JOVIALCODE	105 MDLSYMBOL
39 ESPOLCODE	106 VFORTRANCODE
40 DCALGOLCODE	107 VLINKEDCODE
41 BASICCODE	108 VMCPCODE
42 XFORTRANCODE	109 LCOBOLSYMBOL
43 JOBCODE	110 LCOBOLSL3CODE
44 DMALGOLCODE	111 LCOBOLSL5CODE
45 NEWPCODE	112 SFORTRANCODE
46 SANSCODE	113 SLINKEDCODE

114 SMCPCODE	183 GRAPHICS
115 FIRMWARE	191 BDDATA
116 OHNESYMBOL	192 DATA
165 MDTTEST	193 SEQDATA
166 MDTDUMP	194 GUARDFILE
167 PROMBURNERDATA	195 APLDATA
168 CONFIGURATIONDATA	196 APLWORKSOURCE
169 CONFIDENCECODE	197 DDATA
176 MP1FIRMWARE	198 CSEQDATA
177 MP1ALGOLCODE	199 DBRESTARTSET
180 PAGEBACKUP	200 DBDATA
181 FONT	201 TEXTDATA
182 FORM	202 PRINTERCONTROLFILE

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.

Files with FILEKINDs FORM, GRAPHICS, and FONT are used by the IP subsystem to hold form, graphic, and font data for the image printer.

PRINTERCONTROLFILE should be the FILEKIND of the files referenced by the PRINTERCONTROL attribute.

For tape files, FILEKIND is a nonpreferred spelling of LABELKIND.

Tape FILEKINDs are not supported by the BNA Logical I/O Host Service; LABELKIND can be used instead. Refer to the description of the LABELKIND attribute.

FILENAME

Kinds:	all devices
Interrogate:	anytime
Modify:	anytime
Type:	pointer
Default:	INTNAME
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted values

The FILENAME attribute gives the external file name of a file. FILENAME is used primarily to identify the physical file. The default value for FILENAME is the value of the INTNAME attribute.

If the KIND attribute for the file is DISK or PACK, then 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 is not DISK or PACK, then FILENAME and TITLE return the same value. Refer to the descriptions of the TITLE and FAMILYNAME attributes.

If the file is a disk file, then FILENAME may be changed at any time. Changing the FILENAME of a disk file when the file is assigned also changes the file name 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 interrogated at any time. If interrogated when the file is assigned, then FILENAME returns the value for the physical file.

When a printer or punch backup file is open and assigned to a physical file, the FILENAME attribute returns the file name of the physical file. When the file is closed, the value assigned by the user program is returned.

Tapes have a special file-naming convention. Refer to "Magnetic Tape Files" in this manual.

The FILENAME attribute is accepted as a synonym for STATIONNAME for REMOTE files. Use of the STATIONNAME attribute is preferred to the use of the FILENAME attribute for REMOTE files.

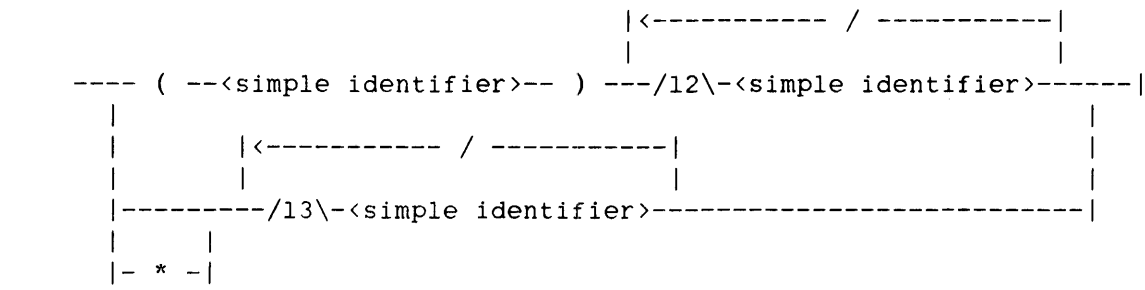
For port files, FILENAME must be of the form <simple identifier>. Attempting to open a port file with an invalid attribute value results in an open error.

A diskette physical file is denoted by a single-level, alphanumeric identifier of up to eight characters beginning with an alphabetic character. The FILENAME attribute is not truncated when searching for a permanent file. A fatal FILENAME error occurs if a task attempts to create a diskette file with a name that does not meet these requirements.

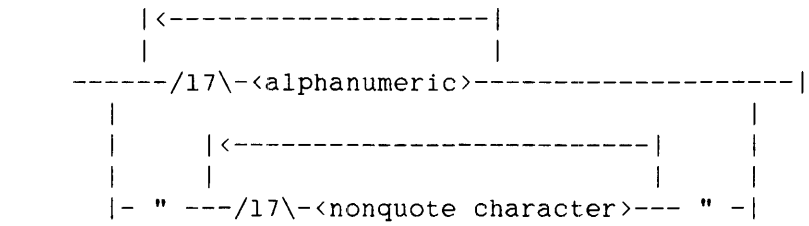
If FILENAME is set to the value TRACKZERO, then track zero of a diskette (the track containing the labels of the files on that diskette) may be read. Such a "file" is read-only.

A valid value for the FILENAME of a file of the remaining KINDs takes the following form:

<file name>

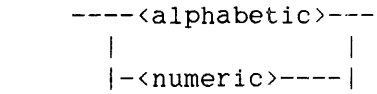


<simple identifier>



The BNA Logical I/O Host Service allows only alphanumeric characters. Hyphens (-) and underscores (_) are not permitted for port files.

<alphanumeric>



<alphabetic>
    Any EBCDIC upper-case letter (A through Z)

<numeric>
    Any EBCDIC digit (0,1,2,3,4,5,6,7,8,9)

<nonquote character>
    Any EBCDIC character except a quotation mark (")

See also
    Magnetic Tape Files 62
    USERBACKUPNAME. 332

FILEORGANIZATION

Kinds: disk
Interrogate: anytime
Modify: when closed
Type: mnemonic
Range: INDEXED, INDEXEDNOTRESTRICTED, NOTRESTRICTED, PLIISAM, RELATIVE
Default: NOTRESTRICTED
Stored Permanently: disk
Parameters: none
BNA Logical I/O: restricted values

The FILEORGANIZATION attribute is available so that the use of files may be restricted as to the organization under which they were created. The currently defined mnemonic values and meanings associated with FILEORGANIZATION are as follows:

Mnemonic Value	Integer Value	Meaning
INDEXED	2	The file was created under an indexed file organization and adheres to the restrictions of such. This value is implemented through the KEYEDIO library. For more information about KEYEDIO, refer to the System Software Support Reference Manual.
INDEXEDNOTRESTRICTED	3	The file was created with relative keys and may have indexed keys. It may be accessed when the logical file is declared as NOTRESTRICTED, INDEXED, or INDEXEDNOTRESTRICTED. This value is implemented through the KEYEDIO library. For more information about KEYEDIO, refer to the System Software Support Reference Manual.
NOTRESTRICTED	0	The file does not adhere to any FILEORGANIZATION restrictions.
PLIISAM	4	The file was created using the PLISUPPORT ISAM intrinsics.
RELATIVE	1	The file has a relative file organization with its restrictions. This organization is derived from COBOL74.

NOTRESTRICTED is supported by the BNA Logical I/O Host Service; INDEXED, INDEXEDNOTRESTRICTED, PLIISAM, and RELATIVE are not supported by the BNA Logical I/O Host Service.

If the disk file was originally created with a FILEORGANIZATION value other than the value in the logical file (this latter value can be the default or a user-specified value), the disk FILEORGANIZATION value is NOTRESTRICTED, RELATIVE, or INDEXED and the file is not a direct file, then an open error occurs when an open is attempted.

Specifying the DEPENDENTSPECS attribute as TRUE (or the FILETYPE attribute as 7 or 8) does not alter the value of FILEORGANIZATION when the file is opened.

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

FILESECTION

Kinds:	tape/diskette
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	1 through 9999 (tape) 1 through 99 (diskette)
Default:	1
Stored Permanently:	tape/diskette
Parameters:	none
BNA Logical I/O:	supported

The FILESECTION attribute is the ISO, BSI, and ANSI file section number of the first file header label record (HDR1). The FILESECTION number is file-relative; that is, the value of FILESECTION is incremented only when the file is involved in a volume switch (when the data of the file requires more than one physical volume of tape).

FILESECTION's value is modified to its default value of 1 whenever the file is closed. Also, when the operator enters an FR (Final Reel) ODT command in response to an RSVP issued to request the next volume, the value of FILESECTION is modified to 1. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of the FR (Final Reel) command.)

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

FILESTATE

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	mnemonic
Range:	(see below)
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	supported

The FILESTATE attribute returns a value that indicates the logical state of the file.

If the file is a port file and more than one subfile exists or the file is a remote file, then FILESTATE requires a subfile index as a parameter. When the FILESTATE attribute for a subfile is interrogated, the CHANGEEVENT for that subfile is reset (set to NOT HAPPENED). (For more information, refer to "Opening Port Files" and "Closing Port Files".)

The mnemonic values and meanings associated with FILESTATE are as follows. The values CLOSED and OPENED are meaningful for all files; the other values are meaningful only for remote and port files.

Mnemonic Value	Integer Value	Meaning
AWAITINGHOST	1	Indicates that the host specified by the YOURHOST subfile attribute cannot be reached or that the file is closed. The subfile toggles between this state and OFFERED until the host can be reached. The value of FILESTATE then changes to OPENED or CLOSED. All I/O operations to a subfile in this state return an end-of-file result. This value is only applicable to port files.
BLOCKED	5	Indicates that the remote host has become temporarily unreachable. The port remains open and all I/O operations are valid. This value is only applicable to port files.

General File Attributes

Mnemonic Value	Integer Value	Meaning
CLOSED	0	Indicates that the file or subfile is closed.
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. This value is only applicable to port files.
DEACTIVATED	8	Indicates that the complementary subfile is closed, that the system operator has terminated communication with the host involved in the subfile dialog by either initiating a fast shutdown of Network Services or clearing the host, or that BNA communication with the remote host has been terminated for some other reason. The local subfile has no queued input. This value is only applicable to port files. A close is the only valid operation for a subfile in this state; all read and write operations return an end-of-file result.
DEACTIVATIONPENDING	7	Indicates that the complementary subfile closed, and the local subfile has queued input or that BNA communication with the remote host has been terminated for some other reason. All write operations return end-of-file, but all read operations are valid. This value is only applicable to port files.
DENIED	9	Indicates that file assignment was denied. This value is only applicable to remote files.

Mnemonic Value	Integer Value	Meaning
DENIEDILLEGALUSE	11	Indicates that an illegal open was attempted. This value is only applicable to remote files.
OFFERED	2	Indicates that an open operation was attempted on the subfile, and the host specified by the YOURHOST attribute can be reached. However, no matching subfile was found. The subfile toggles between this state and AWAITINGHOST until the host can be reached. The value of FILESTATE then changes to OPENED or CLOSED. All I/O operations to a subfile in this state return an end-of-file result. This value is only applicable to port files.
OPENED	3	Indicates that the file or subfile is open and ready for input or output.
POSTPONED	10	Indicates that file assignment was postponed. This value is only applicable to remote files.
SHUTTINGDOWN	4	Indicates that the system operator wants to terminate communication with the host involved in the subfile dialog by either initiating a slow shutdown of Network Services or by requesting that the host be SAVEd. This value provides advance 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. This value is only applicable to port files.

See also

FILETYPE

Kinds:	all devices
Interrogate:	when unassigned (BNA Logical I/O) anytime (local host)
Modify:	when closed
Type:	integer
Range:	0 through 4; 6 through 8
Default:	0
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted values/usage

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

The preferred attributes BLOCKSTRUCTURE and DEPENDENTSPECS can override and change the value of FILETYPE. If neither BLOCKSTRUCTURE nor DEPENDENTSPECS has been specified, then FILETYPE determines the structure of the logical file.

The values and meanings associated with the FILETYPE attribute are as follows:

Value	Meaning
0	Blocked or unblocked, fixed-length records. Using the FIXED value of BLOCKSTRUCTURE is preferable to using this value of FILETYPE.
1	Variable-length records. The record size is contained (in decimal form) in the first four INTMODE characters of the record. If INTMODE is SINGLE, then the record size is stored in the first word of the record in binary. This size includes the length of the size field (for example, for character files, the size is four more than the length of the actual data). The size is expressed in UNITS. Depending on the value of the SIZEVISIBLE attribute, the record length field may or may not be returned as part of the record. Using the VARIABLE value of BLOCKSTRUCTURE is preferable to using this value of FILETYPE.

Value	Meaning
2	Variable-length records. The record size is contained in the first two INTMODE characters of the record and is expressed in binary. If INTMODE is SINGLE, then the record size is stored in the first word of the record. This size includes the length of the size field; for example, for character files, the size is two more than the length of the actual data.
3	Variable-length records. Neither the record itself nor the structure of the file contains information about the length of the record; therefore, this information must be specified by the I/O statement. Unblocked tape, ODT, and remote files are able to discover and return the true length of the record on input using this FILETYPE. For further information, refer to the descriptions of the CURRENTRECORD and STATE attributes. Using the EXTERNAL value of BLOCKSTRUCTURE is preferable to using this value of FILETYPE.
4	Variable-length records. The record size is contained in a fixed location in the record. The attributes SIZEMODE, SIZEOFFSET, and SIZE2 determine the mode of the information, the position of the record size information, and the length of the field containing the information, respectively.
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 SINGLE, and software translation is never attempted. Using the LINKED value of BLOCKSTRUCTURE is preferable to using this value of FILETYPE.

Value	Meaning
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, the attributes BLOCKSIZE, BLOCKSTRUCTURE, FILETYPE, INTMODE, MAXRECSIZE, MINRECSIZE, SIZEMODE, SIZEOFFSET, SIZE2, and UNITS are changed to the values used in the permanent file. FRAMESIZE is not changed, but it can be changed using DEPENDENTSPECS. 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 (that is, a new file is being created), then FILETYPE is set to 0 (or 3, depending upon the initial value of MINRECSIZE), and all the attributes mentioned previously are set to their default values if they were not explicitly set before the file was opened. Setting the DEPENDENTSPECS attribute to TRUE is preferable to using the value 7 for FILETYPE.
8	The action is the same as for FILETYPE 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, the attributes BLOCKSIZE, BLOCKSTRUCTURE, FILETYPE, MAXRECSIZE, MINRECSIZE, SIZEMODE, SIZEOFFSET, SIZE2, and UNITS are reset to their default values. Setting the DEPENDENTSPECS attribute to TRUE is preferable to using the value 8 for FILETYPE.

FILETYPE 0 is supported by the BNA Logical I/O Host Service; FILETYPE 3 is supported by the BNA Logical I/O Host Service only when the file is not blocked. FILETYPEs 1, 2, 4, 6, 7, and 8 are not supported by the BNA Logical I/O Host Service.

If FILETYPE is 0 and MINRECSIZE is greater than 0 but less than MAXRECSIZE, then FILETYPE is changed to 3 when the file is opened.

Files with FILETYPE 6 are always blocked. Files with all other values of FILETYPE are blocked only when BLOCKSIZE is greater than MAXRECSIZE.

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

The value of FILETYPE affects the default value of MINRECSIZE. Variable-length files that 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 0 use a value for MINRECSIZE that is equal to MAXRECSIZE.

The blocking technique used with variable-length, blocked files is as follows: if the record to be written plus the offset into the block is greater than BLOCKSIZE, then a physical write is initiated for the block. 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 (for example, tape files), then only the part of the block that is valid is written. For peripheral units that require fixed-length physical blocks (for example, disk files), a MINRECSIZE field of nulls (4"00") is added to the block as an end-of-block marker and the whole block is written.

Attempting to specify a SPACE to an externally specified variable-length record (where FILETYPE is 3) causes an "ILLEGAL SEEK" error.

The BNA Logical I/O Host Service does not allow the FILETYPE attribute to be modified or interrogated when the file is assigned. BLOCKSTRUCTURE and DEPENDENTSPECS can be used instead.

Attempting to open a port file with an unsupported FILETYPE or an unsupported BLOCKSTRUCTURE results in an open error (an unknown error occurred during an attempt to open the file). Note also that for port files, if SIZEVISIBLE = FALSE, FILETYPE can have only the values 0, 1, or 3. (However, if FILETYPE is either 0 or 3, SIZEVISIBLE has no effect.)

FILEUSE

Kinds:	all except port files
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	IN, IO, OUT
Default:	IO
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The FILEUSE attribute specifies how the file is to be used. The mnemonic values of the FILEUSE attribute and their corresponding meanings are as follows:

Mnemonic Value	Meaning
IN	Input only
IO	Both input and output
OUT	Output only

If the value of FILEUSE is IN and a write is performed, a fatal run-time error occurs and the message "WRITE ON INPUTFILE" is displayed. Similarly, if the value of FILEUSE is OUT and a read is performed, a fatal run-time error occurs and the message "READ ON OUTPUTFILE" is displayed.

Unlike the MYUSE attribute, the values of the FILEUSE attribute have no effect on the file search algorithms (that is, whether to search for an existing file or create a new one). Refer to the description of the MYUSE attribute.

FILEUSE is used for REMOTE files and for disk file security checking.

See also
    Interactions of the FILEUSE and Security Attributes 33

FLEXIBLE

Kinds:	disk
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	TRUE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The FLEXIBLE attribute indicates whether or not a disk file can be allocated more areas, if needed, than the number originally specified by the AREAS attribute. (Refer to the description of the AREAS attribute.) If FLEXIBLE is TRUE and more areas are needed, then 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 I/O 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 a duplicated file, or is an IAD file.

FORMID

Kinds:	printer/punch
Interrogate:	anytime
Modify:	when closed
Type:	string
Default:	(null string)
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The FORMID attribute is a string of up to 100 characters that identifies the kind of paper (or cards) that must be used to print a printer/punch backup file. If a matching string is assigned to a suitable printer, the output file is printed without operator intervention.

If the FORMID attribute is specified for a direct printer file and a matching FORMed printer is not available, then the string assigned to the attribute is displayed as part of the RSVP message requesting operator action. The program is suspended until a printer that has been FORMed with a matching string becomes available or the operator responds with an FM (Form Message), FORM (Assign FORM ID), or OU (Output Unit) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

If the file goes to backup and a matching formed printer is not then the print request for that backup file is marked as "WAITING" in the Print Request List. Refer to the PS SHOWREQUESTS command in the Print System (PrintS/ReprintS) User's Guide.

The default value is a null string, implying DONTCARE.

FORMMESSAGE

FORMMESSAGE is a nonpreferred synonym for FORMID. Refer to the description of the FORMID attribute.

FRAMESIZE

Kinds: all devices
Interrogate: anytime
Modify: when closed
Type: integer
Range: 4, 8, 48
Default: value corresponding to UNITS
Stored Permanently: disk/tape
Parameters: none
BNA Logical I/O: restricted values

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

The possible values and meanings for FRAMESIZE are as follows:

Value	Meaning
4	Data is transferred in units of 4 bits or as hexadecimal characters. INTMODE must be HEX. This value is not supported by the BNA Logical I/O Host Service is not allowed with port files.
8	Data is transferred in units of 8 bits or as EBCDIC or ASCII characters. INTMODE must be 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, then an open error is reported.

If FRAMESIZE is unspecified, then the value of UNITS is used. If FRAMESIZE is specified, then 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 interrogated when the file is open, then the value is computed from the physical file and returned even if FRAMESIZE was not specified for the logical file.

GENERATION

Kinds:	disk/tape
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through catalog level
Default:	0
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

Files with a common FILENAME, CYCLE, VERSION, and FAMILYNAME, but with different times of update, 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 latest time of update. The larger the value of GENERATION, up to the limit of the references in the System Catalog, the less recent the update (or creation).

The GENERATION attribute is ignored when USECATALOG = FALSE.

HOSTNAME

Kinds:	all devices
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	name of local host
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	supported

The HOSTNAME attribute specifies a <simple identifier> of 1 to 17 characters that is used during file assignment to specify the host on which the physical file or complementary port file exists or is to be created.

If the file is a port file and more than one subfile exists, HOSTNAME requires a subport index as a parameter.

In the case of a port-subfile, HOSTNAME is a nonpreferred synonym for YOURHOST.

If the file is closed and unassigned and HOSTNAME has never been set, then HOSTNAME returns the name of the local host. If the file is assigned, then HOSTNAME returns the name of the host on which the file resides.

The BNA Logical I/O Host Service does not allow HOSTNAME to be altered using the FA (File Attribute) ODT command once the file is open. For a description of the FA command, refer to the Operator Display Terminal (ODT) Reference Manual.

See also
MYHOSTNAME. 239
YOURHOST. 338

IAD

Kinds:	disk
Interrogate:	anytime
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

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

Installation-Allocated Disk is supported only on B 6800, B 7700, B 7800, and B 7900 Series systems.

INPUTEVENT

Kinds:	port/port-subfile/remote
Interrogate:	anytime (port/port-subfile)
	when open (remote)
Modify:	never
Type:	event
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional (port/port-subfile)
BNA Logical I/O:	not supported

When a subfile index is given, the INPUTEVENT attribute returns HAPPENED if the CENSUS attribute for the specified subfile is greater than 0. When no subfile index is given, the INPUTEVENT attribute returns HAPPENED if the CENSUS attribute for the file is greater than 0. (Refer to the description of the CENSUS attribute.) Because INPUTEVENT is a read-only attribute, it should not be passed as a parameter to the following statements: ATTACH, CAUSEANDRESET, DETACH, FIX, FREE, LIBERATE, PROCURE, RESET, SET, and WAITANDRESET.

For REMOTE files, the INPUTEVENT attribute returns HAPPENED if the CENSUS attribute is greater than zero. This attribute allows a simpler and more efficient interrogation of whether there exists any queued input to a remote file.

Interrogating INPUTEVENT while the remote file is closed results in a fatal attribute error.

Interrogating INPUTEVENT in conjunction with parameters when KIND is REMOTE also results in a fatal attribute error.

Attribute errors on event-valued attributes result in the termination of the program.

Example (ALGOL)

```
I := WAIT((5),SF(2).INPUTEVENT);

WAIT(F.INPUTEVENT);
```

INPUTTABLE

Kinds:	all except port files
Interrogate:	never
Modify:	anytime
Type:	translatetable
Default:	(no default)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

Setting the INPUTTABLE attribute allows the program to modify the I/O Subsystem's input software translation. INPUTTABLE is used only when the TRANSLATING attribute is TRUE; otherwise, the translatetable (if one was assigned) is ignored.

A translatetable assigned to INPUTTABLE that is declared in a program must be either the first table in a list of translatetables or the only table in that declaration. Each time the logical file is closed, the value of the INPUTTABLE attribute is discarded.

An example of the ALGOL syntax for assigning a translatetable to INPUTTABLE is as follows:

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

For further information regarding translation, refer to the description of the TRANSLATE attribute.

INTERCHANGE

Kinds:	pack only
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	pack only
Parameters:	none
BNA Logical I/O:	not supported

When TRUE, the INTERCHANGE attribute indicates that the file is stored or is 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.

For the special characteristics of interchange packs, refer to "Disk Files".

Interchange packs are supported only on B 6800, B 7700, and B 7800 Series systems.

See also

Disk Files. 49

INTMODE

```
                Kinds: all devices
          Interrogate: anytime
               Modify: when closed
                 Type: mnemonic
                Range: ASCII, BCL, EBCDIC, HEX, SINGLE
              Default: EBCDIC
Stored Permanently: no
           Parameters: none
   BNA Logical I/O: restricted values
```

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

Mnemonic Value	Integer Value	Meaning
ASCII	5	8-bit
BCL	3	6-bit
EBCDIC	4	8-bit
HEX	2	4-bit, packed decimal
SINGLE	0	Word mode, binary, or 48-bit

SINGLE, EBCDIC, and ASCII are supported by the BNA Logical I/O Host Service; HEX and BCL are not supported by the BNA Logical I/O Host Service.

BNA V1 port files support EBCDIC and SINGLE values of INTMODE BNA V2 port files support ASCII, EBCDIC, and SINGLE values of INTMODE. ASCII is only valid for BNA V2 port files. For port files, SINGLE is treated as EBCDIC.

The compiler assigns a default value for each file when creating the code file. The normal default of EBCDIC can be overridden by using the ASCII compiler control option in FORTRAN and ALGOL. COBOL and COBOL74 set INTMODE to the type of the first 01-level entry under the File Description (FD). (For example, if the first 01-level entry is USAGE DISPLAY, then INTMODE is EBCDIC; if the first 01-level entry is USAGE COMP-2, then INTMODE is HEX.) If the first 01-level entry is USAGE COMP and the B 2500 compilation option ($B2500) is set, then the COBOL compiler modifies INTMODE to HEX. Thus, the order of declaration of 01-level items under an FD can make a difference if the level items are

of different USAGEs.

An open error is given for trying to open a file with an INTMODE or EXTMODE of BCL on systems that do not support BCL. If a program opens a file with one or more attributes specified as BCL on a non-BCL system, then the program terminates with the message "BCL NOT ALLOWED ON THIS MACHINE".

Whenever the INTMODE and EXTMODE attributes differ, there is a possibility that the data is being translated. For further information regarding translation, refer to the description of the TRANSLATE attribute. INTMODE, in combination with the FRAMESIZE attribute (or the UNITS attribute), determines whether the data transfer is word- or character-oriented. When INTMODE and FRAMESIZE are not compatible, an error is reported when the file is opened.

If an I/O operation is attempted in which the character size of the array, pointer, or string specified in the I/O statement is different from the INTMODE of the file, the data transfer behaves in the same manner as the ALGOL "REPLACE" statement. For port files, an attempt to do an I/O operation with an array, pointer, or string whose character size is BCL or HEX causes a program fault.

If the type of the array passed to a READ or WRITE statement is not compatible with the INTMODE of the file, then an INVALID OP or INVALID STACK ARGUMENT fault will occur. If the array passed to a READ or WRITE statement is shorter than the amount of data requested, a PAGED ARRAY ERROR or SEG ARRAY ERROR fault will occur. These faults are treated in the same manner as faults in the user's program. As such, they can be handled by code in a fault block in the user's program. If there is no such block, the program is F-DSed.

INTNAME

Kinds:	all devices
Interrogate:	anytime
Modify:	when unassigned
Type:	pointer
Default:	as declared in program
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The INTNAME attribute is the internal file name. File equation is accomplished by matching the internal file name to the file specified in the task initiation statement. 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>. INTNAME may be modified only while the logical file is closed and not assigned to a physical file. INTNAME cannot be modified through the Work Flow Language (WFL). When the internal name of a file is changed, file equation action is initiated using the new internal name. Because only the first 17 characters of the identifier declared in a program are used in the file declaration, file identifiers should be unique in the first 17 characters.

Setting the INTNAME file attribute through either multiple or single file attribute assignment causes file equation statements for the new INTNAME to be processed. The example below illustrates an attempt to open a disk file.

Example (WFL)

```
?BEGIN JOB TEST/INTNAME;
COMPILE X ALGOL GO;
FILE NEWINT (KIND=DISK);
ALGOL DATA
BEGIN
FILE F (KIND=TAPE);
REPLACE F.INTNAME BY "NEWINT.";
OPEN(F);
END.
?END JOB
```

IOCLOCKS

Kinds:	all devices
Interrogate:	anytime
Modify:	never
Type:	integer
Range:	0 through 549755813887
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

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

The value of the IOCLOCKS attribute is always 0 for remote and port files.

IOINERROR

Kinds:	disk/tape/diskette
Interrogate:	when open
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

If the IOINERROR attribute returns TRUE, then a physical I/O error has occurred 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, then the physical I/O is completed asynchronously with the program. When a physical I/O is initiated, the buffers are rotated so that the 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 all the information to be gained from IOINERROR and is probably more useful. Error analysis for physical writes is always the value of BUFFERS behind the position of the logical file.

If IOINERROR returns TRUE, then the attributes ERRORTYPE, RECORDINERROR, and STATE 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

Kinds: all devices
Interrogate: anytime
Modify: when closed
Type: mnemonic
Range: (see below)
Default: DONTCARE
Stored Permanently: no
Parameters: none
BNA Logical I/O: restricted values

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

The mnemonic values associated with the KIND attribute are as follows. The first mnemonic for each value is preferred.

Mnemonic Value	Integer Value	Supported by BNA Logical I/O
DISK or SERIAL	1	yes
DISKETTE	25	no
DONTCARE	0	no
HC	20	no
PACK or DISKPACK	17	yes
PAPERPUNCH or PTP	5	no
PAPERREADER or PAPER or PTR	4	no
PORT	19	no
PRINTER	7	yes
PUNCH or CP	11	yes
READER	9	yes

General File Attributes

Mnemonic Value	Integer Value	Supported by BNA Logical I/O
REMOTE or DC	3	yes
ODT or SPO	2	no
TAPE	45	yes
TAPE7	13	yes
TAPE9	14	yes
TAPEPE or PETAPE (for PE or GCR tapes)	15	yes
VSID	35	no

When the file is opened in such a way that a new file is created, the value DONTCARE is converted to TAPE7.

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

The KIND value PACK is functionally synonymous with DISK, although DISK is the preferred value.

The BNA Logical I/O Host Service requires that KIND be specified.

LABEL

```
                Kinds: all except port files
          Interrogate: anytime
               Modify: never (remote)
                       when unassigned (others)
                 Type: mnemonic
                Range: OMITTED, OMITTEDEOF, STANDARD
              Default: OMITTED (remote)
                       STANDARD (others)
Stored Permanently: no
           Parameters: none
   BNA Logical I/O: supported
```

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

The mnemonic values associated with LABEL are as follows:

Mnemonic Value	Integer Value
OMITTED	1
OMITTEDEOF	3
STANDARD	0

If LABEL is STANDARD when creating a tape file, then the 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 OMITTED or OMITTEDEOF, then the label records are not included. When reading a labeled tape with LABEL equal to either OMITTED or OMITTEDEOF, the DENSITY and the PARITY are taken from the logical file description.

For printer files, if LABEL is STANDARD, standard banner pages are printed at the beginning and end of the file. If LABEL is OMITTED or OMITTEDEOF, only a skip-to-channel-1 is done at the beginning of the file. No skips or banners are added at the end of the file.

The LABEL attribute has no effect on printer backup files.

Labeled punch files use the format discussed under "Card Files". Note, however, that only Direct I/O files can create an unlabeled punch file.

LABEL may not be specified for remote files, and the value of OMITTED is used.

When reading a tape or paper tape file, the differences between OMITTED and OMITTEDEOF become important. When reading an unlabeled tape and a tape mark is encountered, if LABEL is OMITTED, a volume switch is attempted; if LABEL is OMITTEDEOF, then an end-of-file (EOF) action occurs. When LABEL is OMITTED, the operator can use the FR (Final Reel) ODT command to indicate that EOF has been reached. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.)

See also
Card Files. 47

LABELKIND

Kinds:	tape
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 13
Default:	not applicable
Stored Permanently:	tape
Parameters:	none
BNA Logical I/O:	supported

The LABELKIND attribute describes, in more detail than the LABEL attribute, the kind of label on the tape. The values and meanings associated with LABELKIND are as follows:

Value	Meaning	Value	Meaning
0	B 6500 USASI	7	B 6500 LIBRARY TAPE
1	UNLABELED	8	B 6500 LOAD CONTROL TAPE
2	B 5500 STANDARD	10	B 5500 BACKUP TAPE
3	SYSTEM STANDARD	11	B 5500 LIBRARY TAPE
4	B 3500 USASI	12	NON-BURROUGHS USASI
5	B 3500 STANDARD	13	USER-DEFINED
6	B 6500 BACKUP TAPE		

When a user program performs operations to get past the last file on a labeled tape (for example, performing "CLOSE(F,*)" a sufficient number of times in ALGOL), and an attribute that returns actual values from the current physical file (for example, CREATIONDATE or LABELKIND) is interrogated, the attribute acts as if the tape were unlabeled. Normally, the attribute either returns the value declared by the user or gives an attribute error; for LABELKIND, the value 1 (unlabeled) is returned.

LABELTYPE

LABELTYPE is a nonpreferred synonym for LABEL. Refer to the description of the LABEL attribute.

LASTRECORD

Kinds:	disk
Interrogate:	when assigned
Modify:	when assigned
Type:	integer
Range:	-1 through 549755813887
Default:	-1
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The LASTRECORD attribute 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, such as during the transition from writing to reading. The counting of records is zero-relative; that is, if there is only one record in the file, then LASTRECORD returns 0. LASTRECORD returns -1 for an empty disk file.

The LASTRECORD attribute of a permanent file is stored only when the logical file is closed or when a new space is allocated. Therefore, if a system failure occurs while the file is open and records have been added to the file, the new records may be lost because LASTRECORD may still reflect a previous value after the Halt/Load. Specifying the PROTECTION attribute as PROTECTED may be used to prevent this loss of records.

If the file does not have fixed blocking, then LASTRECORD is returned in blocks rather than records. (Blocking is fixed when BLOCKSTRUCTURE is FIXED or FILETYPE is 0).

The LASTRECORD attribute may be modified by a program if and only if PROTECTION is not 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.

The value of LASTRECORD may not be altered by file equation.

LASTSTATION

LASTSTATION is a nonpreferred synonym for LASTSUBFILE. Refer to the description of the LASTSUBFILE attribute.

LASTSUBFILE

Kinds:	port/remote
Interrogate:	when open
Modify:	never (port)
	when open (remote)
Type:	integer
Range:	0 through MAXSUBFILES (port)
	valid RSNs (remote)
Default:	0
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

If the file is a port file, then 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. After a broadcast WRITE operation, LASTSUBFILE returns the subfile index of the highest numbered open subfile.

If the file is a remote file used for non-Direct I/O and the file is open, then LASTSUBFILE returns the Relative Station Number (RSN) of the station from which the last message was received. If the program has modified the value of LASTSUBFILE, LASTSUBFILE returns the value to which it was modified.

When a station is added to the station list of a remote file, the LASTSUBFILE attribute is updated to be the RSN of the newly added station (if no attribute error occurred during the addition). In this way, a user program can determine the RSN of a newly added station for future use.

For remote files, output operations are always directed to the station specified by the LASTSUBFILE attribute; input operations are always nonselective. If a program sets LASTSUBFILE to an invalid RSN or the station is denied, then a write to the file returns end-of-file. Valid RSNs are greater than 0 and correspond to a valid station in the STATIONLIST. When LASTSUBFILE is 0 for normal (non-Direct) files, a write statement is broadcast to every station assigned to the file. If only one station is assigned to the file (where POPULATION equals 1), then a broadcast write is the same as a write directed to a single station. The attribute LASTSUBFILE can also be set by using the "[STATION <arithmetic expression>]" form of the "[<record number or carriage control>]" part of the ALGOL WRITE statement.

For more information regarding RSNs, refer to "Relative Station Number" under "Remote Files". The description of the STATIONLIST attribute also provides helpful information.

Direct I/O remote files do not use LASTSUBFILE to determine the originating RSN. The IORECORDNUM buffer attribute is used because it is associated with the particular buffer rather than the file. The output RSN is specified by IORECORDNUM when LASTSUBFILE is 0 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 particular RSN.

Example (ALGOL)

```
WRITE(<file id>[STATION 0],80,A);    %To broadcast

WRITE (<file id>[STATION 3],...);    %To station 3 only
```

See also

Remote Files. 90
Relative Station Number 91
IORECORDNUM 360

LINENUM

Kinds:	printer
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 255
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The LINENUM attribute indicates the current line number on the logical page as defined by the PAGESIZE attribute. The attribute is meaningful only for printer files where the PAGESIZE attribute is greater than 0. If the PAGESIZE attribute is nonzero when the printer file is opened, then LINENUM is initialized to 1. Refer to the descriptions of the PAGE and PAGESIZE attributes.

Every serial write statement increments LINENUM (while PAGESIZE is greater than 0). The data associated with a serial write resulting in an end-of-page condition is printed. (Refer to the STATE attribute for a description of the end-of-page condition.) An end-of-page condition does not automatically 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 program. After an end-of-page result, LINENUM is set to 1 and PAGE is incremented.

Specifying 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 causes the next write to return an end-of-page result, 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. (Refer to the description of the STATE attribute.)

The value of LINENUM may not be altered by file equation.

LINE Carriage Control

The "[LINE<arithmetic expression>]" form of the write statement provides for positioning of the output page. The action taken depends on the values of <arithmetic expression>, PAGESIZE, and LINENUM, and on the language being used. When the language used is ALGOL and the WRITEAFTER compiler control option is FALSE, the "[LINE<arithmetic expression>]"

form of the WRITE statement temporarily suspends the usual action of spacing after printing and instead spaces forward to the logical line specified by <arithmetic expression> before printing.

An end-of-page exception is returned if this WRITE statement is used when the PAGESIZE attribute has a value of 0 (in other words, when logical page accounting is not being done). Otherwise, when the value of <arithmetic expression> is less than or equal to PAGESIZE and greater than or equal to LINENUM, the page is spaced forward to the logical line number <arithmetic expression> and LINENUM is set to <arithmetic expression>. If the value of <arithmetic expression> is less than LINENUM, then spacing to the next logical page is performed (incrementing the PAGE attribute but not skipping to channel one), followed by spacing forward on the page to the logical line as described previously. Printing is done either before or after spacing, depending on the value(s) of the appropriate parameter(s) in the language.

Under the error condition when <arithmetic expression> is less than 1, LINENUM is set to 1, PAGE is incremented, the line is printed without spacing, and an end-of-page exception is returned to the program. A similar action occurs if <arithmetic expression> is greater than PAGESIZE.

SKIP Carriage Control

The "[SKIP <arithmetic expression>]" 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. PAGE is not incremented by skip-to-channel actions.

SPACE Carriage Control

When the carriage control is SPACE and the number of lines specified is greater than or equal to 0, the page is spaced forward the number of lines specified. If the number of lines specified is less than 0, then the page is spaced forward one line and an end-of-page message is returned to the program. LINENUM is incremented by the number of lines spaced, and if the sum at least equals PAGESIZE, then LINENUM is set to 1, PAGE is incremented, and an end-of-page result is returned to the program.

Refer to the descriptions of the PAGE, PAGESIZE, and STATE attributes.

MAXCENSUS

Kinds:	port-subfile
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 63
Default:	63
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	not supported

The value of the MAXCENSUS attribute specifies the number of input messages that can be queued for a subfile (CENSUS) before the MCP requests that the complementary subfile stop sending messages. Exceeding this limit will cause the complementary subfile to be suspended when it next attempts to perform a WRITE operation.

If the DONTWAIT option is specified in the WRITE statement, the WRITE operation returns a result with bit 8 equal to 1, indicating that no buffer is available. (For more information, see the STATE attribute.) Otherwise, the sending program is suspended. The program will be resumed when enough messages have been read from the receiving subfile to bring it several messages below the MAXCENSUS value.

Because of the time lag in communication between the two subfiles, some messages may be sent by the complementary subfile after the stop request is sent but before it is received by the complementary subfile. This may result in the CENSUS value exceeding the MAXCENSUS value before the complementary subfile is actually suspended.

MAXCENSUS requires one parameter, the subfile index.

See also
Port File I/O Operations. 84

MAXRECSIZE

Kinds:	all devices
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	see below
Default:	(see below)
Stored Permanently:	disk/tape
Parameters:	1 optional
BNA Logical I/O:	restricted values

The MAXRECSIZE attribute specifies the maximum size of records in the logical file. If FRAMESIZE is not 48 (or UNITS is CHARACTERS), then MAXRECSIZE is expressed in INTMODE units; otherwise, it is expressed in words. If MAXRECSIZE is unspecified or 0, then 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 BLOCKSIZE attribute.

The following list shows legal MAXRECSIZE values for various KINDs of files:

Kind	Range
Local diskette files	2 through 128 characters
Other local files	1 through 65535 characters
BNA Logical I/O files	1 through 65487 character
Port files (BNA V1)	1 through 65535 characters
Port files (BNA V2) local dialog	1 through 65513 characters
Port files (BNA V2) remote dialog	1 through 20000 characters

For a diskette file, the MAXRECSIZE value may range from 2 to 128 characters (or 1 to 21 words). Diskette files always contain one logical record per sector regardless of the value of MAXRECSIZE.

For a port file, MAXRECSIZE is a nonpreferred synonym for ACTUALMAXRECSIZE (when used as a subfile attribute) or REQUESTEDMAXRECSIZE (when used as a file attribute). See the descriptions of the ACTUALMAXRECSIZE and REQUESTEDMAXRECSIZE attributes for more details. The MAXRECSIZE value for a port-subfile can never be changed. Non-A Series systems may have a different default and maximum value. The maximum will always be at least 2000 characters.

The BLOCKSTRUCTURE, KIND, MAXRECSIZE, and MINRECSIZE attributes are closely related. MAXRECSIZE must be less than or equal to BLOCKSIZE. If BLOCKSTRUCTURE is FIXED, then BLOCKSIZE must be a multiple (possibly

0 or 1) of MAXRECSIZE. If MINRECSIZE is greater than MAXRECSIZE, then it is set to MAXRECSIZE.

If MINRECSIZE is greater than 0 and less than MAXRECSIZE and BLOCKSTRUCTURE is unspecified, then the value of BLOCKSTRUCTURE is set to EXTERNAL. If MAXRECSIZE is 0 and BLOCKSIZE is greater than 0, then the value of MAXRECSIZE is set to the value of BLOCKSIZE.

If both MAXRECSIZE and BLOCKSIZE are 0 when the file is opened, then the default values assigned to both attributes depend upon the KIND of peripheral unit assigned to the file. The default values are as follows:

Kind	Default MAXRECSIZE and 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 words else 22 words
CARDREADER or CARDPUNCH	If EXTMODE is BINARY, then 20 words else if EXTMODE is EBCDIC, then 14 words else 10 words
TAPE	10 words
PORT	320 words

If the FRAMESIZE attribute is not 48 (or UNITS is CHARACTERS), then 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, then the maximum value is 80 characters.

The value of the SIZEVISIBLE attribute affects the meanings of the MINRECSIZE, MAXRECSIZE, and BLOCKSIZE attributes. When SIZEVISIBLE is FALSE at file creation time, the values of MINRECSIZE and MAXRECSIZE are considered not to include the length of the system-maintained record size field, and are adjusted upward accordingly before being stored in the physical file header. If this adjustment results in MAXRECSIZE being greater than BLOCKSIZE, BLOCKSIZE is also adjusted upward. The values returned to the user when the attributes MINRECSIZE and MAXRECSIZE are interrogated reflect the logical values originally specified before they were adjusted to include the system overhead fields. BLOCKSIZE always returns the actual size of the block being used.

See also

ACTUALMAXRECSIZE. 108
Port File I/O Operations. 84
REQUESTEDMAXRECSIZE 274
SIZEVISIBLE 296

MAXSUBFILES

Kinds:	port
Interrogate:	anytime
Modify:	increase only (see below)
Type:	integer
Range:	1 through 1023 (BNA V1)
	1 through 65535 (BNA V2)
Default:	1
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The MAXSUBFILES attribute specifies the maximum number of subfiles that may be opened for the file. The subfiles are assigned indices from 1 through the value of MAXSUBFILES, inclusive. If MAXSUBFILES is unspecified and the file is a port file, then MAXSUBFILES is assigned a value of 1 when the file is opened. If the file is opened without specifying an explicit subfile (or 0) when MAXSUBFILES is greater than 1, then an open error is reported.

The value of MAXSUBFILES cannot be decreased even when the file is closed.

MINRECSIZE

Kinds:	all devices
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through MAXRECSIZE
Default:	depends on BLOCKSTRUCTURE
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The MINRECSIZE attribute specifies the minimum size of records in the logical file. If the FRAMESIZE attribute is not 48 (or UNITS is CHARACTERS), then MINRECSIZE is expressed in INTMODE units; otherwise, it is expressed in words. If MINRECSIZE is unspecified or 0, then a default value, which depends upon the value of the attribute MAXRECSIZE, is assigned when the file is opened. If MINRECSIZE is greater than MAXRECSIZE, then it is modified to the value of MAXRECSIZE.

The minimum record size used by the I/O Subsystem for deblocking a file is determined from the maximum value 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 BLOCKSTRUCTURE and FILETYPE attributes).

Files with BLOCKSTRUCTURE not equal to FIXED (where FILETYPE is greater than 0) require that the minimum record size be large enough to contain the link word or record length information.

The value of the SIZEVISIBLE attribute affects the meanings of the MINRECSIZE, MAXRECSIZE, and BLOCKSIZE attributes. When SIZEVISIBLE is FALSE at file creation time, the values of MINRECSIZE and MAXRECSIZE are considered not to include the length of the system-maintained record size field, and are adjusted upward accordingly before being stored in the physical file header. If this adjustment results in MAXRECSIZE being greater than BLOCKSIZE, BLOCKSIZE is also adjusted upward. The values returned to the user when the attributes MINRECSIZE and MAXRECSIZE are interrogated reflect the logical values originally specified before they were adjusted to include the system overhead fields. BLOCKSIZE always returns the actual size of the block being used.

See also
SIZEVISIBLE 296

MYHOST

Kinds:	port
Interrogate:	anytime
Modify:	never
Type:	pointer
Default:	local host name
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The MYHOST attribute returns the name of the local host where the logical file is declared. MYHOST is a pointer-valued attribute that contains the name of the local host at which the process using the port file is executing. It is used during the subfile matching process. The value of MYHOST must match the value of YOURHOST for the complementary subfile. (Refer to the description of the YOURHOST attribute.)

See also

Opening Port Files. 76
YOURHOST. 338

MYHOSTGROUP

Kinds:	port
Interrogate:	anytime
Modify:	never
Type:	pointer
Default:	hostgroup of local host
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The MYHOSTGROUP attribute applies only to BNA V2 port files.

The MYHOSTGROUP attribute returns the group of hosts to which the local host belongs. In port matching, the MYHOSTGROUP of a port must match the YOURHOSTGROUP of the complementary port.

See also

Opening Port Files. 76
YOURHOSTGROUP 339

MYHOSTNAME

Kinds:	port
Interrogate:	anytime
Modify:	never
Type:	pointer
Default:	local host name
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

MYHOSTNAME is the nonpreferred synonym for MYHOST. Refer to the description of the MYHOST attribute.

See also
MYHOST. 237

MYNAME

Kinds:	port
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	(null string)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The MYNAME attribute is a string of up to 100 characters terminated by a period or a null. The MYNAME attribute is used for matching complementary port-subfile descriptions.

The value of MYNAME for the local port file must match the value of YOURNAME for the complementary subfile before the subfile can be opened. (Refer to the description of the YOURNAME attribute and to "Opening Port Files".)

See also

YOURNAME. 340
Opening Port Files. 76

MYUSE

Kinds:	all except port files
Interrogate:	when unassigned (BNA Logical I/O)
	anytime (local host)
Modify:	when unassigned (BNA Logical I/O)
	when closed (local host)
Type:	mnemonic
Range:	CLOSED, IN, IO, OUT
Default:	(see below)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	restricted usage

As of the Mark 3.7 release, MYUSE will be synonymous with FILEUSE.

To specify how the file will be used, FILEUSE should be set to the appropriate value. To aid in the transition, MYUSE will continue to be used if the user has not explicitly set FILEUSE.

The MYUSE attribute specifies how the file is to be opened. The mnemonic values associated with the MYUSE attribute are as follows:

Mnemonic Value	Integer Value
CLOSED	0
IN	1
IO	3
OUT	2

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, then MYUSE is set to IN (unless the assigned peripheral unit (KIND) is an output-only device, in which case MYUSE is set to OUT).

When using the BNA Logical I/O Host Service, MYUSE OUT acts as NEWFILE TRUE; all other values of MYUSE act as NEWFILE FALSE. If NEWFILE is specified, then MYUSE is ignored.

As of the Mark 3.4 release, the MYUSE attribute does not imply a request for the update method of access when the logical file is assigned to a disk file. Only the value of the UPDATEFILE attribute determines the method of access, and a value of IO for the MYUSE attribute specifies only that the file may be used for both input and output. (Refer to the description of the UPDATEFILE attribute.)

The KIND, NEWFILE, and MYUSE attributes determine whether a new file is created or a permanent file is assigned when the file is opened. The NEWFILE attribute overrides any effect that MYUSE has on this process. More specific information is given under "New Files Versus Permanent Files".

For remote files, a MYUSE of OUT means output only. MYUSE specified as IN or IO allows both input and output. For more information, refer to "Remote 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 I/O subsystem.

The BNA Logical I/O Host Service does not allow MYUSE to be modified or interrogated when the file is assigned.

Using the FILEUSE and NEWFILE attributes is recommended in place of using the MYUSE attribute. (Refer to the descriptions of the FILEUSE and NEWFILE attributes.)

See also

New Files Versus Permanent Files. 29
Remote Files. 90

NEWFILE

Kinds:	all except port
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	unspecified, TRUE, FALSE
Default:	unspecified
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The NEWFILE attribute has three states: unspecified, TRUE, and FALSE. If NEWFILE is unspecified and KIND is DISK or PACK, then the value of MYUSE determines whether a permanent file is to be assigned or a new file created. (For more information, refer to "New Files Versus Permanent Files".)

If NEWFILE is TRUE, then a new file is created regardless of the settings of other attributes. If NEWFILE is FALSE, then an existing file is sought regardless 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.

If NEWFILE is specified and disagrees with MYUSE (as when NEWFILE is FALSE and MYUSE is OUT, or when NEWFILE is TRUE and MYUSE is IN or IO), then NEWFILE overrides MYUSE.

The NEWFILE attribute may be read at any time. The value is TRUE if NEWFILE has been modified to TRUE. If NEWFILE is TRUE and the file is opened with a READ statement, then end-of-file (EOF) action takes place (this is analogous to a read on an empty file).

If the attribute has not been specified, then the value returned is FALSE whether or not a new file was created according to the default criteria.

When NEWFILE is TRUE, header labels are written on tape; when NEWFILE is FALSE, header labels are not written.

For tape files, if NEWFILE is TRUE and the file is closed with retention (for example, when using "CLOSE (F,REWIND)" in ALGOL or "CLOSE F" in COBOL), then, when the file is reopened, NEWFILE is still TRUE unless modified after the close and header labels are written that may overwrite an existing file on the tape.

See also

New Files Versus Permanent Files. 29

NEXTRECORD

Kinds:	all except port and remote
Interrogate:	anytime
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The NEXTRECORD attribute returns the current position of the file. In other words, it specifies the record in the file at which a serial read would be performed if that read were the next I/O statement performed on the file. NEXTRECORD functions in a similar manner with regard to a non-Update serial write. (For an explanation of Update I/O action, refer to the description of the UPDATEFILE attribute.)

The NEXTRECORD attribute is well defined for all situations that may arise. In contrast, the RECORD attribute does not return meaningful values in all situations (for example, after a SEEK and after an I/O with "[NO]" as the carriage control).

The counting of records is zero-relative; that is, the first record in the file is record 0. Before I/O has been performed on the file, or after an attempted I/O beyond the end-of-file, NEXTRECORD returns a 0. Any type of close performed on the file resets NEXTRECORD to 0.

The NEXTRECORD attribute is not meaningful for non-disk Direct I/O files.

NORESOURCEWAIT

Kinds:	disk
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The NORESOURCEWAIT attribute applies to Direct I/O files only. It specifies whether or not a program using the file waits when allocating a new area or activating an old one.

If NORESOURCEWAIT is TRUE and disk space is not available or the appropriate continuation pack is not present, then the program receives an immediate error result for that I/O. If the error occurs, then "BUFFER.IORESULT" is 4'101', "BUFFER.IOERRORTYPE" is 5, and "BUFFER.IOEOF" is FALSE.

Regardless of the value of NORESOURCEWAIT, the program may wait if any of the following conditions occur:

a. A "NO FILE" condition occurs on open if the AVAILABLE attribute was not used.

b. The removal of another file with the same title on close or open is required and the the file is protected because the option AUTORM is not set.

c. The file is protected.

d. A close lock on a temporary file causes sectors to be required for the directory.

e. The file is on an interchange pack and the pack goes "NOT READY" after the first area on that pack has been activated but before all areas on that pack have been activated.

f. The file is on an interchange pack and segments are required for the first area of a file header on a given continuation pack.

g. The file is flexible and directory segments are required when stretching the header.

h. A pack goes "NOT READY" during an I/O.

Interrogating either the IORESULT or IOERRORTYPE value is sufficient to detect the error. It is not necessary to interrogate the values of both buffer attributes.

See also

IOERRORTYPE 355
IORESULT. 361

NOTE

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	string
Default:	(null string)
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The NOTE attribute enables the user to print any message on the banner page (card) preceding a file. The string can contain up to 255 characters. Only a non-null string affects printing of the banner page.

If the BANNER attribute is TRUE, a non-null NOTE string is printed in block characters on the banner page.

The default value is a null string.

If PRINTDISPOSITION = DIRECT, the NOTE attribute does not apply and is thus ignored.

OLDYOURUSERCODE

Kinds:	port
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	TRUE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The OLDYOURUSERCODE attribute is used to specify how the YOURUSERCODE attribute will be handled.

See also

YOURUSERCODE. 341

OPEN

Kinds:	all devices
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

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

If the file is closed, then setting OPEN to TRUE opens the file. The open operation is equivalent to the following to ALGOL statements.

For Port Files:

```
OPEN (F[SUBFILE 0],WAIT)
```

For All Others:

```
OPEN (F)
```

If the file is open, then setting OPEN to FALSE closes the file and retains the peripheral unit. This is equivalent to the ALGOL expression "CLOSE (F[SUBFILE 0],WAIT)" (for port files) or "CLOSE (F)" (for all other files).

If the file cannot be opened when an attempt is made to modify the OPEN attribute to TRUE, an attribute error is given. Using an OPEN statement is preferable to setting the OPEN attribute to TRUE, and using a CLOSE statement is preferable to setting the OPEN attribute to FALSE.

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 OPEN attribute cannot be specified in a file declaration or used in file equation.

OPTIONAL

Kinds:	all except port
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The OPTIONAL attribute indicates whether or not (TRUE or FALSE) the assignment of a permanent file is optional. If the permanent file described by the FILENAME and KIND attributes is not present when the file is opened, then a "NO FILE" message is sent to the ODT and the program is suspended. If OPTIONAL is TRUE, then the operator may respond with an OF (Optional File) ODT command. If the operator does so, 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 (File Attribute) or DS (DiScontinue) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of the OF, FA, and DS commands.)

Optional files are read-only. All READ statements result in end-of-file action. Writing to such a file is invalid and causes program termination.

If a permanent file is to be opened only if it is present and available, then an OPEN statement with a type of AVAILABLE should be used instead of the OPTIONAL attribute.

OUTPUTEVENT

Kinds:	port-subfile
Interrogate:	anytime
Modify:	never
Type:	event
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The OUTPUTEVENT attribute returns HAPPENED when output buffers are available and NOT HAPPENED when no output buffers are available. Because OUTPUTEVENT is a read-only attribute, it should not be passed as a parameter to the following statements: ATTACH, CAUSE, CAUSEANDRESET, DETACH, FIX, FREE, LIBERATE, PROCURE, RESET, SET, and WAITANDRESET.

Attribute errors on event-valued attributes result in the termination of the program.

OUTPUTTABLE

Kinds:	all except port
Interrogate:	never
Modify:	anytime
Type:	translatetable
Default:	(no default)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

Setting the OUTPUTTABLE attribute allows the program to modify the I/O Subsystem's output software translation. OUTPUTTABLE is used only when the TRANSLATING attribute is TRUE; otherwise, the translatetable (if one was assigned) is ignored.

A translatetable assigned to OUTPUTTABLE that is declared in a program must be either the first table in a list of tables or the only table in that declaration. Each time the logical file is closed, the value of the OUTPUTTABLE attribute is discarded.

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

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

For further information regarding software translation, refer to the description of the TRANSLATE attribute.

PACKNAME

PACKNAME is a nonpreferred synonym for FAMILYNAME. Refer to the description of the FAMILYNAME attribute.

PAGE

Kinds:	printer
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 65535
Default:	0
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The PAGE attribute indicates the number of the current logical page of a printer file. 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, then PAGE is initialized to 1. As long as PAGESIZE is nonzero and less than 255, then whenever the LINENUM attribute becomes greater than or equal to PAGESIZE, the attribute PAGE is incremented by 1, LINENUM is reset to 1, and the write statement returns an end-of-page result (which has the same format as end-of-file).

The value of the PAGE attribute is not altered by skip-to-channel actions.

The value of PAGE may not be altered by file equation.

PAGESIZE

Kinds:	printer
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	0 through 255
Default:	0
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	restricted usage

The PAGESIZE attribute indicates the number of lines on a logical page.

When PAGESIZE is nonzero for a printer file, the I/O Subsystem maintains the values of the PAGE and LINENUM attributes. When PAGESIZE is 0 for a printer file (this is the default case), the attributes PAGE and LINENUM are ignored.

When PAGESIZE is nonzero, the I/O 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; the I/O Subsystem supplies 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 0. It may be altered to any value between 1 and 255 if it is not 0.

When PAGESIZE is 255, carriage actions are the same as for other nonzero PAGESIZE settings, except that end-of-page is never returned to the program, PAGE is never incremented, and LINENUM is counted continuously, wrapping around from 255 to 0. The automatic skip-to-channel-1 after channel 12, however, is still suppressed.

For remote files, using the SCREENSIZE attribute is preferable to using the PAGESIZE attribute.

PAGESIZE is not supported for remote files across a BNA network. SCREENSIZE may be used instead.

Refer to the description of the SCREENSIZE attribute.

PARITY

Kinds:	paper tape/tape
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	NONSTANDARD, STANDARD
Default:	STANDARD
Stored Permanently:	tape
Parameters:	none
BNA Logical I/O:	supported

The PARITY attribute indicates the parity used on the file.

The mnemonic values and meanings associated with the PARITY attribute are as follows:

Mnemonic Value	Integer Value	Meaning
EVEN (NONSTANDARD)	1	Alpha or even parity
ODD (STANDARD)	0	Binary or odd parity

ODD and EVEN are preferred synonyms for STANDARD and NONSTANDARD, respectively.

PARITY can be modified to EVEN (NONSTANDARD) only for 7-track magnetic tape files (where KIND is TAPE7) and paper tape files. Modifying PARITY to EVEN (NONSTANDARD) on tape files when KIND is TAPE9 or TAPEPE generates a file attribute error when the tape file is opened.

POPULATION

Kinds:	disk/remote
Interrogate:	when assigned (disk) when open (remote)
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The POPULATION attribute returns the number of logical files assigned to a disk file or the number of stations assigned to a remote file.

If the POPULATION of a remote file is not equal to its STATIONCOUNT, then the attributes STATIONSDENIED and DISPOSITION can be used to discover the file's state. (For more information, refer to "Remote Files".)

For remote files, using the STATIONSALLOWED attribute is preferable to using the POPULATION attribute.

See also

Remote Files. 90

PRESENT

Kinds:	all except port
Interrogate:	anytime
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The PRESENT attribute allows a program to open a logical file whenever the permanent file exists or physical unit is available, without requiring the operator to respond to "NO FILE" or "FILE REQUIRED" conditions. If PRESENT returns TRUE, then the file has been opened. If PRESENT returns FALSE, then the file does not exist. If the attribute EXCLUSIVE is TRUE or the program is run in an environment in which disk files are opened exclusively, then 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.

The PRESENT attribute may also return TRUE when no permanent file exists, that is, 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, then it is considered to exist.

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

The PRESENT attribute returns FALSE if the value of SECURITYUSE for a physical file is incompatible with the value of FILEUSE (if FILEUSE is specified) or with the value of MYUSE (if FILEUSE is not specified) for the logical file.

If assigning a logical file to an existing physical file would cause an OPEN ERROR (such as when blocking is incompatible or translation is invalid) then the PRESENT attribute returns FALSE. Where it can be used, an OPEN statement with a type of AVAILABLE, which has the same effect as using the AVAILABLE attribute, is preferable to using the PRESENT attribute.

Example (ALGOL)

```
I := OPEN(F,AVAILABLE);
```

PRINTCHARGE

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	pointer
Default:	(null string)
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The PRINTCHARGE attribute indicates the name (a <simple identifier> up to 17 characters in length) to be used by the Print System when recording information for charging the cost of printing a printer/punch backup file. The default value is inherited from the session or job that issues an explicit print request or from the task for which an automatic print request is generated. Otherwise, the default value is a null string.

The system rules for chargecode validation are applied by the Print System through the USERDATA interface. If chargecode validation is necessary, PRINTCHARGE is validated when the file is opened.

PRINTCOPIES

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	integer
Range:	1 through 1000
Default:	one per destination
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The PRINTCOPIES attribute specifies the number of copies of the file to be printed at each destination specified in the DESTINATION attribute. The value of PRINTCOPIES does not apply to any destination for which the number of copies is specified in the DESTINATION attribute. If no destinations are explicitly specified, the number of copies specified by PRINTCOPIES are printed at the central site.

The default value is one copy per destination. If PRINTDISPOSITION = DIRECT, the PRINTCOPIES attribute does not apply and is thus ignored.

PRINTDISPOSITION

Kinds:	printer/punch
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	DONTPRINT, DIRECT, CLOSE, EOT, EOJ
Default:	EOJ
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The PRINTDISPOSITION attribute indicates whether or not the file should be spooled, and if so, at what point the system should issue an automatic print request for the file. Possible mnemonic values are the following:

DONTPRINT

The file will be spooled, but no request for printing will be issued by the system.

DIRECT

The file is to be printed directly (not spooled) on a printing device. The value DIRECT is effective only at the time the file is opened.

CLOSE

A print request will automatically be issued by the system when the file is closed.

EOT

A print request will automatically be issued by the system when the task that created the file is completed.

EOJ

A print request will automatically be issued by the system at the end of the job that created the file or when the CANDE session for which the file was created is terminated or SPLIT.

The default value is EOJ.

PRINTERCONTROL

Kinds:	printer
Interrogate:	anytime
Modify:	anytime
Type:	pointer
Default:	(null string)
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The value of the PRINTERCONTROL attribute is the title of a file used to control the printer. The file contains, among other things, a default value for the PRINTERKIND attribute. PRINTERCONTROL files have a FILEKIND of PRINTERCONTROLFILE.

The PRINTERCONTROL attribute returns the name of the actual PRINTERCONTROL file being used when the printer file is assigned. The PRINTERCONTROL attribute returns a null identifier (".") when it has not been assigned or when the printer file is assigned and the PRINTERCONTROL file is being ignored. When the printer file is unassigned, the PRINTERCONTROL attribute returns the most recent value assigned to it.

When the printer file is opened, the system will attempt to open the associated PRINTERCONTROL file. This attempt may cause the program to wait on a "NO FILE" RSVP. The operator can respond with the ODT commands OF, DS, OK, FA, UL, or IL. The responses OF, FA, IL, and UL affect only the PRINTERCONTROL file and not the printer file it is associated with. Entering DS terminates the program attempting to open the printer file. The OF response causes the PRINTERCONTROL file to be ignored.

The PRINTERCONTROL file is ignored in the following circumstances:

1. The PRINTERKIND specified in the PRINTERCONTROL file is incompatible with that assigned explicitly by the user.

2. The VERSION of the PRINTERCONTROL file is incompatible with that of the MCP or other system software.

3. An I/O error occurs while processing the PRINTERCONTROL file.

4. The PRINTERCONTROL file could not be found and the operator entered OF, or the printer file was file-equated or FAed so as to ignore the PRINTERCONTROL file.

An appropriate message is displayed when the PRINTERCONTROL file is ignored.

Examples

The following are examples of assigning a value to the PRINTERCONTROL attribute in an ALGOL file declaration.

```
FILE F(KIND=PRINTER,PRINTERCONTROL="XYZ.");
FILE F(KIND=PRINTER,PRINTERCONTROL="(X)XYZ.");
FILE F(KIND=PRINTER,PRINTERCONTROL="(X)XYZ ON ABC.");
```

The following are examples of changing the value of the PRINTERCONTROL attribute in an ALGOL program.

```
REPLACE F.PRINTERCONTROL BY "ABC.";
REPLACE F.PRINTERCONTROL BY "(A)ABC/BBB/CCC ON XYX.";
```

PRINTERKIND

Kinds:	printer
Interrogate:	anytime
Modify:	anytime
Type:	mnemonic
Range:	DONTCARE, IMAGEPRINTER, LINEPRINTER
Default:	DONTCARE
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The PRINTERKIND attribute indicates the kind of device the file is to be printed on. The mnemonic IP is a synonym for IMAGEPRINTER, while the mnemonic LP is a synonym for LINEPRINTER.

The value of the PRINTERKIND attribute, if assigned explicitly, overrides the default value of the PRINTERKIND attribute contained in the PRINTERCONTROL file (if any).

All PRINTERCONTROL files created by the Forms Manager specify a PRINTERKIND of IMAGEPRINTER.

This attribute enables the user to indicate the kind of device to be used to print the file, without concern for which specific device will service the print request. For example, the user may have a file that is intended to use the features of an image printer; the user may not know which device is an image printer, but does know the kind of device needed.

The PRINTERKIND attribute returns the value corresponding to the type of printer that is assigned. When no printer is assigned, the PRINTERKIND attribute returns the last value that was assigned to it.

The default value is DONTCARE.

Examples

The following are examples of assigning a value to the PRINTERKIND attribute in an ALGOL file declaration.

```
FILE F(KIND=PRINTER,PRINTERKIND=IMAGEPRINTER);
FILE F(KIND=PRINTER,PRINTERKIND=LINEPRINTER);
FILE F(KIND=PRINTER,PRINTERKIND=DONTCARE);
```

The following are examples of changing the value of the PRINTERKIND attribute in an ALGOL program.

```
F.PRINTERKIND := VALUE(LINEPRINTER);
F.PRINTERKIND := VALUE(IMAGEPRINTER);
F.PRINTERKIND := VALUE(DONTCARE);
```

PROTECTION

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	PROTECTED, SAVE, TEMPORARY
Default:	TEMPORARY
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted usage

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

Mnemonic Value	Integer Value
PROTECTED	2
SAVE	1
TEMPORARY	3

If PROTECTION is 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 that makes it possible to discover the last valid block written on that area in the event of a Halt/Load. The PROTECTION of a physical file cannot be modified by means of the PROTECTION attribute once that file has been created.

After a Halt/Load, if the system notices a tape file that is uptape and whose PROTECTION attribute is PROTECTED, a tape mark is written on the tape before it is rewound, thus making the file readable after the system is reinitialized.

If PROTECTION is SAVE, then 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. However, no special action is taken to ensure that the correct end-of-file pointer is maintained across a system failure, so blocks written to the file may not be recoverable after the Halt/Load. PROTECTION equal to SAVE is meaningless for tape files.

If PROTECTION is TEMPORARY and a disk file is being created, then that 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, then an entry is made in the directory and the file becomes permanent. If the file is a tape file, then the value TEMPORARY means that no special action is performed. The disposition of the tape file is determined by the value of the SAVEFACTOR attribute. (Refer to the description of the SAVEFACTOR attribute.)

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

PROTECTION is not supported by the BNA Host Logical I/O Service for tape files.

RECEPTIONS

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The RECEPTIONS attribute indicates the number of messages received from the station when the Relative Station Number (RSN) is specified as a parameter or indicates the number of messages received from all the stations of the remote file when the parameter is omitted.

See also
Relative Station Number 91

RECORD

Kinds:	all devices except port and remote
Interrogate:	anytime
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The RECORD attribute returns the current position in the file. When the file is open but no I/O has been performed upon the file, or was opened with a READ[NO] operation, or is closed, RECORD returns -1.

The RECORD attribute is not meaningful for non-disk Direct I/O files.

For open direct disk files, the RECORD attribute reports the NEXTRECORD value. NEXTRECORD performs essentially the same functions as the RECORD attribute, except that it returns meaningful values for many more situations than does RECORD.

The use of NEXTRECORD is preferred to the use of RECORD. The description of the NEXTRECORD attribute provides details about the effective use of the attribute.

The counting of the RECORD attribute is zero-relative; that is, at the first record in the file, RECORD is 0. When reading in reverse, the RECORD attribute starts at 0, the same as when reading forward.

RECORDINERROR

Kinds:	disk/tape/diskette
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The RECORDINERROR attribute returns the record number (zero-relative) of the first record in the block that is contained in the buffer currently in use for data transfer to or from the program. Refer to the description of the IOINERROR attribute.

REEL

REEL is a nonpreferred synonym for FILESECTION. Refer to the description of the FILESECTION attribute.

REQUESTEDMAXRECSIZE

Kinds:	port
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 65535
Default:	see below
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The REQUESTEDMAXRECSIZE attribute applies only to BNA V2 port files.

The REQUESTEDMAXRECSIZE file attribute specifies the maximum text size desired for each subfile. If FRAMESIZE is not 48 (or UNITS is CHARACTERS), REQUESTEDMAXREQSIZE is expressed in INTMODE units; otherwise, it is expressed in words. The default is 320 words or 1920 INTMODE units. Non-A Series systems may have a different default and maximum value. The maximum will always be at least 2000 characters.

The function of this attribute is equivalent to that of MAXRECSIZE with no specified subfile index.

See also

ACTUALMAXRECSIZE. 108
FRAMESIZE 206
MAXRECSIZE. 232
Port File I/O Operations. 84

RESIDENT

Kinds:	all except port
Interrogate:	anytime
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The RESIDENT attribute 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 assigned or the file exists, then RESIDENT returns TRUE. The RESIDENT attribute is unaffected by the setting of the EXCLUSIVE attribute. Whether or not a disk file is in use exclusively by another program, RESIDENT returns TRUE if the file exists. Whether or not a disk file exists, RESIDENT returns TRUE if opening the file causes the successful creation of a new file (for example, if NEWFILE is TRUE).

If RESIDENT returns TRUE and DEPENDENTSPECS is TRUE (or FILETYPE is 7 or 8), then the other attributes that describe the permanent file, that is, BLOCKSIZE, EXTMODE, INTMODE (only when FILETYPE is 7), MAXRECSIZE, MINRECSIZE, SIZEMODE, SIZEOFFSET, SIZE2, and UNITS 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 descriptions of the AVAILABLE and PRESENT attributes.

RESIDENT returns 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 RESIDENT attribute returns FALSE if the value of SECURITYUSE for a physical file is incompatible with the value of FILEUSE (if FILEUSE is specified) or with the value of MYUSE (if FILEUSE is not specified) for the logical file.

ROWADDRESS

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	disk
Parameters:	1 required, 1 optional
BNA Logical I/O:	not supported

The ROWADDRESS attribute returns the physical disk address of an area of a disk file.

The value returned by the ROWADDRESS attribute contains the FAMILYINDEX in the field [41:08] and the segment address in [32:33]. A value of 0 is returned if the row is not assigned. To find out only whether or not the area is allocated, using the AREAALLOCATED attribute is preferable to using the ROWADDRESS attribute.

In Mark 3.5 and earlier releases, the FAMILYINDEX was in the field [29:08] of the value returned by ROWADDRESS, and the segment address was in the field [21:22]. Because of these changes, the Mark 3.5 and Mark 3.6 releases of the MCP issue run-time warnings for all programs that use the ROWADDRESS attribute.

The ROWADDRESS attribute requires a parameter, the row number. If the disk file is DUPLICATED, then the copy number is also required as a parameter.

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

Examples (ALGOL)

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

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

ROWSINUSE

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 1000
Default:	not applicable
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The ROWSINUSE attribute returns the actual number of AREAs allocated for a disk file. (Refer to the description of the AREAS attribute.) Because disk areas are allocated only when needed, it is possible, using random I/O, to create a disk file in which areas in the logical middle of the file are never referenced and hence never allocated.

SAVEBACKUPFILE

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

When TRUE, the SAVEBACKUPFILE attribute prevents the printer backup file from being removed after it is printed.

When SAVEBACKUPFILE is FALSE, the backup file will be removed after it is printed by the Print System (unless the file is skipped; see the PS SKIP command in the Print System (PrintS/ReprintS) User's Guide). If the value of PRINTCOPIES (or the number of copies derived from the DESTINATION attribute) is greater than 1, then the file is not removed until the last copy of it has been printed.

If a WFL "PRINT" statement is used to print the file and if SAVEBACKUP is not FALSE, the backup file is saved after printing.

The default value is FALSE.

This attribute does not apply to backup files on tape. This attribute is also ignored if used with non-backup files.

SAVEFACTOR

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 999
Default:	0
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The SAVEFACTOR attribute indicates the expiration date of a file in terms of the number of days past the creation date. When an expired tape is mounted with a write ring, it is identified as scratch but not purged. This is also the case when a tape file is created without setting SAVEFACTOR because the default value is 0.

The SAVEFACTOR attribute has no system-defined meaning for a disk file. However, when the disk file is the output file of a SORT procedure, the SORT procedure locks the file if SAVEFACTOR is nonzero; otherwise, it closes the file with release.

The SAVEFACTOR attribute affects the functioning of CLOSE WITH RELEASE and BLOCKEXIT-CLOSE for diskette files. A SAVEFACTOR value of 0 for these CLOSE calls causes the files to be removed.

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

See also

Diskette Files. 53

SCREEN

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

The SCREEN attribute 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.

See also

Relative Station Number 91

SCREENSIZE

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

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 (ALGOL)

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

See also
Relative Station Number 91

SECURITYGUARD

Kinds:	disk
Interrogate:	anytime
Modify:	anytime
Type:	pointer
Default:	(null string)
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	restricted values

The SECURITYGUARD attribute identifies the guard file if GUARDED or CONTROLLED security is to be invoked for the file. Refer to the discussion of the GUARDFILE utility program in the System Software Utilities Reference Manual.

For a physical file with a corresponding logical file, the SECURITYGUARD attribute returns the TITLE of the guard file in the following cases:

a. When the user is privileged

b. When the user is a non-privileged user and owner of the physical file

c. When the program accessing the physical file is privileged

In all other cases, a null string is returned.

If a logical file has not been assigned to the physical file and a guard file has been specified (the SECURITYGUARD attribute specified) for the logical file, then the name of the specified guard file is returned.

The format of the SECURITYGUARD attribute is that of an external file title. (For more information, refer to the description of 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, then the guard file of the physical file is changed. Changing the guard file of the physical file may be done only by the owner of the physical file or by 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 when 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 cannot be found on the specified pack, then SECURITYTYPE of PRIVATE is assumed instead of GUARDED or CONTROLLED. If the specified pack for the guard file is not present, the system message "REQUIRES PK" is displayed (unless the opening program is an MCS, in which case the open attempt fails with a FILE NOT AVAILABLE condition). Appropriate operator action is required to either continue or terminate the process of accessing the permanent disk file.

The BNA Logical I/O Host Service allows guard files to grant access only by usercode; that is, the access specification in the guard file must be of the following form:

```
USERCODE X;
```

If the access specification is in either of the two following forms, access to the file will not be granted to the specified program:

```
PROGRAM Y;
USERCODE X USING PROGRAM Y;
```

The BNA Logical I/O Host Service imposes the same restrictions on SECURITYGUARD that it imposes on FILENAME.

SECURITYTYPE

```
                Kinds: disk/port
          Interrogate: anytime
               Modify: anytime (disk)
                       when closed (port)
                 Type: mnemonic
                Range: CONTROLLED, GUARDED, PRIVATE, PUBLIC
              Default: (see below)
  Stored Permanently: disk
           Parameters: none
     BNA Logical I/O: restricted values
```

The SECURITYTYPE attribute specifies who (as identified by usercode) may access a file. The mnemonic values and meanings associated with SECURITYTYPE are as follows:

Mnemonic Value	Integer Value	Meaning
CONTROLLED	3	Identical to GUARDED but with the added restriction that even the owner (if nonprivileged) requires permission from the guard file before he may access the file.
GUARDED	2	Requires permission from the guard file (identified by the SECURITYGUARD attribute) before a nonprivileged user, other than the owner, may access the file.
PRIVATE	0	For files other than port files, this implies that only a privileged user or the owner may access the file.
PUBLIC	1	Allows access to any user who references the file using the (USERCODE)TITLE form of the file title.

CLASSA and CLASSB are nonpreferred synonyms for PUBLIC and GUARDED, respectively.

GUARDED, PRIVATE, and PUBLIC are supported by the BNA Logical I/O Host Service; CONTROLLED is not supported by the BNA Logical I/O Host Service.

The default value of SECURITYTYPE is PRIVATE if the file is created under a usercode (for example, all CANDE-created files) and for all port files; otherwise, it is PUBLIC.

The only valid values of SECURITYTYPE for port files are PRIVATE and PUBLIC. If SECURITYTYPE is PRIVATE, the YOURUSERCODE attribute of the port-subfile must match the usercode of the task offering the complementary subfile. (Refer to the description of the YOURUSERCODE attribute.) If SECURITYTYPE is PUBLIC, usercodes are not used in matching. (Refer to "Opening Port Files".)

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

See also

Opening Port Files. 76
USERBACKUPNAME. 332

SECURITYUSE

```
                    Kinds: disk
             Interrogate: anytime
                  Modify: anytime
                    Type: mnemonic
                   Range: IN, IO, OUT, SECURED
                 Default: IO
Stored Permanently: disk
              Parameters: none
   BNA Logical I/O: supported
```

The SECURITYUSE attribute specifies the manner in which a disk file that is protected by security may be accessed by nonprivileged users using nonprivileged programs. SECURITYUSE is ignored if a guard file is invoked. The mnemonic values and meanings associated with SECURITYUSE are as follows:

Mnemonic Value	Integer Value	Meaning
IN	1	Indicates that users have read-only access to the file.
IO	3	Indicates that users have read/write access to the file.
OUT	2	Indicates that users have write-only access to the file.
SECURED	0	Implies that no user may access the file, with the exception that if it is a code file, it may be executed.

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

SENSITIVEDATA

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

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. (Refer to the description of the PROTECTION attribute.)

SERIALNO

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	(see below)
Type:	word
Range:	(see below)
Default:	(no default)
Stored Permanently:	disk/tape
Parameters:	1 optional
BNA Logical I/O:	restricted usage

The SERIALNO attribute 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 at the ODT through the SN (Serial Number) ODT command. The RC (Reconfigure Disk) or LB (ReLaBel Pack) ODT commands can be used to establish a serial number for a disk peripheral. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

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, or an integer in the range of 1 through 999999.

The SERIALNO attribute can be modified for the current volume only when the file is closed and unassigned. When modifying this attribute, an optional parameter 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. The following WFL file equation statement is an example:

```
FILE F(SERIALNO=("FIRST","SECOND",,"FOURTH"));
```

SERIALNO can be read at any time. When the logical file is open or closed with retention (that is, the logical file is assigned to a physical file), the serial number of the physical volume (for tape) or base unit (for disk) 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 non-disk, 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, then 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.

Specifying SERIALNO for a particular tape volume indicates that checking for serial numbers is requested when the file is assigned. For permanent tape files, the serial number of the physical tape must match the SERIALNO attribute's value. (For more information, refer to "File Assignment".)

When creating a tape file, if SERIALNO is specified, file assignment succeeds only if the serial numbers match. The tape must have a write ring and cannot be locked, saved, assigned, or not ready. If the tape is not SCRATCH, the tape is rewound so that new labels can be written at the beginning of the volume (in effect purging the tape). Setting SERIALNO to all null characters (where all bits are equal to 0) indicates that the attribute is not to be considered during file assignment.

When using the BNA Logical I/O Host Service, if SERIALNO is specified before the file is opened, serial numbers for volumes other than the first volume are ignored.

An optional FILESECTION number (in ALGOL, "F(FS_NUMBER).SERIALNO") may be specified for multivolume files. The attribute may be modified while the logical file is assigned as long as the element in the SERIALNO list as indicated by the FILESECTION number (one-relative) is not the FILESECTION currently in use.

If a permanent tape file is found that meets all the requirements except for the serial number, then an "UNMATCHED SERIALNO" notification is given to the operator. The operator can respond by making a tape with the matching SERIALNO available or by using the IL (Ignore Label), OF (Optional File), FA (File Attribute), or DS (DiScontinue) ODT commands. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

On output, the "FILE REQUIRES" message is displayed with the desired serial number in brackets. The operator may respond with a DS (DiScontinue), OU (Output Operator), or SN (Serial Number) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.)

File assignment to a diskette requires use of the SERIALNO attribute. Diskette serial numbers are handled in the same manner as magnetic tape serial numbers. When a multivolume file is going be to accessed (or created), a list of serial numbers should be supplied.

The RESIDENT and PRESENT attributes return FALSE on "UNMATCHED SERIALNO" conditions. The AVAILABLE attribute returns a 6 on an "UNMATCHED SERIALNO" condition.

The MCP run-time option SERIALNUMBER (option 24) inhibits the assignment of scratch tapes unless the serial numbers match. For further information about this option, refer to the OP (OPtions) ODT command in the Operator Display Terminal (ODT) Reference Manual.

The SERIALNO attribute may be assigned a value by means of a program 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 fill

c. 0 (the null value for SERIALNO)

The following ALGOL and WFL texts 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, because ALGOL short strings are right-justified with NULL fill and thus have integer values.

See also
File Assignment 28

SINGLEPACK

SINGLEPACK is a nonpreferred synonym for SINGLEUNIT. Refer to the description of the SINGLEUNIT attribute.

SINGLEUNIT

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The SINGLEUNIT attribute 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, then they are inserted into the rotational scheme of allocation.

When space is unavailable on the original unit, the system message "REQUIRES FAMILYINDEX(N) SECTORS" is displayed. At this point, the OK (Reactivate) ODT command may be used to override the family index requirement, allowing the file to overflow to another unit in the family, or the DS (DiScontinue) ODT command may be used to terminate the program. (Refer to the Operator Display Terminal (ODT) Reference Manual for descriptions of these commands.) The OK command results in a system message indicating the FAMILYINDEX on which space was found:

SECTORS FOUND ON FAMILYINDEX (M)

SIZEMODE

Kinds:	all except port
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	ASCII, BCL, EBCDIC, HEX, SINGLE
Default:	SINGLE
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The SIZEMODE attribute 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 4.

The mnemonic values and meanings associated with SIZEMODE are as follows:

Mnemonic Value	Integer Value	Meaning
ASCII	5	8-bit
BCL	3	6-bit
EBCDIC	4	8-bit
HEX	2	4-bit, packed decimal
SINGLE	0	Word mode, binary, or 48-bit

Although this attribute is allowed for all devices, a value different from INTMODE or a value of SINGLE is meaningful only on devices that can support a complex record format (for example, disk and tape).

SIZEOFFSET

Kinds:	all except port
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 65535
Default:	0
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The SIZEOFFSET attribute indicates the zero-relative offset from the beginning of the record where the record length information is contained. The SIZEOFFSET attribute, along with the SIZEMODE and SIZE2 attributes, is used with files of FILETYPE 4. The value of the SIZEOFFSET attribute is specified in SIZEMODE units.

SIZEVISIBLE

Kinds:	all
Interrogate:	anytime
Modify:	when unassigned
Type:	Boolean
Range:	TRUE, FALSE
Default:	TRUE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The attribute SIZEVISIBLE specifies whether or not the size field of a file whose BLOCKSTRUCTURE = VARIABLE is visible within the record.

If the size field is visible (SIZEVISIBLE = TRUE), then the first four characters of each record must contain a decimal number indicating the number of INTMODE-sized characters in the record. The number itself is encoded using the INTMODE character set. For example, if a record is 50 characters long and INTMODE is EBCDIC, the size field would contain "0050" in EBCDIC characters. If INTMODE = SINGLE, then the size is stored in the first word of the record in binary. The user program is responsible for maintaining the size field.

If the size field is invisible (SIZEVISIBLE = FALSE), then the records contain no size field. Instead, the MCP maintains the sizes of the records. On a WRITE statement, the size of the record is set to the length of the data written. On a READ statement, the size of the record read is returned in the logical result descriptor (see the description of the STATE attribute for details).

Attempting to write a record whose size cannot be represented in the size field results in a data error result. Any positive integer can be represented when INTMODE = SINGLE. For other INTMODEs, the maximum value that can be represented in the size field is 9999 if SIZEVISIBLE is TRUE and 9995 if SIZEVISIBLE is FALSE. The minimum value that can be represented is 4 if SIZEVISIBLE is TRUE and 0 if SIZEVISIBLE is FALSE.

When SIZEVISIBLE is FALSE, FRAMESIZE is 48 (or UNITS is WORDS), and INTMODE is not SINGLE, the system is unable to correctly maintain the sizes of the records. When a file is created with this combination of the SIZEVISIBLE, FRAMESIZE, and INTMODE attributes, the error "SIZEVISIBLE/FRAMESIZE/INTMODE CONFLICT" occurs when an attempt is made to open that file.

The value of the SIZEVISIBLE attribute affects the meanings of the MINRECSIZE, MAXRECSIZE, and BLOCKSIZE attributes. When SIZEVISIBLE is FALSE at file creation time, the values of MINRECSIZE and MAXRECSIZE are considered not to include the length of the system-maintained record size field, and are adjusted upward accordingly before being stored in the physical file header. If this adjustment results in MAXRECSIZE being greater than BLOCKSIZE, BLOCKSIZE is also adjusted upward. The values returned to the user when the attributes MINRECSIZE and MAXRECSIZE are interrogated reflect the logical values originally specified before they were adjusted to include the system overhead fields. BLOCKSIZE always returns the actual size of the block being used.

The length of the system-maintained size field is the same as the length that must be maintained by the user when SIZEVISIBLE is TRUE. (See the discussion of BLOCKSTRUCTURE = VARIABLE.)

Files can be created with SIZEVISIBLE = TRUE and then read or updated with SIZEVISIBLE = FALSE and vice versa.

See also

BLOCKSTRUCTURE. 135
INTMODE 213
MAXRECSIZE. 232
MINRECSIZE. 236
STATE 299

SIZE2

Kinds:	all except port
Interrogate:	anytime
Modify:	when closed
Type:	integer
Range:	0 through 23
Default:	0
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The SIZE2 attribute indicates the length, in SIZEMODE units, of the field in the variable-length record pointed to by the SIZEOFFSET attribute. The SIZE2 attribute is used with files of FILETYPE 4. If SIZEMODE is SINGLE (words), then a value of 1 is used for SIZE2.

STATE

Kinds:	all devices
Interrogate:	when assigned
Modify:	never
Type:	word
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	restricted usage

The STATE attribute 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. Refer to the description of the IORESULT attribute for information relating to Direct 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 error information in field [16:16] is nonzero.
[16:16]	Error information is as follows:

Field	Contents
[1:1]	The attention bit for duplicated files. (Refer to "Duplicated Files".)
[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 occurred (size error for variable-length records).
[5:1]	If this bit is on, a deleted or duplicated record was referenced in a file whose FILEORGANIZATION was not NOTRESTRICTED.

Field	Contents
[6:1]	For files with FILEORGANIZATION of INDEXED or INDEXEDNOTRESTRICTED, a primary key error occurred for sequential writes. This bit indicates that the primary keys are not in sequential order. For rewrites and deletes, this bit indicates that the primary keys are not equal.
[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 exception field ([0:1]) being turned on.
[13:1]	If this bit is on, the remote file has encountered a break on output.
[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 remote files, indicating the reason for EOF (refer also to "Opening Remote Files"):

Value	Meaning
1	File Assignment denied
2	File assignment postponed
3	Illegal file use

[27:16]	These bits indicate the duplicated disk file error mark. Note that bit 12 is copy number 1, bit 13 is copy number 2, and so forth. Bit 27 is unused. (For further information, refer to the description of the COPYINERROR attribute.)
[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. This value is undefined if the previous I/O was a SEEK. This field corresponds to the value of the CURRENTRECORD attribute.

When using the BNA Logical I/O Host Service, error results for WRITE statements are reported one WRITE after the WRITE statement that reuses the buffer that originally had the error. That is, in this case, the error is reported one buffer later than normal. (Normally, error results are reported exactly at the WRITE that reuses the buffer having the error).

See also

Opening Remote Files. 92
IORESULT. 361
Duplicated Files. 373

STATIONCOUNT

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The STATIONCOUNT attribute returns the number of stations in the remote file's STATIONLIST. 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 remote file's description in the Network Information File (NIF). If the STATIONLIST attribute has been used to subtract stations from the file, then the largest valid Relative Station Number (RSN) may be larger than the STATIONCOUNT.

For further information regarding RSNs, refer to "Relative Station Number" in "Remote Files".

See also

Remote Files. 90
Relative Station Number 91

STATIONLIST

Kinds:	remote
Interrogate:	never
Modify:	when open
Type:	pointer
Default:	name corresponding to the task attribute STATION
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The STATIONLIST attribute may be used to add or subtract a station or set of stations (the family of a file described in the DATACOMINFO file or the Network Information File (NIF)) to or from a remote file. The format used with the STATIONLIST attribute is that of an external file name. Refer to the descriptions of the STATIONNAME and FILENAME attributes.

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

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

COBOL:    CHANGE ATTRIBUTE STATIONLIST OF FILEID UP BY "AAA1.".
          CHANGE ATTRIBUTE STATIONLIST OF FILEID DOWN BY
          "BBB2.".
```

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

Attempting to add or subtract a station or file not found in the NIF causes an attribute error. Subtracting all stations from the file does not close the file; however, subsequent I/O statements while STATIONCOUNT is 0 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 that station, and all knowledge of that station from the logical file's point of view is lost. When a station is added to a remote file's family, the STATIONCOUNT is increased and the controlling MCS is sent a "FILE OPEN" message for that station.

When a station is subtracted from the STATIONLIST, the Relative Station Numbers (RSNs) for all other stations in the list remain the same. If a write operation is directed to an RSN corresponding to a station that has been subtracted from the list, then an end-of-file result is returned. (For a full explanation of RSNs, refer to "Relative Station Number" under "Remote Files".) Information about how to reference the individual stations of a remote file is given as part of the description of the LASTSUBFILE attribute.

When a station is added to the STATIONLIST of a remote file, the LASTSUBFILE attribute is updated to be the RSN of the newly added station (if no attribute error occurred during the addition). In this way, a user program can determine the RSN of the newly added station for future use.

The value of STATIONLIST may not be altered by file equation.

See also
Relative Station Number 91

STATIONNAME

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	pointer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

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

Example (ALGOL)

```
REPLACE POINTER (A) BY REMOTFYL(THIS_RSN).STATIONNAME;
```

See also
Relative Station Number 91

STATIONSALLOWED

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

The STATIONSALLOWED attribute returns the number of stations assigned to a remote file. If STATIONSALLOWED for a remote file is not equal to the file's STATIONCOUNT, then the STATIONSDENIED and DISPOSITION attributes can be used to determine the file's state. (Refer to the description of these attributes.)

STATIONSDENIED

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

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

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

SUBFILEERROR

Kinds:	port-subfile
Interrogate:	anytime
Modify:	never
Type:	mnemonic
Range:	(see below)
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	not supported

The SUBFILEERROR attribute returns an integer value corresponding to the errors, if any, that 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 mnemonic values and meanings associated with the SUBFILEERROR attribute are as follows:

Mnemonic Value	Integer Value	Meaning
NOERROR	0	No error occurred.
DISCONNECTED	1	Communication with the corresponding port-subfile was lost.
DATALOST	2	All data may not have been successfully transmitted before the port-subfile was closed.
NOBUFFER	3	A write with the DONTWAIT option failed because no buffer space was available.
NOFILEFOUND	4	An open operation failed because a matching port-subfile was not found. (The result from the OPEN function or AVAILABLE attribute was 2.)

Mnemonic Value	Integer Value	Meaning
UNREACHABLEHOST	5	The foreign host became unreachable during the open operation. (The result from the OPEN function or AVAILABLE attribute was 38.)
UNSUPPORTEDFUNCTION	6	An attempted open on this port-subfile resulted in a request for an unsupported function.
UNSUPPORTED-PROTOCOLTYPE	7	An attempted open on this port-subfile resulted in incompatible protocol types. Applies to BNA V2 port files only.
PROTOCOLERROR	8	An attempted open on this port-subfile resulted in detection of an error in the involved protocol. Applies to BNA V2 port files only.
LACKOFRESOURCES	9	An attempted open on this port-subfile failed due to a lack of resources. Applies to BNA V2 port files only.
HOSTNOTINHOSTGROUP	10	An attempted open on this port-subfile failed because the YOURHOST specification is not a legal member of the specified YOURHOSTGROUP. Applies to BNA V2 port files only.
UNAUTHORIZEDFOR-APPLICATIONGROUP	11	An attempted open on this port-subfile failed because the user is not an authorized user of the specified application group. Applies to BNA V2 port files only.
BNAIPCUNAVAILABLE	12	An attempted open on this port-subfile failed because the BNA IPC service is presently unavailable. Applies to BNA V2 port files only.

Mnemonic Value	Integer Value	Meaning
BADATTRIBUTESFOROPEN	13	An attempted open on this port-subfile failed because one or more of the attributes affecting the open process either has illegal syntax or has an invalid value. Applies to BNA V2 port files only.
UNAVAILABLEFUNCTION	14	An attempted open on this port-subfile failed because the service requested in unavailable. Applies to BNA V2 port files only.
UNSUPPORTEDINTMODE	15	An attempted open on this port-subfile failed because the INTMODE value specified is unsupported on the remote host specified by YOURHOST. Applies to BNA V2 port files only.
NETWORKINGNOT-SUPPORTED	16	An attempted open on this port-subfile failed because the networking service is not supported on this host. Applies to BNA V2 port files only.

Refer to the descriptions of the AVAILABLE and MAXSUBFILES attributes for related information.

TANKING

Kinds:	remote
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	UNSPECIFIED, NONE, SYNC, ASYNC
Default:	UNSPECIFIED
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The TANKING attribute is associated with remote output only. TANKING may be used to specify or deny output tanking or leave the decision to the MCS. The mnemonic values associated with the TANKING attribute are as follows:

Mnemonic Value	Integer Value	Meaning
ASYNC	3	Causes 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 remote output continues to be detanked and transmitted until it is exhausted. The associated Message Control System (MCS) cannot override ASYNC.
NONE	1	Causes no tanking to occur and prevents the MCS from specifying tanking.
SYNC	2	Causes the file to be tanked. In addition, the task is unable to continue past file close, for the file specifying SYNC, until all tanked output has been transmitted. The associated MCS cannot override SYNC.
UNSPECIFIED	0	Causes tanking not to occur unless overridden by the MCS at the time the file is assigned.

Causing break-on-output for a file having tanked output results in the currently tanked output for the file being discarded.

TAPEREELRECORD

Kinds:	tape
Interrogate:	when assigned
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	tape
Parameters:	none
BNA Logical I/O:	supported

The TAPEREELRECORD attribute returns the record number of the last record read or written, relative to the first record on the current volume of the tape file. For example, after reading the first record of the second volume of a tape file, TAPEREELRECORD returns 0. TAPEREELRECORD is valid only for labeled tape files. The value returned before the first I/O is -1.

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

TIMELIMIT

Kinds:	remote
Interrogate:	anytime
Modify:	anytime
Type:	real
Range:	0 through 549755813887
Default:	0
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	see below

The TIMELIMIT attribute is a positive real number that specifies how long, in seconds, the I/O Subsystem is to wait for an I/O to occur. The BNA Logical I/O Host Service, however, only supports integer values. If specified as real, the value is rounded off to the nearest integer. The attribute can be modified in read or write statements (in ALGOL using the "[TIMELIMIT <arithmetic expression>]" form of the "[<record number or carriage control>]" part of the I/O statement).

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

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

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

TITLE

Kinds:	all devices (except remote)
Interrogate:	anytime
Modify:	anytime (disk) when closed (other devices)
Type:	pointer
Default:	FILENAME
Stored Permanently:	disk/tape
Parameters:	1 optional
BNA Logical I/O:	restricted usage/values

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

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

1. The FILENAME attribute is set to <file name>.

2. The FAMILYNAME attribute is set to <family name>.

3. The KIND attribute is set to PACK.

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

The TITLE attribute for a disk file or tape file returns the name of the physical file whenever one is assigned to the logical file. The TITLE returned for an open and unassigned file or a tape file that is positioned beyond the last file on the tape is a null string. 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 <family name>" phrase. The default family name is "DISK".

When a printer or punch backup file is open and assigned to a physical file, the TITLE attribute returns the title of the physical file. When the file is closed, the TITLE attribute returns the value that has been assigned by the user program. For example, the following ALGOL program fragment will return "PRINTFILE." when the file is closed:

```
FILE LINE(KIND=PRINTER,TITLE="PRINTFILE.")
ARRAY A[0:44];

REPLACE POINTER(A) BY LINE.TITLE;
```

When the file is open and assigned to backup on pack, the fragment above will respond in the following form:

```
*BD/000<job #>/000<mix #>/001<file name> ON PACK.
```

Note that the TITLE attribute can include an "ON <family name>" phrase but that the FILENAME attribute never can.

The TITLE attribute may be changed when the file is assigned only if the logical file is assigned to a disk file. Under this condition, changing the TITLE of the logical file causes the file name of the physical disk file to be changed also. If a family name is supplied in this case, it must match the FAMILYNAME of the physical file. When changing the file name of a disk file, using the FILENAME attribute is preferable to using the TITLE attribute.

When using the BNA Logical I/O Host Service, the TITLE of a file may not be changed while the file is assigned, but the FILENAME may be. The BNA Logical I/O Host Service imposes the same restrictions on TITLE that it imposes on FILENAME. (Refer to the description of the FILENAME attribute.)

The construct "<file name> on TAPE" is useful for accessing tape files; when this construct is used, the I/O Subsystem sets the value of the KIND attribute to TAPE.

For a duplicated disk file whose TITLE is AAA, the file names of the individual files would be as follows: AAA/"COPY#01", AAA/"COPY#02", and so on. For more information about duplicated files, refer to "Duplicated Files".

For port files, the FILENAME attribute is preferred over TITLE. In addition, a port file TITLE must be a <simple identifier>.

The TITLE attribute is accepted as a synonym for STATIONNAME for REMOTE files. Use of the STATIONNAME attribute is preferred to the use of the TITLE attribute for REMOTE files.

For the special conventions used for the naming of tape files, refer to "Magnetic Tape Files".

For the special conventions used for the naming of files on INTERCHANGE packs, refer to "Interchange Packs" under "Disk Files".

See also

TRAINID

Kinds: printer
Interrogate: anytime
Modify: when closed
Type: mnemonic
Range: (see below)
Default: NOID
Stored Permanently: no
Parameters: none
BNA Logical I/O: supported

The TRAINID attribute is used to specify a particular print train for a train printer. When this attribute is modified to a value other than NOID, the translatetable is loaded, as needed, from the file SYSTEM/TRAINTABLES and the MCP waits until the selected print train is made available.

The mnemonic values and meanings associated with TRAINID are as follows:

Mnemonic Value	Integer Value	Meaning
ASCII72	15	A printer with an ASCII72 train
BCL64	12	A printer with a BCL64 train
EBCDIC18	1	A printer with an EBCDIC18 train
EBCDIC48	4	A printer with an EBCDIC48 train
EBCDIC72	5	A printer with an EBCDIC72 train
EBCDIC96	16	A printer with an EBCDIC96 train
NOID	0	Any available printer

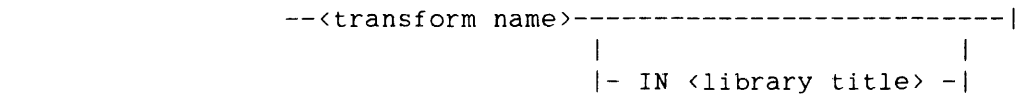
On B 6800, B 7700, and B 7800 machines, files that are printed on a train printer when TRAINID has the value NOID are printed just as they would be printed on a drum printer. The character set used is the BCL64 set or a similar set of 64 characters. Characters that would print as question marks on the drum printer are also 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 in order to print more than the default 64 characters.

TRANSFORM

Kinds:	printer/punch
Interrogate:	anytime
Modify:	anytime
Type:	pointer
Default:	null string
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	not supported

The TRANSFORM attribute specifies a list of transforms to be executed between the retrieval of records from source files and the printing of those records on destination devices. Transforms allow users to tailor the Print System to their specific needs. For more information, see "Writing Transform Functions" in the Print System (PrintS/ReprintS) User's Guide.

When this attribute is set, the string value must conform to the following syntax:



If a library title is specified, the transform name is assumed to be an entry point within the specified library. If no library title is specified, the name is assumed to refer to a standard transform.

When multiple destinations are specified with the DESTINATION attribute, the same transform is used for each destination.

The default value is a null string.

If PRINTDISPOSITION = DIRECT, the TRANSFORM attribute does not apply and is ignored.

TRANSLATE

Kinds:	all except BNA V1 port files
Interrogate:	anytime
Modify:	when closed
Type:	mnemonic
Range:	DEFAULTTRANS, FORCESOFT, FULLTRANS, NOSOFT, NOTRANS, SOFTONLY, USERTRANS
Default:	(see below)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	restricted values

The TRANSLATE attribute indicates the scope of translation. The mnemonic values and meanings associated with the TRANSLATE attribute are as follows:

Mnemonic Value	Integer Value	Meaning
DEFAULTTRANS	0	Software translation may occur when translation is required and hardware translation is not provided (see below). The DEFAULTTRANS value is not supported by the BNA Logical I/O Host Service. Any request for DEFAULTTRANS is changed to FULLTRANS.
FORCESOFT	3	Software translation takes 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 (see below). While performing binary I/O, no translation is permitted. A program specifying TRANSLATE = FORCESOFT while doing binary I/O is discontinued with the message, "TRANSLATE=FORCESOFT NOT ALLOWED WITH BINARY IO". If translation would not otherwise be required, the program must supply the TRANSLATETABLES or the program will receive a fatal field error on the first I/O. Refer also to the descriptions of the INPUTTABLE and OUTPUTTABLE attributes. In BNA V2 port files, any request for FORCESOFT causes an OPEN error.

Mnemonic Value	Integer Value	Meaning
FULLTRANS	1	Software translation occurs when translation is required and hardware translation is not provided. On the B 6800, B 7700, and B 7800 Series systems, FULLTRANS is the default value of TRANSLATE when the run-time system option FULLTRANSLATION is set.
NOSOFT	4	An invalid translation open error occurs whenever translation is required and hardware translation is not provided.
NOTRANS	5	An invalid translation open error occurs whenever hardware or software translation is required.
SOFTONLY	2	Only software translation takes place whenever translation is required. (No hardware translation takes place.)
USERTRANS	6	If translation is required, it will be done by the user. Translation is required when INTMODE is not equal to EXTMODE and TRANSLATING is FALSE (Supported for BNA V2 port files only).

On the B 6800, B 7700, and B 7800 Series systems, DEFAULTTRANS is the default value of TRANSLATE when the run-time system option FULLTRANSLATION is reset, and FULLTRANS is the default value of TRANSLATE when FULLTRANSLATION is set. (Refer to the Operator Display Terminal (ODT) Reference Manual, under the command "OP (OPtions)", for specific information regarding the FULLTRANSLATION run-time system option.)

Translation is not supported for BNA V1 port files. For BNA V2 port files, the default value of TRANSLATE is always NOTRANS.

On DLP-based systems, FULLTRANS is the default value of TRANSLATE regardless of the value of FULLTRANSLATION. When using the BNA Logical I/O Host Service, the default value is FULLTRANS.

The INTMODE attribute describes the mode (or character encoding) of the logical file, and the EXTMODE attribute describes the mode of the physical file. Translation occurs between these modes, provided that neither INTMODE nor EXTMODE has a value of SINGLE (word mode or binary).

For further information regarding translation, refer to "Hardware and Software Translation".

See also
Hardware and Software Translation 377

TRANSLATING

Kinds:	all devices
Interrogate:	when assigned
Modify:	never
Type:	Boolean
Range:	TRUE, FALSE
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	supported

The TRANSLATING attribute returns TRUE if software translation is being performed on the records of the file. The value of the TRANSLATING is determined when the file assignment is complete, taking into consideration the values of the BLOCKSTRUCTURE (or FILETYPE), EXTMODE, INTMODE, and TRANSLATE attributes. (Refer to the descriptions of these attributes for related information.)

For port files, the value of the TRANSLATING attribute is determined during the matching process. Refer to "Opening Port Files" for details.

In BNA V2 port files, the TRANSLATING attribute has a subfile index as a required parameter, if MAXSUBFILES is greater than 1.

See also

BLOCKSTRUCTURE. 135
EXTMODE 178
FILETYPE. 198
INTMODE 213
MAXSUBFILES 235
Opening Port Files. 76
TRANSLATE 319

TRANSMISSIONNO

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

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

See also

Relative Station Number 91

TRANSMISSIONO

TRANSMISSIONO is a nonpreferred synonym for TRANSMISSIONNO. Refer to the description of TRANSMISSIONNO.

TRANSMISSIONS

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The TRANSMISSIONS attribute returns the number of output messages sent to the station if a Relative Station Number (RSN) is specified as a parameter. If the parameter is omitted, TRANSMISSIONS returns the total number of output messages sent to all the stations in the file.

See also

Relative Station Number 91

TRIMBLANKS

Kinds:	printer/punch backup,remote
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Default:	TRUE for printer/punch backup, FALSE for remote
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The TRIMBLANKS attribute specifies whether trailing EBCDIC blank characters (spaces) will be stored in printer/punch backup files or transmitted in remote file WRITE statements. The attribute only applies to non-direct files with an INTMODE and EXTMODE of EBCDIC.

For backup files, the user print/punch line is stripped of trailing blank characters to the word boundary of the first word that is not all blank. The last partial EBCDIC word containing less than six blank characters will not be trimmed.

For remote files with UNITS = CHARACTERS, all trailing blank characters will be trimmed. If UNITS = WORDS, all trailing blank words will be trimmed. I/O statements of the ALGOL form "WRITE(F[STOP], ..." are not affected.

Interrogating TRIMBLANKS will return TRUE for an open file only if blank trimming is being done, for example, in remote and printer/punch backup files.

This attribute can make printer/punch backup files much smaller and can considerably reduce the I/O time of programs that produce them. Also, the data communication line time of programs that write to remote files can be reduced, especially over slow data communication lines or when the program writes one line at a time. However, use of TRIMBLANKS will increase the processor time required to do each WRITE operation.

See also

EXTMODE 178
INTMODE 213
UNITS 328

UNITNO

Kinds:	all devices (except port and remote)
Interrogate:	anytime
Modify:	when unassigned
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

For a disk file that is open, the UNITNO attribute returns the memory index of the disk file header. In all other cases, the UNITNO attribute returns the hardware unit number to which the file is assigned or is to be assigned.

The UNITNO attribute can be modified to a unit number to which the file is to be assigned. The attribute can be modified only before the logical file is assigned to a physical file. The attribute cannot be modified by file equation. If the UNITNO attribute has been set to a value that is associated with a physical disk unit or to which no physical unit is assigned, if the KIND attribute of the file is not the same type as the physical device, or if the logical and physical files are in any way incompatible, then the program is terminated. (Refer to the description of the KIND attribute.)

UNITS

Kinds:	all devices
Interrogate:	when unassigned (BNA Logical I/O) anytime (local host)
Modify:	when closed
Type:	mnemonic
Range:	CHARACTERS, WORDS
Default:	WORDS
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	restricted usage

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

The mnemonic values and meanings associated with the UNITS attribute are as follows:

Mnemonic Value	Integer Value	Meaning
CHARACTERS	1	INTMODE-defined units
WORDS	0	48-bit units

If UNITS is CHARACTERS, then, unless INTMODE is SINGLE, the file is character-oriented.

The FRAMESIZE attribute overrides and may change the value of UNITS. For this reason, the use of the FRAMESIZE attribute is preferred over the use of the UNITS attribute.

The BNA Logical I/O Host Service does not allow UNITS to be interrogated when the file is assigned. FRAMESIZE can be used instead.

UPDATEFILE

Kinds:	disk
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	restricted values

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

When a disk file is open and UPDATEFILE is TRUE, a serial (non-random) 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.

When a disk file is opened, UPDATEFILE is used to indicate whether or not update I/O action is to be performed on the file. Regardless of the setting of the attribute, if the logical file is assigned to a non-disk peripheral unit, then the value of the attribute is FALSE.

In order to retain the efficiency gained by means of update I/O, the default number of buffers for a file with UPDATEFILE TRUE is three. Having less than three buffers results in the MCP Logical I/O routines waiting for look-ahead reads to complete. (For related information, refer to the description of the BUFFERS attribute.)

If interrogated when the file is closed, UPDATEFILE returns the last value assigned to it.

The BNA Logical I/O Host Service supports only the value FALSE.

USECATALOG

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	(see below)
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	supported

When the USECATALOG attribute is TRUE, permanent disk or tape 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. (For more information, refer to the Operator Display Terminal (ODT) Reference Manual under the command "OP (OPtions)".)

USEDATE

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 99999
Default:	0
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	supported

The USEDATE attribute returns the date when the file was last read or written to by a user program or, in the case of a code file, the date when it was last executed. The value returned by USEDATE is an integer in the form YYDDD, where YY and DDD represent the year and day, respectively, in Julian form. The USEDATE of a file is unaffected by library maintenance. USEDATE is not updated for files on write lock-out protected disk.

For code files, USEDATE is changed upon execution (at BOJ/BOT). For a permanent disk file, USEDATE is changed when the file is closed if a read, a write, a seek, or a space has been performed on the file. For control files, guard files, the USERDATAFILE, and other system files, USEDATE is updated when the file is closed. For the various disk files referenced by the MCP, USEDATE will usually be updated when the reference is made. For example, libraries that have been labeled using the SL (System Library) ODT command will be marked when they are initiated, as will INTRINSIC files, data communication files, and disk firmware files.

See also
USETIME 334

USERBACKUPNAME

Kinds:	printer/punch
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

The USERBACKUPNAME file attribute overrides the system naming conventions for backup files. When the USERBACKUPNAME file attribute is TRUE, the system will use TITLE as the title of the backup file and apply family substitution as it does for other files. When USERBACKUPNAME is FALSE, the system will use its own naming conventions for backup files.

If USERBACKUPNAME is TRUE, the file title by default includes the usercode of the task creating the file. Backup file that include a usercode in their title by default have SECURITYTYPE equal to PRIVATE.

Regardless of the setting of USERBACKUPNAME, the TITLE file attribute will return the title of the backup file when a PRINTER file is open. This will be true whether the title was constructed by the system or supplied by the user.

Note that if a family name is desired in the title, it should be specified using the FAMILYNAME file attribute. This is because the MCP changes the KIND of backup file from PRINTERBACKUPDISK to DISK when it finds the "ON <family name>" clause in the TITLE attribute.

Example:

```
USERBACKUPNAME = TRUE
```

See also

TITLE 314
SECURITYTYPE. 285

USERINFO

Kinds:	disk
Interrogate:	when assigned
Modify:	when assigned
Type:	real
Default:	0
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The USERINFO attribute is available for general use. Its meaning is determined by the user.

The file must be assigned in order to access this attribute. The value for USERINFO can be modified for any disk file and interrogated later. The timestamp of the disk file header is updated when the attribute is modified.

The USERINFO attribute can be set only by a user with write-access to this file, but can be read by any user. If read when the file is assigned, USERINFO returns the value for the physical file.

USERINFO is not implemented for Installation-Allocated Disk (IAD) files. The values it returns are unaffected by library maintenance.

USETIME

Kinds:	disk
Interrogate:	when assigned
Modify:	never
Type:	integer
Range:	0 through 86400000000
Default:	0
Stored Permanently:	disk
Parameters:	none
BNA Logical I/O:	not supported

The USETIME attribute returns the time of day, in microseconds since midnight, associated with the USEDATE attribute. The value returned by USETIME is unaffected by library maintenance.

For code files, USETIME is changed upon execution (at BOJ/BOT). For a permanent disk file, USETIME is changed when the file is closed if a read, a write, a seek, or a space has been performed on the file. For control files, guard files, the USERDATAFILE, and other system files, USETIME is updated when the file is closed. For the various disk files referenced by the MCP, USETIME will usually be updated when the reference is made. For example, libraries that have been labeled using the SL (System Library) ODT command will be marked when they are initiated, as will INTRINSIC files, data communication files, and disk firmware files.

See also
USEDATE 331

VERSION

Kinds:	disk/tape/diskette
Interrogate:	anytime
Modify:	when not in directory (disk) when closed (tape)
Type:	integer
Range:	0 through 99
Default:	0
Stored Permanently:	disk/tape
Parameters:	none
BNA Logical I/O:	supported

The VERSION attribute, in conjunction with the CYCLE attribute, is used to distinguish successive iterations of the same generation of a permanent file.

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

Refer also to the description of CYCLE.

WIDTH

Kinds:	remote
Interrogate:	when open
Modify:	never
Type:	integer
Default:	not applicable
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	supported

The WIDTH attribute 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;
```

See also

Relative Station Number 91

WRITECHECK

Kinds:	diskette
Interrogate:	anytime
Modify:	when closed
Type:	Boolean
Range:	TRUE, FALSE
Default:	FALSE
Stored Permanently:	no
Parameters:	none
BNA Logical I/O:	not supported

If the value of the WRITECHECK attribute is TRUE, the Diskette Controller checks for data errors after a write through a CRC (Cyclic Redundancy Check) code.

YOURHOST

Kinds:	all devices
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	null string
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	supported

The YOURHOST attribute specifies a <simple identifier> of 1 to 17 characters long that is used during file assignment to specify the host on which the physical file or complementary port file exists or is to be created.

If YOURHOSTGROUP is set before YOURHOST, the value of YOURHOST is implicitly set to NULL.

If the file is a port file and more than one subfile exists, YOURHOST requires a subfile index as a parameter.

In the case of a port-subfile, YOURHOST specifies the host in the network that contains the complementary process with which this process attempts to communicate using this subfile. YOURHOST is used during the subfile matching process. The value of YOURHOST for each subfile must match the value of MYHOST for the complementary subfile. (Refer to the description of the MYHOST attribute.)

If the file is closed and unassigned and YOURHOST has never been set, then YOURHOST returns the name of the local host. If the file is assigned, then YOURHOST returns the name of the host on which the file resides.

The BNA Logical I/O Host Service does not allow YOURHOST to be altered using the FA (File Attribute) ODT command once the file is open. For a description of the FA command, refer to the Operator Display Terminal (ODT) Reference Manual.

For non-port files, YOURHOST is a nonpreferred synonym for HOSTNAME.

YOURHOSTGROUP

Kinds:	port-subfile
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	(null string)
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The YOURHOSTGROUP attribute is available only in BNA V2 port files.

The YOURHOSTGROUP attribute indicates the name of a group of hosts, one of which contains the remote process with which the local process asks to communicate. The value of YOURHOSTGROUP for each subfile, if non-null, must match the value of MYHOSTGROUP for the complementary subfile. If YOURHOSTGROUP is set before YOURHOST, the value of YOURHOST is implicitly set to NULL.

If MAXSUBFILES is greater than 1, the YOURHOSTGROUP attribute requires a subport index as a parameter.

YOURNAME

Kinds:	port-subfile
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	(null string)
Stored Permanently:	no
Parameters:	1 optional
BNA Logical I/O:	not supported

The YOURNAME attribute is a string of up to 100 characters 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. A null value of YOURNAME will match any complementary MYNAME value. (Refer to the description of MYNAME.)

See also

Opening Port Files. 76

YOURUSERCODE

Kinds:	port-subfile
Interrogate:	anytime
Modify:	when closed
Type:	pointer
Default:	MYSELF.USERCODE
Stored Permanently:	no
Parameters:	1 required
BNA Logical I/O:	not supported

The YOURUSERCODE attribute is a <simple identifier> used by the subfile-matching algorithm to match complementary subfiles.

If the value of the port file's SECURITYTYPE attribute is PRIVATE, then the subfile's YOURUSERCODE attribute must equal the usercode under which the program offering the complementary subfile is running. Refer to "Opening Port Files" and the description of SECURITYTYPE.

The default value of YOURUSERCODE is the usercode of the task that opens the file. If OLDYOURUSERCODE is TRUE (the default value), then assigning the null string to YOURUSERCODE causes its value to revert to the system-supplied default value.

However, if OLDYOURUSERCODE is FALSE, assigning the null string to YOURUSERCODE causes its value to be the null string. Also, when the subfile is opened, YOURUSERCODE will contain the usercode of the task declaring the complementary subfile. YOURUSERCODE will revert to its previous value when the subfile is closed.

See also
Opening Port Files. 76
OLDYOURUSERCODE 249

5 DIRECT I/O

5.1 INTRODUCTION

Direct I/O is an I/O technique that allows a 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 to code files, except by programs with FILEKIND specified as COMPILERCODEFILE. (Refer to the MC (Make Compiler) ODT command described in the Operator Display Terminal (ODT) Reference Manual.) Direct I/O writes also are not allowed to crunched files if the BLOCKSIZE does not equal the original BLOCKSIZE, nor are they allowed to a copy of a duplicated file.

The length parameter in Direct I/O read and write statements contains two fields: the word count ([16:17]) and the character count ([19:3]). The character count is actually a physical frame count; the frame size is usually eight bits but may be six bits for some devices on some systems. (Physical frame size is set by default by the MCP. In some cases, the user can modify the physical frame size through the IOFRAMESIZE bit ([41:1]) in the direct buffer attribute I/O Control Word [IOCW].)

Some peripheral controls require frames to be transmitted in pairs; attempts to transmit an odd number of frames can cause a descriptor error.

The frame count can range from zero through five for 8-bit frames, and from zero through seven for 6-bit frames; 0 is normally used in word-mode files. The I/O length is always specified in number of 48-bit words and number of additional physical frames, regardless of the values of the EXTMODE, FRAMESIZE (or UNITS), and INTMODE attributes.

When performing Direct I/O along with the SPACE operation, the device's spacing limitation overrides any user-specified arithmetic expression part of the SPACE operation. In the case of a line printer, the maximum spacing is 2.

Buffer attributes are provided to aid in controlling the Direct I/O and to describe its results. Buffer attributes are accessible in ALGOL and COBOL. The following is an ALGOL example:

```
ERR := DIRECTARRAYID.IOERRORTYPE;
```

Note that in all examples in this section, the direct array (used as the buffer) is referenced, not the file identifier. For Direct I/O operations, as for I/O operations in general, the file variable is protected by the system. However, because the user manages the direct arrays used as buffers in Direct I/O operations, he should avoid attempting multiple, simultaneous use of the direct array in his program.

See also

BLOCKSIZE 133
EXTMODE 178
FILEKIND. 185
FRAMESIZE 206
INTMODE 213
UNITS 328

5.2 BUFFER ATTRIBUTE CHARACTERISTICS

Each buffer attribute described in this section lists all or part of the following items directly beneath the attribute heading:

Interrogate:
Modify:
Type:
Range:
Default:

INTERROGATE

"Interrogate" indicates whether or not an attribute can be interrogated (that is, read) and, if so, under what restrictions. The possible values and associated meanings for this item are as follows:

Value	Meaning
never	The attribute can never be interrogated. In other words, it is a write-only attribute.
when closed	The file must be closed in order for the attribute to be interrogated.
when open	The file must be open in order for the attribute to be interrogated.
when assigned	The logical file must be assigned to a physical file in order for the attribute to be interrogated.
when unassigned	The logical file must not be assigned to any physical file in order for the attribute to be interrogated.
anytime	The attribute may be interrogated under any circumstances.

MODIFY

"Modify" indicates whether or not an attribute can be modified and, if so, under what restrictions. The possible values and associated meanings for this item are as follows:

Value	Meaning
never	The attribute can never be modified. In other words, it is a read-only attribute.
when closed	The file must be closed in order for the attribute to be modified.
when open	The file must be open in order for the attribute to be modified.
when assigned	The logical file must be assigned to a physical file in order for the attribute to be modified.
when unassigned	The logical file must not be assigned to any physical file in order for the attribute to be modified.
anytime	The attribute may be modified under any circumstances.

TYPE

"Type" indicates the data type required when using an attribute. The possible values and associated meanings for this item are as follows:

Value	Meaning
Boolean	Boolean-valued
integer	integer-valued
word	48 bits of information as described in the explanation of the attribute.

RANGE

"Range" indicates a list of possible values associated with the attribute. These values vary according to the "type" of the given attribute.

DEFAULT

"Default" indicates the value that the system uses when the user does not specify a value for the attribute. Depending on the attribute, the possible entry for this item is usually either a value that is a member of the defined range for that attribute or "not applicable". If no default exists for the attribute, the entry for this item is "no default".

5.3 INDIVIDUAL BUFFER ATTRIBUTE DESCRIPTIONS

Each buffer attribute description begins with a formatted list of attribute characteristics of the kind just described followed by a narrative discussion of the attribute's uses. Where appropriate, this discussion includes a list of the various combinations of values, mnemonics, and meanings associated with that attribute.

The individual buffer attribute descriptions are presented in alphabetical order.

IOADDRESS

CAUTION

The IOADDRESS attribute has been deimplemented for the Mark 3.6 Release. Use of IOADDRESS will cause a compilation error message and a fatal run-time error message.

IOCANCEL

Interrogate: anytime
Modify: anytime
Type: Boolean
Range: TRUE, FALSE
Default: not applicable

The IOCANCEL attribute returns TRUE if the last I/O attempted on this buffer was canceled. If this is the case, the IORESULT attribute returns 1. Specifying IOCANCEL as TRUE on a buffer where IOPENDING is TRUE purges the I/O queue and cancels all the outstanding I/Os for the device that were initiated using direct arrays declared in the same stack as the direct array for which IOCANCEL is being specified to TRUE.

IOCANCEL has no effect for disk and remote files.

IOCHARACTERS

Interrogate: anytime
Modify: never
Type: integer
Default: not applicable

The IOCHARACTERS attribute returns the number of characters read into this buffer. The value returned after reading in reverse is unpredictable on B 5000/B 6000 Series systems; the value returned after a write is unpredictable on all systems.

Following a forward read operation on a tape file, the value returned indicates the actual size of the tape block.

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

Interrogate: anytime
Modify: never
Type: Boolean
Range: TRUE, FALSE
Default: not applicable

The IOCOMPLETE attribute returns TRUE if the I/O action for this buffer is complete.

IOCW

Interrogate: anytime
Modify: anytime
Type: word
Range: valid IOCWs
Default: device-dependent

The IOCW attribute accepts or returns the I/O control word for the buffer. If this attribute is modified, it is used instead of the system-supplied IOCW. The IOCW reverts to the system-supplied default when the attribute is modified to the value 0.

Modifying the DIRECTION attribute may affect the value of IOCW.

The IOCW attribute is ignored for HC files.

See also
DIRECTION 168

IOEOF

Interrogate: anytime
Modify: never
Type: Boolean
Range: TRUE, FALSE
Default: not applicable

The IOEOF attribute returns TRUE if an end-of-file condition was encountered as a result of the last I/O on this buffer.

IOERRORTYPE

Interrogate: anytime
Modify: never
Type: integer
Range: (see below)
Default: not applicable

The IOERRORTYPE attribute identifies what error, if any, occurred as a result of the last I/O on this buffer. When the event signifying the IOCOMPLETE has not been caused (IOPENDING is TRUE), the value of IOERRORTYPE is undefined. 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. For an HC I/O, this result indicates a coordinated error; that is, an exception result has been returned to both the read side and the write side of an attempted ISC transfer.
3	Rewinding
4	Descriptor error
5	For tape I/O, end of tape or beginning of tape. For disk I/O, NORESOURCEWAIT is TRUE and a "NO SPACE" condition has been encountered.
6	End-of-file or write lock-out. For an HC write operation, this result indicates that write access is denied, which means the destination HC (specified WRITEPARTNER) is not marked as input-capable.
7	Attempt to exceed disk area or read past EOF
8	I/O was canceled

Value	Meaning
9	Short record. For an HC read operation, this result indicates a long block, which means that the writing HC sent more data than the reading direct array could hold. This error is not a coordinated error; that is, the corresponding HC write operation does not get an exception result.
10	Break on output (remote files only)
11	Disk pack program or operator error
12	Disk pack hardware error
13	Disk pack hardware or data error
14	Disk pack data error
15	Disk storage unit timeout or blank tape timeout
16	Area write locked out by DMS (program has read a writelocked area)
18	HC file write timeout. An HC write specified an input-capable WRITEPARTNER, but there was no corresponding read operation at the destination HC within the allowed 30 seconds.
19	Bad I/O partner specification. An HC file user attempted to override a HUBMAP level READPARTNER or WRITEPARTNER specification by modifying the corresponding direct array attribute.
20	No write partner specification. An HC write was attempted for which there was no HUBMAP level WRITEPARTNER specification and for which the direct buffer attribute WRITEPARTNER was not specified.
21	Input from wrong hubindex. An HC file read with the READPARTNER specified received input from a different hubindex.

Indications of an end-of-tape condition (which occurs when the tape is positioned beyond the end-of-tape marker) is system-specific but is always consistently reported by the IORESULT and IOERRORTYPE attributes. On B 6800 Series systems, end-of-tape conditions are never reported. On B 7700, B 7800, and B 7900 Series systems, end-of-tape conditions are reported when they are encountered on tape read operations only if the appropriate bit is set in the IOMASK attribute; otherwise, they are never reported. On B 5900 and B 6900 Series systems, end-of-tape conditions are reported when they are encountered on tape read operations. On all systems, if an end-of-file condition occurs in conjunction with an end-of-tape condition, IOERRORTYPE reports only the end-of-file condition.

See also

NORESOURCEWAIT. 246
WRITEPARTNER. 365

IOMASK

Interrogate: anytime
Modify: anytime
Type: word
Default: 0

The IOMASK attribute provides the ability to modify or interrogate 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 that the user will handle these errors programmatically. Once the IOMASK attribute is modified, it is used instead of the system-supplied default. The IOMASK reverts to the system-supplied default when the attribute is modified to the value 0.

On all systems, a value of 1 in bit 9 of IOMASK causes the normal system end-of-file action to be suppressed. This action is of particular utility for tape files because suppression of the system's end-of-file action circumvents automatic volume switching.

On B 5900 and B 6900 Series systems, the IOMASK attribute is ignored for DISK and PACK devices.

On B 7700, B 7800, and B 7900 Series systems, the IOMASK attribute can cause certain conditions to be reported (via the IORESULT attribute) that would otherwise be suppressed. For example, a value of 1 in bit 8 of IOMASK causes end-of-tape conditions to be reported when they are encountered on tape read operations.

The IOMASK attribute is ignored for HC files.

IOPENDING

Interrogate: anytime
Modify: never
Type: Boolean
Range: TRUE, FALSE
Default: not applicable

The IOPENDING attribute returns TRUE if an I/O is queued or in process on this buffer.

IORECORDNUM

Interrogate: anytime
Modify: never
Type: integer
Default: not applicable

The IORECORDNUM attribute returns the logical record number of the last disk I/O performed on this buffer or the value of the LASTSUBFILE attribute for Direct remote files.

See also
LASTSUBFILE 226

IORESULT

Interrogate: anytime
Modify: never
Type: word
Default: not applicable

On B 6800, B 7700, and B 7800 Series systems, the IORESULT attribute returns the hardware result descriptor for the last I/O on this buffer. IORESULT returns 1 if the I/O was canceled.

On all other systems, the logical result descriptor is returned (that is, the same one returned on a WAIT statement on the buffer).

Indications of an end-of-tape condition (which occurs when the tape is positioned beyond the end-of-tape marker) is system-specific but is always consistently reported by the IORESULT and IOERRORTYPE attributes. On B 6800 Series systems, end-of-tape conditions are never reported. On B 7700, B 7800, and B 7900 Series systems, end-of-tape conditions are reported when they are encountered on tape read operations only if the appropriate bit is set in the IOMASK attribute; otherwise, they are never reported. On B 5900 and B 6900 Series systems, end-of-tape conditions are reported when they are encountered on tape read operations. On all systems, if an end-of-file condition occurs in conjunction with an end-of-tape condition, IOERRORTYPE reports only the end-of-file condition.

For HC file reads that complete without errors or have IOERRORTYPE results of "PARITY" (2), "LONG BLOCK" (9), or "INPUT FROM WRONG HUBINDEX" (21) (that is, reads in which data transfer occurs), the field [23:4] contains the hubindex from which the data was written.

Note that on all systems except B 6800, B 7700, and B 7800 Series systems, beginning with the Mark 3.6 release, the field [24:8] of the IORESULT attribute will be 0, except for HC file reads as noted in the preceding paragraph.

IOTIME

Interrogate: anytime
Modify: never
Type: integer
Default: not applicable

The IOTIME attribute returns the time, in 2.4-microsecond units, for the last I/O on this buffer. The I/O time is a measured value that is used as the basis for system I/O accounting. Its precise definition is system-specific and peripheral-dependent.

IOWORDS

Interrogate: anytime
Modify: never
Type: integer
Default: not applicable

The IOWORDS attribute 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 (where m is greater than 0), IOWORDS returns n + 1. The value returned after reading in reverse is unpredictable on B 5000/B 6000 Series systems; the value returned after a write is unpredictable on all systems.

Following a forward read operation on a tape file, the value returned indicates the actual size of the tape block.

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.

READPARTNER

Interrogate: anytime
Modify: anytime
Type: integer
Default: -1

The READPARTNER attribute specifies the hubindex from which input is expected for HC read operations. This attribute is meaningful only when the KIND of the file with which the I/O on a direct array is associated is HC.

Because 0 is a valid READPARTNER specification, a value of -1 is used to indicate an unspecified I/O partner.

A READPARTNER should be specified at the direct array level only for operations on HCs that do not have installation-imposed I/O partner restrictions in the HUBMAP. If a program specifies READPARTNER and attempts an I/O operation to an HC that has a READPARTNER specified at the HUBMAP level, a "BAD IOPARTNER SPECIFICATION" (IOERRORTYPE 19) error results.

See also
Host Control Files. 56
KIND. 218

WRITEPARTNER

Interrogate: anytime
Modify: anytime
Type: integer
Default: -1

The WRITEPARTNER attribute specifies the target hubindex to which HC write operations are directed. This attribute is meaningful only when the KIND of the file with which the I/O on the direct array is associated is HC.

Because 0 is a valid WRITEPARTNER specification, a value of -1 is used to indicate an unspecified I/O partner.

A WRITEPARTNER must be specified at the direct array level only for operations on HCs that do not have an installation-specified WRITEPARTNER in the HUBMAP. If a program specifies WRITEPARTNER and attempts an I/O operation to an HC that has a WRITEPARTNER specified at the HUBMAP level, a "BAD IOPARTNER SPECIFICATION" (IOERRORTYPE 19) error results. If WRITEPARTNER for an HC is not specified at the HUBMAP level, an attempt to write with a direct array that does not have WRITEPARTNER specified results in a "NO WRITE PARTNER SPECIFICATION" (IOERRORTYPE 20) error.

See also
Host Control Files. 56
KIND. 218

5.4 DIRECT I/O ON DISK FILES

The use of Direct I/O on disk files permits considerable flexibility but also involves some fine distinctions. Direct I/O on most devices links the programmer very closely to the input/output device. However, this connection is less direct for disk files, which exist on devices that may be shared by many users. The normal disk file management system is active in allocating regions of disk to temporary or permanent files. Direct I/O is a means of accessing file data and can be used independently of how the file was created.

PHYSICAL FRAME SIZE, ODD FRAMES

For disk files, the physical frame size is always 8 bits; any attempt to change the frame size is ignored. Even in HEX or BCL files, where the unit size is 4 bits and 6 bits, respectively, the Direct I/O length is specified by the number of 48-bit words plus the number of 8-bit frames. (Refer to the description of the length parameter in the introduction to this section.) For disk files, an odd number of frames may be requested, and the end-of-file reckoning is done with the number specified. The I/O Subsystem allows I/O-length values other than in 30-word multiples, but the hardware will always write that many (using zero-filling) and reads must always begin at a sector boundary.

Only an even number of 4-bit units can be specified. One extra unit must be written in any block containing an odd number of HEX records.

See also
MAXRECSIZE. 232

AREAS, BLOCKS, RECORDS, SEGMENTS

The BLOCKSIZE, FRAMESIZE (or INTMODE and UNITS), and MAXRECSIZE attributes define the logical block and record size for the file. Because these attributes define the way a file is handled with Logical (non-Direct) I/O, their application to Direct files requires some explanation. Direct files deal primarily with disk sectors and secondarily with blocks; record numbers are used mainly for addressing blocks.

The unit of disk storage addressability is the sector, which holds 30 words (180 8-bit bytes). Each I/O operation begins at a sector boundary and transmits one or more contiguous sectors. On a write, if the data runs out before the end of a sector, the disk subsystem pads the last sector with nulls. Every block begins on a sector boundary and occupies one or more contiguous sectors. If the block size is not a multiple of 180 bytes, some wasted space remains at the end of each block.

If the record size is fixed (BLOCKSTRUCTURE is FIXED or FILETYPE is 0) and it is smaller than the block size, then each block contains a fixed number of records and records per block must be a fixed integer equal to the ratio of BLOCKSIZE to MAXRECSIZE. If the record size is variable, then, for Direct Disk I/O, the term "record" is considered to be a synonym for "block", so that records per block always equals 1.

The file may contain several areas, each of which contains some number of whole blocks. The area size is determined by the value of the AREALENGTH attribute or AREASIZE attribute.

In ALGOL READ and WRITE statements, a "[<record number or carriage control>]" part, containing some text in brackets, may appear immediately after the file name. If this part is present, "random" I/O occurs; otherwise, the I/O is "serial". In COBOL, the "KEY IS" clause invokes Random I/O. Serial I/O begins at the sector of the file just past the last sector read or written, regardless of any record boundaries. Random I/O always begins at the beginning of a block: the specified record number R is adjusted to (R DIV B) * B, where B is the number of records per block.

The "[<record number or carriage control>]" specifying Random I/O in ALGOL takes several forms:

"[<arithmetic expression>]" is the usual syntax, where the value of <arithmetic expression> denotes the record number.

"[SPACE <arithmetic expression>]" specifies the current record position plus the value of <arithmetic expression> as the new record number.

"[NO]" specifies the current record number as the new record number, but does not update the record or sector position, so that a subsequent Serial I/O occurs at the same place.

Any other form of the "[<record number or carriage control>]" is ignored, and the current record position is used.

Direct I/O permits transmission to begin at any sector in the file and to continue for any length, bounded only by the buffer size and the end of the current area (or the end of the file). Direct I/O allows reading or writing anywhere in the file, including in the gaps at the ends of logical blocks.

After a direct read or write, the IORECORDNUM buffer attribute reports the record number where the transmission began, and the file attribute RECORD indicates the record number from which a subsequent serial transmission is to proceed. Neither of these points is necessarily at the beginning of a record if a serial read or write is done with lengths different from whole blocks.

Usually, writing to, or beyond, the end of the file simply extends the file and updates the end-of-file pointer; no error is reported. The exceptions to this rule include crunched files, files with the maximum number of areas, and files with FLEXIBLE equal to FALSE and the number of allowed areas (as specified by the AREAS attribute) already allocated. A crunched file cannot be extended past the end-of-file sector, and the other two types of files cannot be extended beyond the last area. Attempts to write outside a nonextendible file are treated just like other attempts to read or write outside the file.

If I/O is attempted completely outside the file, end-of-file action is taken: no I/O takes place, bits 9 and 0 are set in the logical result descriptor returned by the WAIT function or by the buffer attribute IORESULT, and the buffer attribute IOERRORTYPE reports 6. If I/O begins within the file but extends across the area boundary or across end-of-file, special action is taken: data transfer occurs with the length truncated, the logical result descriptor has bits 10 and 0 set, and IOERRORTYPE reports 7. Direct I/O can read or write the entire sector to which the end-of-file points; for writing, the end-of-file pointer is adjusted to the end of the write.

See also

END-OF-FILE POINTERS

The end-of-file pointer in a disk file specifies the last bit that has been written in the file. For direct disk files, the end pointer is set according to the starting position and length of any write beyond the previous end position. Because Direct I/O length is specified in 8-bit physical frames, the end-of-file pointer cannot always be placed at a record boundary.

A similar situation can arise with 6-bit units, where the logical and physical frame boundaries align only at whole-word and half-word boundaries.

When the end-of-file pointer is used by the Logical I/O subsystem to determine the number of records in the file, the following are discarded: any partial logical frames, any partial second or subsequent record in a block, and any data in the last sector of a block past the end of the logical block.

See also
MAXRECSIZE. 232

ZERO-LENGTH I/O

Because zero-length, direct read/write statements transfer no data, zero-length, serial operations have no effect on the record pointer in the file. However, random operations reassign the record pointer, thus affecting subsequent nonzero-length, serial operations.

Random reads or writes generate end-of-file action if the specified record number is less than 0. No other end-of-file checking is done for zero-length reads, but zero-length writes generate end-of-file action if the record number is past the end of a crunched or otherwise nonextendible file.

If the addressed record is in a new area, a zero-length write causes the disk space to be allocated.

In summary, a zero-length random read operation functions as a seek, whereas a zero-length random write operation functions as a seek but also may allocate disk space.

5.5 OPTIMIZATION OF DIRECT I/O

In certain cases, performance is improved for Direct I/O initiation (READ or WRITE) and WAIT for completion when an event or event array element is provided with the initiation statement and then used in the WAIT statement.

There are several forms of I/O initiation statements that can be used with direct files. Assume that DF is a direct file, DA is a direct array, and E is an event or event array element. The I/O initiation statements are as follows:

1. READ(DF, length, DA);
   WRITE(DF, length, DA);

2. READ(DF, length, DA) [E];
   WRITE(DF, length, DA) [E];

In fact, case 2 above can be subdivided into two subcases. In the first subcase (say case 2A), the compiler can ensure that E will never occur at a lexical level higher than that of DA. In the second subcase (say case 2B), the compiler cannot ensure this relationship.

In case 2B, the lexical level check is performed at run time by the MCP Logical I/O Read or Write procedure. Less processor time is used when the event or event array element provided for use at initiation time is the same as the one used for the previous I/O on the direct array. When more than a few I/O operations are done, the processor time used is same as that in case 2A.

WAITs for completion of an I/O operation can also be divided into two cases:

1. WAIT(DA); % Used with a case 1 I/O initiation statement

2. WAIT(E): % Where E was supplied as a completion event in
   % an I/O initiation statement (case 2A or 2B)

In general, a case 2 WAIT uses less processor time than a case 1 WAIT.

6 DUPLICATED FILES

The purpose of duplicated files is to avoid the loss of information if a fatal error on disk occurs. An error is considered "fatal" (that is, irrecoverable) when a program does a READ from a disk file and the MCP tries ten times to do an error-free READ without success. To reduce the likelihood of encountering a fatal error, attributes can be specified so that multiple copies of a disk file are created and maintained by the I/O Subsystem.

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. The MCP maintains a separate disk file for each copy of the duplicated file.

LOGICAL I/O

Duplicated files can be requested by specifying the Boolean attribute DUPLICATED when the file is created. Within duplicated files, there is no original; that is, because all the files are identical, all images are referred to as copies.

The value of the COPIES attribute indicates the total number of physical area images that exist for the logical area; for example, COPIES equal to 2 indicates two physical areas for each logical area.

Those file attributes that apply to a physical area (AREAALLOCATED, FAMILYINDEX, and ROWADDRESS) can be qualified to indicate the copy number. For example, in the phrase "FID(m,n).ROWADDRESS", m is the area number and n is the copy number.

Blocked duplicated files behave exactly the same way as blocked nonduplicated files; logical reads and writes do not always cause corresponding physical I/O operations.

Each time a physical write must be done, the block is serially written to all copies. The buffer containing the block is tied up until all copies have been written. If any physical write encounters an error, the attention bit (bit 1) is set in the logical result descriptor.

Each time a logical read causes a physical read, the MCP initiates only one physical read from only one copy (unless an error occurs on that read). Successive physical reads are initiated to successive copies of the specified area, where the next copy is determined on a rotational basis.

When an error occurs on a physical read, the MCP causes another physical read, but on the next copy of the corresponding area. An error on any block causes the attention bit (bit 1) to be set. All logical reads from that block will have the attention bit (bit 1) set in the logical result descriptor. An error on all blocks causes the exception bit (bit 0) to be set. Action labels are triggered by the setting of the exception bit (bit 0), so branching occurs only if all copies are in error.

Direct I/O read and/or write operations are not allowed on duplicated files.

See also
AREAALLOCATED 114
COPIES. 150
COPYINERROR 151
COPYNAME. 152
DUPLICATED. 171
FAMILYINDEX 181
ROWADDRESS. 276

RESULT DESCRIPTORS

An "overall result" is available for each logical READ or WRITE statement.

The overall result for a duplicated file I/O operation contains the following information in addition to that for nonduplicated files:

Bits [27:16]	Error bit for each copy. Bit 12 is copy number 1, bit 13 is copy number 2, and so forth. Bit 27 is unused. This field is meaningful 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 is 1) only if all copies are in error.

Refer to the description of the STATE attribute under "General File Attributes" for a full explanation of results of I/O statements.

Example (ALGOL)

Assume the following statements are in an ALGOL program:

```
BOOLEAN RESULT;

RESULT := READ (F, 10 A[*]);
```

RESULT contains the overall result of the READ statement. A fatal error could be detected by means of the following:

```
IF RESULT THEN HANDLECATASTROPHE;
```

However, when using duplicated files, the surveillance for a bad copy could be performed as follows:

```
IF RESULT.[1:1] THEN HANDLEACOPYPROBLEM;
```

Because the attention bit ([1:1]) is TRUE if any copy has a problem, then it is conceivable that more than one copy has a problem.

The following produces an integer count of how many copies have problems:

```
X := ONES (REAL(RESULT.[27:16]));
```

To find out which copies had errors, the COPYINERROR attribute can be used, as follows:

```
Y := F.COPYINERROR;
WHILE Y NEQ 0 DO
    BEGIN
    <handle a problem on copy Y>
    Y := F.COPYINERROR
    END;
```

See also
COPYINERROR 151
STATE 299

LIBRARY MAINTENANCE FUNCTIONS

Because 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 by means of an ODT command or through 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 to be opened, the copies are found. If one of the copies' names is changed (to B/C/X, for example), then when the duplicated file A/B is opened, the file B/C/X would not be considered a copy and (in this example) the number of copies would then be two.

7 HARDWARE AND SOFTWARE TRANSLATION

The MCP provides for either hardware or software translation, depending on the KIND value of the file and the physical and logical character sets in which the file is encoded. Table 7-1 should be read in conjunction with this description because it shows how translation is handled for each combination of a KIND value and a physical and logical character set.

The system takes advantage of hardware translation wherever the combination of KIND, INTMODE, and EXTMODE values require it and the hardware allows it. Software translation is provided in most cases where hardware translation is not provided (as shown in Table 7-1). However, software translation is restricted to files with data stored in HEX, BCL, EBCDIC, or ASCII character mode; binary data (SINGLE or BINARY) cannot be translated and must remain unaltered in its 48-bit pattern.

Certain constraints apply to software translation as well. As previously mentioned, binary data cannot be translated; as a result, logical files having an INTMODE or EXTMODE value of SINGLE are never translated. The same constraint holds true for card files with an EXTMODE value of BINARY. The FORTRAN-linked record format (where BLOCKSTRUCTURE is LINKED or FILETYPE is 6), which implies that the data is binary, and the generalization of the COBOL variable-length record format (where FILETYPE is 4), which allows the mixing of modes in the records, are not translated. Nor do Direct I/O and port files receive software translation; port files cannot have an INTMODE or an EXTMODE of either HEX or BCL.

Whether or not translation is involved for a file, the attributes AREALENGTH, BLOCKSIZE, CURRENTBLOCK, CURRENTRECORD, MAXRECSIZE, and MINRECSIZE are given in terms of the logical units of the file as defined by the FRAMESIZE (or UNITS and INTMODE) attribute. 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.

Word-oriented data transfers, where the character framesizes differ between the internal mode and external mode, require either contraction or expansion of the records; hence, the logical and physical record and block sizes are not the same. For example, for an EBCDIC and a BCL word-oriented card file, each logical record contains 80 characters, but the record size of the EBCDIC file is 14 words and that of the BCL file is 10 words.

Individual attribute descriptions in "General File Attributes" provide detailed explanations of each attribute mentioned in this description of translation.

DEFAULTTRANS

Originally, the system only performed translation when the I/O hardware provided it and DSed any other programs having files that required translation. In actuality, some programs were not caught, and translation was not performed even though it was appropriate, and the programs were allowed to proceed (refer to "Explanation of Table 7-1" for details). When software translation for I/O was implemented, the system then had the capability to translate in all appropriate cases where INTMODE and EXTMODE were different.

However, because some programs had been written to use the system as it had been before software translation was implemented (those cases where programs were not caught), one method of translation was implemented that used software translation only in those cases where programs were previously caught and DSed and still allowed the other cases to circumvent translation.

This method of translation is implemented as the DEFAULTTRANS value of the TRANSLATE file attribute. DEFAULTTRANS was made the default value of that attribute to ensure that old programs that relied on the old semantics ran as they did before software translation was implemented.

Because DEFAULTTRANS is only the result of an oversight and does not function in a reasonable, consistent manner, it is preferable to perform translation whenever INTMODE and EXTMODE differ and translation is appropriate, which corresponds to the FULLTRANS value of the TRANSLATE file attribute. FULLTRANS is therefore recommended over DEFAULTTRANS.

In order to allow all installations to control selection of the default, a run-time system option named FULLTRANSLATION has been added, which when set changes the default value of TRANSLATE to FULLTRANS on B 6800, B 7700, and B 7800 systems. DLP-based systems always use FULLTRANS as the default value of TRANSLATE, regardless of whether or not the FULLTRANSLATION run-time option is set. (Refer to the description of this option under the OP (OPtions) ODT command in the Operator Display Terminal (ODT) Reference Manual.) Thus, portability problems may arise when changing from earlier systems to later systems, due to the differing actions of DEFAULTTRANS and FULLTRANS, but in these cases, the default can be overridden by setting the value of the TRANSLATE attribute in the program.

If a program is to access an existing file whose EXTMODE is not equal to INTMODE, then the value of the TRANSLATE file attribute must be the same each time the program is executed. Portability problems are possible for programs that access such files without explicitly setting TRANSLATE, because the system-supplied default value for the TRANSLATE file attribute depends on the machine type and on run-time option settings.

In order to notify users of this problem, a warning mechanism has been implemented, which will be active if and only if the run-time option TRANSWARNINGS is set. The warnings are given for the conditions identified by an "F" in table 7-1, as explained below. The code file is marked as having been given the warnings. This allows programs with portability problems to be identified by means of the LFILE command in CANDE or by running SYSTEM/FILEDATA.

B 6800, B 7700, and B 7800 (Non-DLP-based) Systems

On B 6800, B 7700, and B 7800 systems, the default value for TRANSLATE is governed by the FULLTRANSLATION run-time option. If the option is set, the default value is FULLTRANS, otherwise the default value is DEFAULTTRANS. However, on DLP-based systems, the default value for TRANSLATE is always FULLTRANS regardless of the value of the FULLTRANSLATION run-time option.

Therefore, programs accessing permanent files with INTMODE different from EXTMODE and with TRANSLATE = DEFAULTTRANS by default are not portable to DLP-based systems. Running the same program against the same file on a DLP-based system will use a default value of TRANSLATE = FULLTRANS, resulting in incorrect data access and/or file corruption.

A program that opens an existing file with INTMODE not equal to EXTMODE on a non-DLP-based system is now issued the following warning if the TRANSLATE attribute is not specified, the FULLTRANSLATION run-time option is reset, and the TRANSWARNINGS run-time option is set:

NOT PORTABLE TO DLP-BASED SYSTEMS; SEE 3.5 DNOTE D06065

B 5900, B 6900, B 7900, and A Series (DLP-based) Systems

To assist users who have already migrated to DLP-based systems, an RSVP warning system has been implemented. When a program tries to open a file with INTMODE not equal to EXTMODE (under the conditions identified by an "F" in table 7-1), the program is issued the following RSVP warning if the TRANSLATE attribute is not specified and the TRANSWARNINGS run-time option is set:

UNEXPECTED RESULTS MAY OCCUR;SEE 3.5 DNOTE D06065. THE TRANSLATE ATTRIBUTE NOW PROPERLY DEFAULTS TO FULLTRANS.

The valid replies are "DS" or "OK". A reply of "DS" will abort the program execution. A reply of "OK" will cause the program to proceed using the value of FULLTRANS for the TRANSLATE attribute. Incorrect data access or data corruption might occur if the program is not portable to DLP-based systems. By "not portable", it is meant that the program might have migrated from a non-DLP-based system to a DLP-based system and depends on the default value of DEFAULTTRANS for the TRANSLATE file attribute.

Note that the issuance of this warning does not necessarily mean that the program will function incorrectly, only that it might do so. This ambiguity results from the inability of the MCP to determine how a given program is intended to work. If a program is a "native" DLP-based system program and has, from its inception, assumed that the default value for TRANSLATE is FULLTRANS, then the warning is unnecessary. However, if a program has migrated from a non-DLP-based system to a DLP-based system and depends on the default value for TRANSLATE being DEFAULTTRANS, the warning may prevent incorrect data access or data corruption.

It is strongly recommended that all users of DLP-based systems eliminate occurrences of this warning by program recompilation or run-time file equation with TRANSLATE set explicitly to DEFAULTTRANS or FULLTRANS, as required.

If an undesirably large number of warnings are observed, the user can temporarily suppress the RSVP by resetting the TRANSWARNINGS run-time option, which has the same effect as replying "OK" to the RSVP warning.

COBOL AND COBOL74 CONSIDERATIONS

Translation requires special considerations when programming in COBOL, which makes no provision for character set translation in the standard language definitions. (However, a Burroughs extension to COBOL provides for translation whenever it is required by MOVE statements.) Because the MCP requires all logical files to have an INTMODE that informs the system whether or not translation is required, the COBOL compiler sets the INTMODE to the type of the first 01-level entry under the File Description (FD). For example, if the first 01-level entry is USAGE DISPLAY, then INTMODE is EBCDIC; if the first 01-level entry is USAGE COMP-2, then INTMODE is HEX; and if the first 01-level entry is USAGE COMP and the B 2500 compilation option ($B2500) is set, then INTMODE is also HEX. Thus, the order of declaration of 01-level items under an FD can make a difference if the level items are of different USAGEs.

Consider the following COBOL program fragments, assuming the $B2500 option is SET:

```
FD  FILEID-A.
01  RECID-1     PIC 9(20)  USAGE DISPLAY.
01  RECID-2     PIC 9(40)  USAGE COMP-2.

FD  FILEID-B.
01  RECID-2     PIC 9(40)  USAGE COMP-2.
01  RECID-1     PIC 9(20)  USAGE DISPLAY.
```

If two programs have the same record descriptions but the 01-level entries occur in a different order, then the compiler will set the INTMODEs differently. The INTMODE for "FILEID-A" in the preceding example is EBCDIC; the INTMODE for "FILEID-B" is HEX.

If the file is created with "FILEID-A", the EXTMODE of the physical file will be EBCDIC. If the file is then read with "FILEID-B", whose INTMODE is HEX, undesired translation will occur.

The problem discussed in the preceding paragraphs does not occur if all USAGEs of the same physical file have a COPY clause after the FD, which copies the record descriptions from the same copy library file. This clause ensures that all FDs have the same order of record declarations.

DEVICE-SPECIFIC TRANSLATION

Table 7-1 shows the interrelationship of the TRANSLATE and KIND attributes.

Table 7-1. Table of Translation

INTMODE	EXTMODE	PACK or DISK	PRINTER	PUNCH or READER	TAPE7	TAPE9 or TAPEPE	ODT or REMOTE	PAPER TAPE	PORT BNA V2
H	B	F	<E>	F	F	F	-	D	-
H	E	F	F	F	F	F	D	D	-
H	A	D	<E>	-	D	D	-	D	-
B	H	F	<E>	-	F	F	-	D	-
B	E	D	H	D	F	F	D	D	-
B	A	D	<E>	-	D	D	-	D	-
E	H	F	<B>	-	F	F	-	D	-
E	B	D	H	H	H*	F	-	H	-
E	A	D	<B>	-	D	D	-	H**	P
A	H	D	<E>	-	D	D	-	<E>	-
A	B	D	<E>	D	D	D	-	<E>	-
A	E	D	D	D	D	D	D	D	P

A means ASCII

B means BCL

E means EBCDIC

H means HEX

D means soft translation done if TRANSLATE is FULLTRANS or DEFAULTTRANS.

F means soft translation done only if TRANSLATE is FULLTRANS.

H means hardware translation

P means, for BNA V2 port files, FULLTRANS or USERTRANS

\- means invalid combination of INTMODE and EXTMODE for that device

A letter enclosed in broken brackets (< >) indicates an invalid combination of INTMODE and EXTMODE for that device. In this case, the system automatically changes EXTMODE to the value contained in broken brackets (ASCII, BCL, EBCDIC, or HEX).

For an explanation concerning the * and ** items, refer to "Explanation of Table 7-1".

EXPLANATION OF TABLE 7-1

In Table 7-1, an "F" appearing under a file KIND indicates the cases in which FULLTRANS performs translation and DEFAULTTRANS does not:

1. For disk and pack, when INTMODE is HEX and EXTMODE is either EBCDIC or BCL, or when INTMODE is either EBCDIC or BCL and EXTMODE is HEX.

2. For tapes, when both INTMODE and EXTMODE are EBCDIC, BCL, or HEX, that is, when neither INTMODE nor EXTMODE is ASCII or SINGLE. For TAPE7, DEFAULTTRANS performs translation correctly if INTMODE is EBCDIC and EXTMODE is BCL.

3. For printers, whenever INTMODE is HEX.

4. For card readers and card punches, any valid case where INTMODE is HEX.

TAPE7 and PAPERTAPE files (indicated by "*" and "**", respectively, in Table 7-1) require a more thorough explanation. For TAPE7 (7-track tape) files, hardware translation or transformation occurs in two ways that do not apply to any other type of physical unit. Nonstandard (alpha, even) parity 7-track tapes always have an EXTMODE value of BCL, and hardware translators allow the INTMODE value to be either BCL or EBCDIC. Standard parity tapes coming from the B 3500 require BCL-to-BCL or EBCDIC-to-EBCDIC transformation through the hardware. All other combinations are handled in 6-bit binary. When hardware is concerned, as in the nonstandard parity case, all INTMODE values other than BCL are assumed to have an 8-bit frame size.

PAPERTAPE files had a special use for the combination "INTMODE=EBCDIC, EXTMODE=ASCII" that was implemented before the system handled 8-bit ASCII. With this combination of modes and the attribute TRANSLATE specified as DEFAULTTRANS, the data is sent through the hardware translators. Otherwise, ASCII is transferred in 8-bit binary exactly as EBCDIC is handled.

See also

8 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, which denotes a system-library file, or by a usercode in parentheses, which denotes a user-library file. As many as 13 identifiers may be used in constructing a file name.

EXAMPLES OF FILE NAMES

In the following 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. The third and fourth examples would be preserved on the permanent tape file as D/F and G/J.

```
A
B/C
D/E/F
G/H/I/J
(U)V
```

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 that identifies the file. The FAMILYNAME specifies the physical disk family on which that file is stored.

FILENAME Specification

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/B 5000 Series system hierarchical disk file directory) that is identified by the usercode. In normal situations, the user supplies an arbitrary file name (consisting of 1 to 13 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 nonprivileged 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, in WFL:

```
FILE F (FILENAME = *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 permitted between the closing parenthesis and the start of the file name. For example, in ALGOL:

```
FILE F(FILENAME = "(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 asterisk (*) 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 (for a nonprivileged task).

Examples

The following examples are syntactically correct.

RUN statement example:

```
RUN (HISUSERCODE)HISDIRECTORY/HISPROGRAM
```

File equation example:

```
FILE ZZZ (FILENAME = *SYSTEMFILE)
```

Task code file TITLE example:

```
REPLACE T.NAME BY "(AUSERCODE)PROGRAMNAME.";
```

FAMILYNAME Specification

The selection of random-access, mass-storage devices varies depending on the value of the FAMILYNAME attribute. The system supplies a FAMILYNAME of "DISK" unless it has been explicitly set to another value.

9 DISK FILE AND SYSTEM-ACCESS SECURITY

INTRODUCTION

The system provides a mechanism for identifying system users, restricting 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 a unique 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 may access 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 specially defined to utilize this function. A privileged program may remove the requirement of a privileged usercode. Refer to the description of the PP (Privileged Program) ODT command in the Operator Display Terminal (ODT) Reference Manual.

USERCODES AND PASSWORDS

Usercodes and passwords are identifiers satisfying the same syntax rules as file identifiers; each contains up to 17 EBCDIC characters, normally digits and upper-case letters. Nonalphanumeric characters may be included only if the identifier is enclosed in quotation marks; the quotation marks are not counted in the identifier length, and the identifier may not contain a quotation mark.

The usercode is a matter of limited public record. It is assigned and controlled by the installation management and displayed on various output media. The password should be known only to the usercode holder. It can be assigned or changed by him; it is never displayed.

Assignment of Usercodes

A usercode is defined by its appearance in a file called USERDATAFILE, which is maintained by the installation management to keep information about system users. Facilities to create and maintain this file are described in the MAKEUSER section of the System Software Site Management Reference Manual. New usercodes may be created through the MAKEUSER program or the intrinsic USERDATA. At the installation's option, the system operator may create usercodes through the MU (Make User) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.)

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 desirable 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. Subject to installation-imposed restrictions, passwords may be assigned or changed by individual users.

Accesscodes and Passwords

An additional level of control may be imposed on the access of a file by means of 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 ODT) or by entering information from a terminal by way of the Data Communications Subsystem. In the former case, the access is controlled by the Work Flow Language (WFL) compiler; in the latter case, a Message Control System (MCS) program is involved. Once a program is running, it can access files belonging to other usercodes by means of the Inter-Process Communication (IPC) constructs in various programming languages or by means of a ZIP statement, provided that security criteria are met.

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 share the same 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 "/password" clause is omitted if no password is associated with that 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) ODT command; it may be relaxed through the "SR-" version of this command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this command.)

Remote Stations

Individual MCS programs may accept or require usercode specifications. CANDE requires all users to log on with a usercode. At the option of the system operator, RJE can accept or require a usercode specification to log on to the RJE terminal and to run individual RJE jobs. The USER statement is used in RJE jobs in the same manner as it is used in WFL jobs. In addition, RJE has the capability of running each job with a default usercode, that is, the one with which the terminal was logged on.

Tasks

When a task is RUN, PROCESSed or CALLed by means of the IPC mechanism, a usercode/password may be supplied by means of the pointer-valued task attribute USERCODE.

For example, the ALGOL syntax is as follows:

```
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" with a new usercode/password combination. A task running without a usercode may acquire one by using 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 has privileged status).

ZIPped Arrays

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.

DISK FILE ACCESS

Access to a given DISK is controlled by three attributes of the file: SECURITYGUARD, SECURITYTYPE, and SECURITYUSE. These attributes are also described under "General File Attributes".

SECURITYGUARD Attribute

If SECURITYTYPE is GUARDED or CONTROLLED, then the title of the guard file is specified by the SECURITYGUARD attribute. The guard file may be constructed using the GUARDFILE utility. (For a description of the GUARDFILE utility, refer to the System Software Utilities Reference Manual.)

If SECURITYGUARD is not specified for a GUARDED or CONTROLLED file or if the specified guard file cannot be found on the specified pack, then the file is treated as PRIVATE. If the specified pack for the guard file is not present, the system message "REQUIRES PK" is displayed (unless the opening program is an MCS, in which case the open attempt fails with a FILE NOT AVAILABLE condition). Appropriate operator action is required to either continue or terminate the process of accessing the file.

SECURITYTYPE Attribute

The SECURITYTYPE attribute specifies who, apart from the owner, may access the file. The attribute values are CONTROLLED, GUARDED, PRIVATE, or PUBLIC. CONTROLLED is equivalent to GUARDED except the guard file will be examined even for the creating (owning) usercode. GUARDED indicates that permission is required from the guard file (identified by the SECURITYGUARD attribute) before a nonprivileged user, other than the owner, may access the file. 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)FILENAME form of the file name.

The default value of SECURITYTYPE is PRIVATE for files created by programs that run under a usercode; otherwise, the value is PUBLIC.

SECURITYUSE Attribute

Regardless of the the value of SECURITYTYPE, the use of a file is controlled by the SECURITYUSE attribute, which may have values of IN, IO, OUT, or SECURED. IN indicates that the title may be read only. IO indicates that the file may be both read or written to. OUT indicates that the file may be written to only. SECURED indicates that the file may be neither read nor written to. 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: (1) if SECURITYTYPE is PRIVATE, the usage for that program is set to SECURED, (2) if SECURITYTYPE is PUBLIC, usage is determined by SECURITYUSE, (3) if SECURITYTYPE is GUARDED or CONTROLLED, usage is determined from data in the guard file 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 non-owner is permitted for a PUBLIC file independent of SECURITYUSE and is controlled for a GUARDED or CONTROLLED file by the data in the guard file. In an Inter-Process Communication (IPC) environment, the security tests for the code files are based on the usercode of the new task, not that of the parent task.

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 (that is, 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. When interrogating the attribute, if the file is not open, then the attribute must have been explicitly specified prior to the attempt to read the attribute 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 (because those privileges were determined when the file was opened).

b. When modifying 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 is changed whenever the file is opened. Note that once a security attribute has been set in a file variable, it is acted upon every time the file is opened. Changing a security attribute on an open file causes the permanent copy of the attribute value to be changed, but it has no effect on the privileges of any program that already has the file open.

LIBRARY MAINTENANCE SECURITY

A user may remove or change the FILENAME of any file that "belongs" to him. A user may copy any file that he is allowed to read, with the exception that the "COPY =" function is restricted to the user's own files. A nonprivileged user may copy any files from a library maintenance tape, but only into his own usercode library.

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 USERDATAFILE entry. The privilege may be afforded or withdrawn by the installation management through the MAKEUSER program or (at installation option) by the system operator through the MU (Make User) ODT Command. (Refer to the discussion of the MAKEUSER utility program in the "MAKEUSER" section of the System Software Site Management Reference Manual.)

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 (Privileged Program) ODT command. (Refer to the Operator Display Terminal (ODT) Reference Manual for a description of this 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 by means of the TASKATTERR task attribute. If the new usercode/password

combination is valid, then the task's privilege status is set to the privilege status of the new usercode. A privileged task may remove its usercode association. The following is an ALGOL example:

```
REPLACE MYSELF.USERCODE BY ".";
```

The privileged status of the task is not affected when its usercode association is removed.

For maximum security, installations should carefully limit the creation of privileged usercodes and programs and monitor their use.

SECURITY VIOLATIONS

Security violations detected during an open operation are fatal to the program.

If a read or write is used as a Boolean primary and a violation is detected during that read or write, then the violation is nonfatal and bit 16 is set in the Software I/O Result Descriptor. (Refer to the description of the STATE attribute under "General File Attributes".)

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 name of the violation subject (typically the file in question). Violation codes are defined with the format of the log messages and are further discussed in the SUMLOG section of the System Software Support Reference Manual.

See also
STATE 299

A FORMAT OF LIBRARY MAINTENANCE TAPES

The basic format of a library maintenance 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, and so forth, with MTA/FILE000 being the tape directory.

The general tape layout of a library maintenance tape is as follows:

VOL1

HDR1

HDR2

tapemark

(The directory, which consists of one or more records of 1024 words each. The last record may be shorter.)

tapemark

EOF1

EOF2

tapemark

HDR1

HDR2

tapemark

First dumped file, beginning with one record, which is the disk header, followed by the actual file data in 901-word records (2701-word records for GCR tapes). The last record may be shorter, that is, no padding for short records.

```
          tapemark

     EOF1

     EOF2

          tapemark
          .
          .
          .

     More dumped files

          .
          .
          .

     EOV1 (if going to another volume)

          tapemark

          tapemark
```

The first word in all records is a transaction number. In the tape directories, the number in the first record is -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 1024 words long consisting of a 1-word transaction count and one word of link information, followed by up to 1022 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 multivolume 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

Bits [1:2]:

0 For MCP use
1 Neither a usercode nor "*" was specified.
2 System file ("*" specified)
3 Usercode specified
(The first identifier is the usercode.)

Bit [2:1]:

0 No FAMILYNAME is specified.
1 The last identifier is the FAMILYNAME.

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 volume contains a partial copy of the tape directory, containing entries for those files not dumped on the preceding reels. Thus, a COPY of a file dumped to the third volume, for example, need only examine the third volume because it can determine 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]
MOREINFO	[12:13]

CHARCNT and SKIP are needed because the directories of the succeeding volumes 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 volume. CHARCNT indicates 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 indicates how many complete file names to skip over in order to get to the first complete file dumped to this volume.

MOREINFO is the relative number of the file that 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 the time of volume change. The example is of a multivolume dump, with the following assumptions:

n files were dumped.

Two volumes were used.

The (k-1)th file was being written at the time of volume change.

Two directory records were used.

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-1 (end)
2nd directory record          Names:  end of k-3 through n
file k
.
.
.
file n
```

Note that the directories of any but the first volume will not be the first item on that volume. Reel switch will be sensed while in the middle of a file. The remainder of that file will be written onto the new volume before the directory. The split file is logically considered to be on the volume on which it was begun.

The directory on succeeding volumes is again called MTA/FILE000. Furthermore, the subsequent files are numbered sequentially, starting where the preceding volume left off, and have in their label records that this is their first FILESECTION. Only files actually split over two volumes in their label records will have a "FILESECTION 2" designation.

B FORMAT OF PACK LABELS

Position	Length (bytes)	Contents
000-003	004	"VOL1"
004-009	006	Pack Serial Number
010	001	Reserved
011-027	017	Pack Identification (FAMILYNAME)
028-029	002	System-Interchange Code Native Mode = 67 Interchange = 00
030	001	Reserved
031-036	006	Reserved
037-050	014	Owner's Identification
051-056	6	Mirrored disk date
057	1	Relative index of pack within set
058	1	Mask of on-line mirrored copies
059	1	Recovery option "D" = DMS " " = DISCARD
060	1	Auditing marker "V" = Set closed "C" = Creation or Audit application
061-078	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>

Position	Length (bytes)	Contents
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
119	001	Integrity Flag
120-125	006	Reserved
126-131	006	Reserved
132-179	048	Reserved
180-359	180	Reserved for Security Information

C FORMAT OF INTERCHANGE DIRECTORY RECORDS

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

D FORMAT OF INTERCHANGE FILE HEADERS

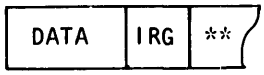
Position	Length (bytes)	Contents
000-005	006	Core Address
006-013	008	Self Pointer
014-017	004	Header Size
018-019	002	File Type
020-031	012	Reserved
032-037	006	Record Size
038-040	003	Records-Block
041-049	009	Block Size
050-055	006	Blocks-Area
056-061	006	Sectors-Area
062-065	004	Areas Requested
066-069	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
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

Position	Length (bytes)	Contents
166-167	002	Number Open Output
168-199	032	Reserved
200-207	008	First Area Link
208-359	152	19 More Area Links

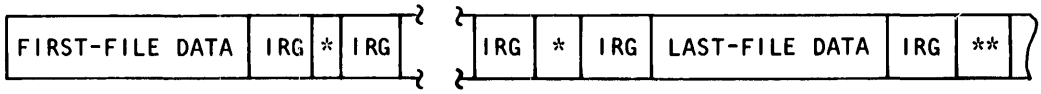
E STANDARD TAPE LABEL FORMATS

Format of Unlabeled Tapes

SINGLE FILE REEL



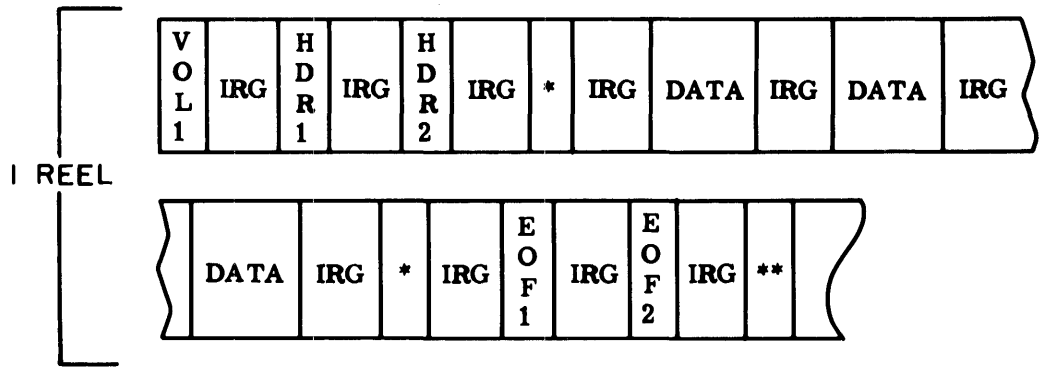
MULTIFILE REEL



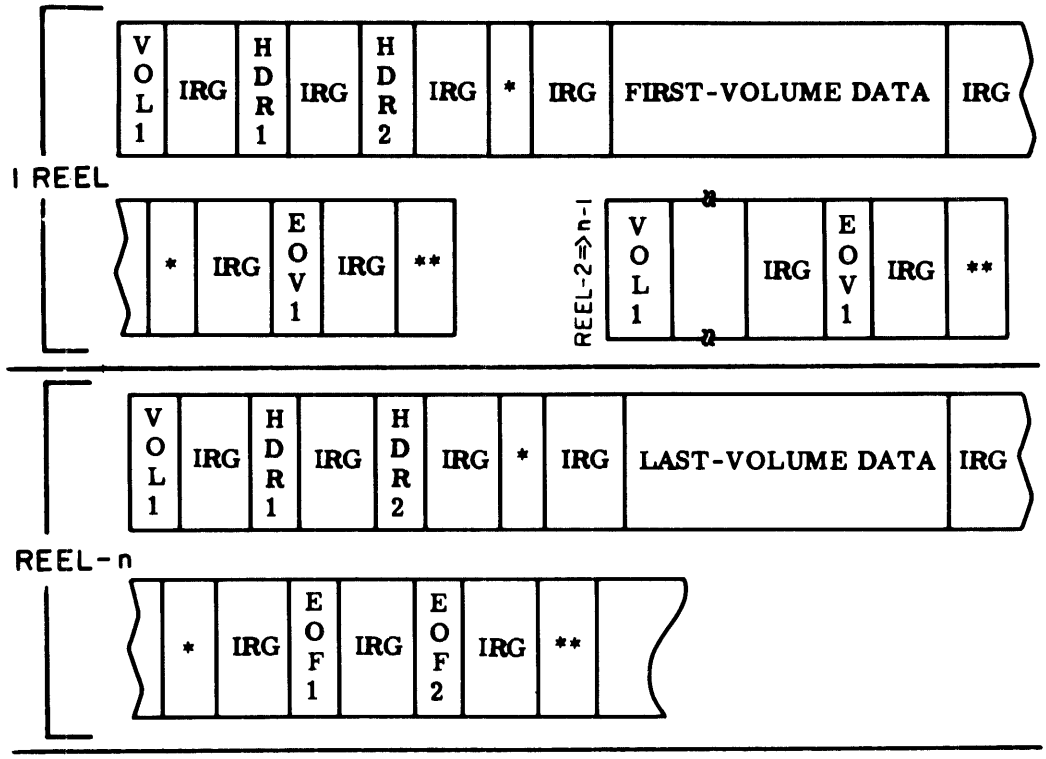
Note: Symbol * denotes a tape mark. A double tape mark consists of two tape marks separated by an inter-record gap.

Format of Labeled Tapes

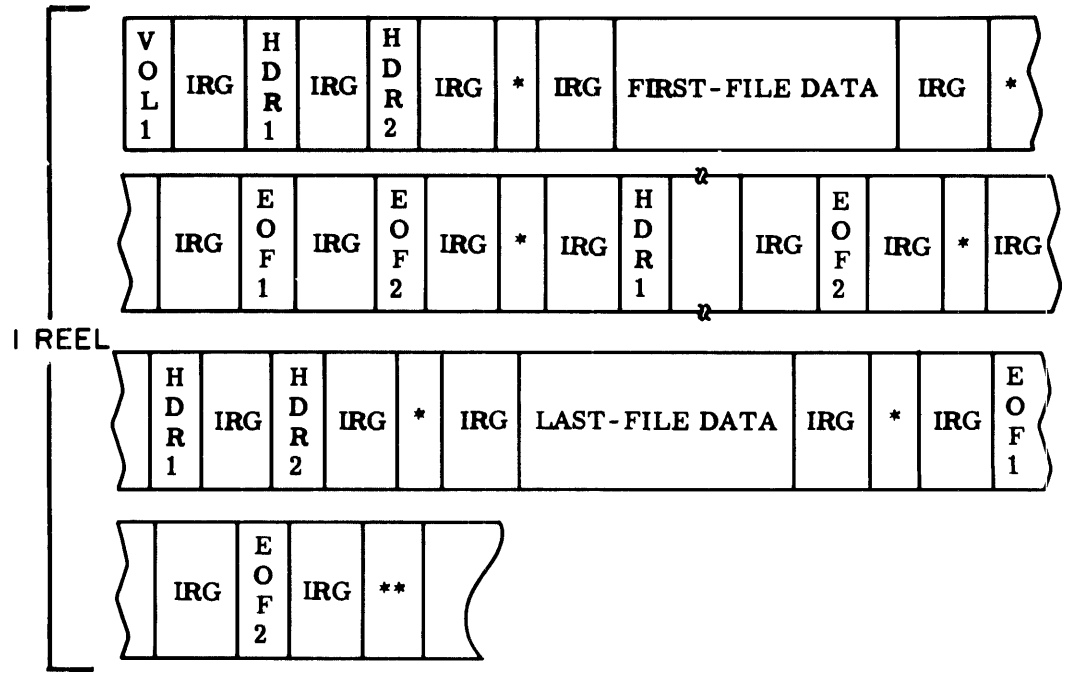
SINGLE FILE, SINGLE REEL



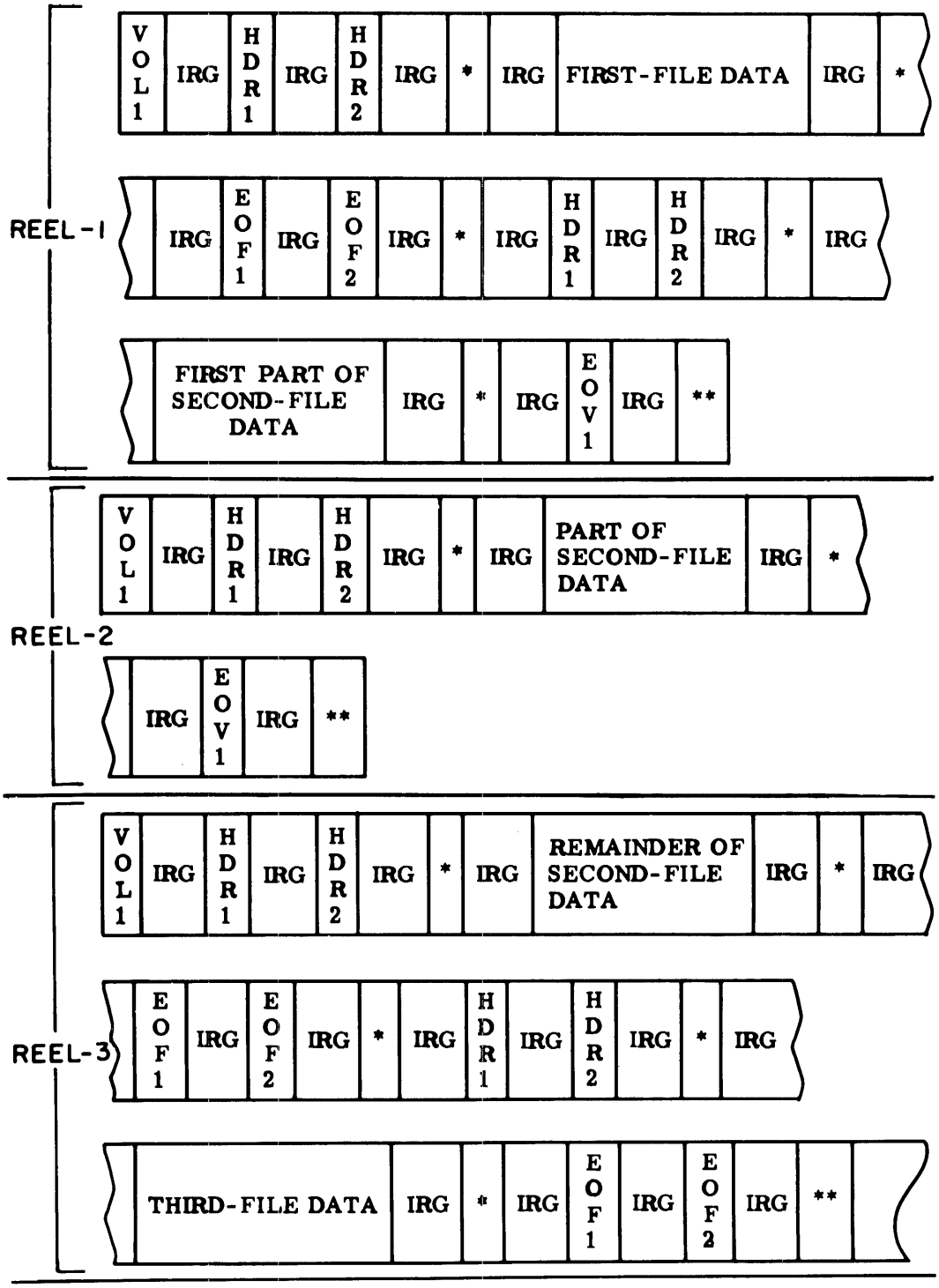
MULTIREEL FILE



MULTIFILE REEL

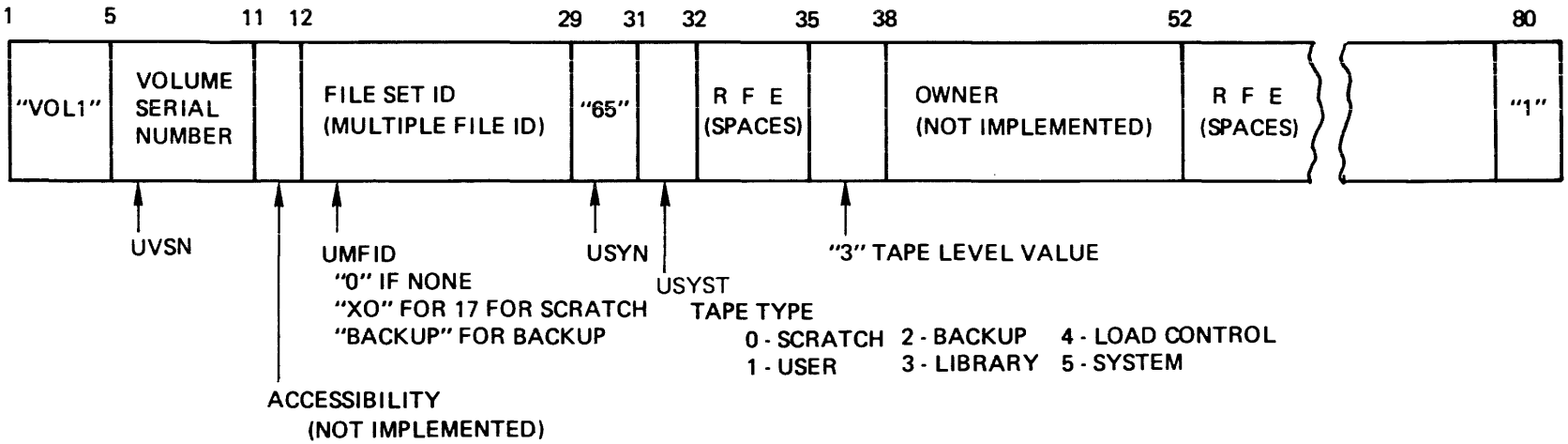


MULTIFILE, MULTIREEL

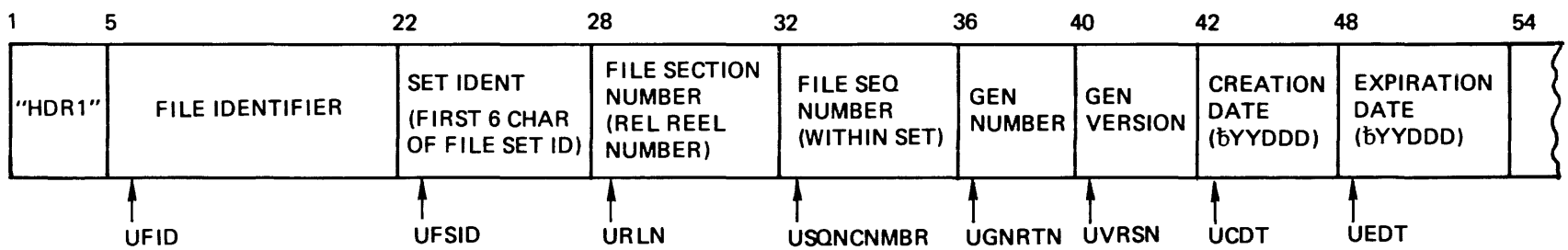


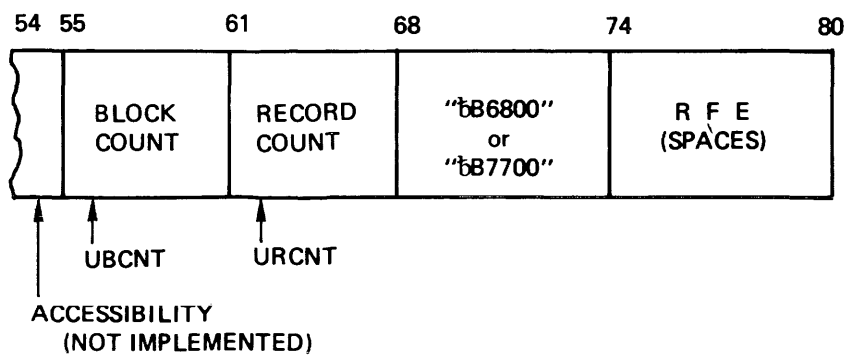
Notes: 1. Symbol * denotes a tape mark.
2. User's header labels may appear immediately after HDR2, and user's trailer labels may appear after either EOF2 or EOV1.

VOLUME HEADER

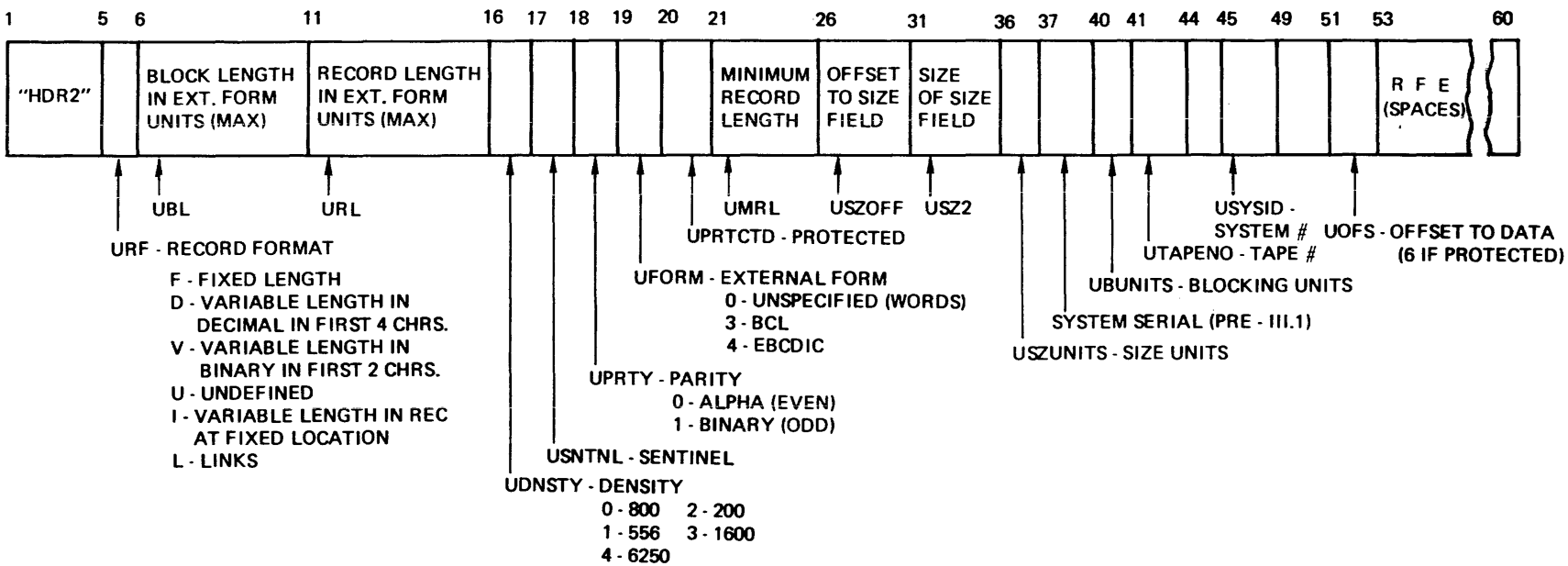


FIRST FILE HEADER





SECOND FILE HEADER



FIRST END-OF-FILE LABEL

SAME AS FIRST FILE HEADER, EXCEPT FOR FIRST FOUR CHARACTERS: "EOF1."

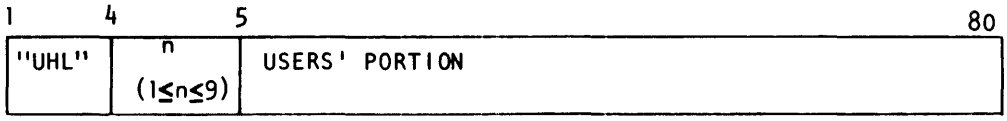
SECOND END-OF-FILE LABEL

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."

USERS' HEADER LABEL



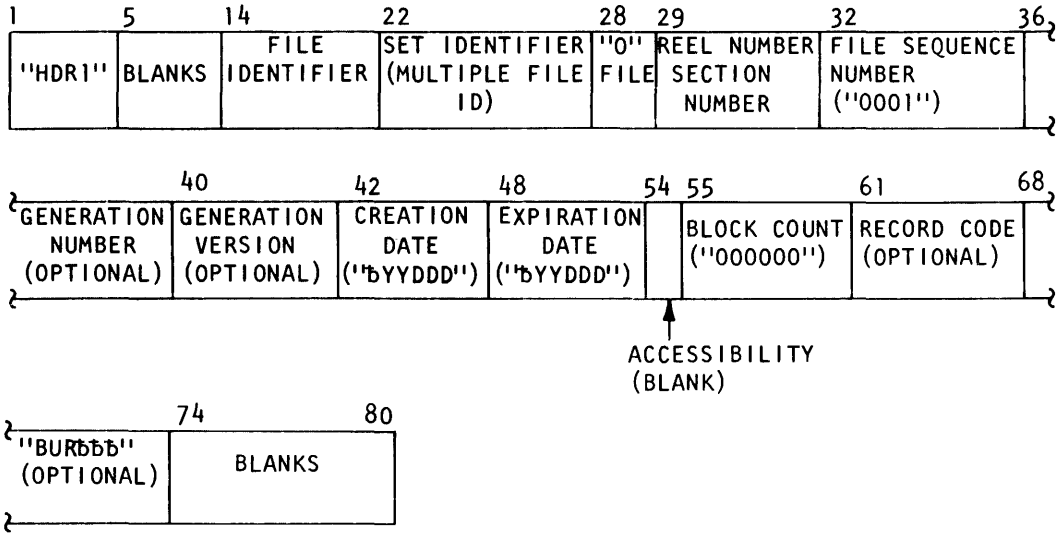
USERS' TRAILER LABEL

Same as users' header label, except for first three characters: "UTL."

B 3500 USASI Tape Label

VOLUME HEADER

FIRST FILE HEADER



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 among 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-volume, 0 = end-of-file)
6	1-5	41-45	Block count
6-7	6-8 1-4	46-52	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 used)
9-end	1-end	65-end	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)

F FORTRAN77 PROGRAMS

FORTRAN77 programs behave slightly different from programs written in other languages, as explained below.

The following semantics for the TITLE attribute are implemented only for FORTRAN77 programs (FORTRAN77 refers to this attribute as "FILE").

a. If an attempt is made to set the file name with the TITLE attribute, then an implicit close is performed if the file is already open and the name of the file is different from the current name.

b. If the name of the file remains the same and the file is open, then no error is given.

An implicit close as in "a" above results in the closing of the physical file and a title change in the logical file. Any subsequent open creates a new physical file with the new file TITLE.

An attribute error will be suppressed if the following attributes are modified by a FORTRAN77 program while the file is open and if the resulting action would not change the attribute's value:

MAXRECSIZE
NEWFILE
PROTECTION
UNITS

UNDERSTANDING RAILROAD DIAGRAMS

WHAT IS A RAILROAD DIAGRAM?

A railroad diagram is a way of representing the syntax of a command or statement graphically. It shows which items are required or optional, the order in which they should appear, how often you can repeat them, and any required punctuation.

HOW TO READ A RAILROAD DIAGRAM

Normally, you read a railroad diagram from left to right. However, there are some exceptions; in those cases, arrows indicate a right-to-left direction.

If a diagram is too long to fit on one line and must continue on the next, a right arrow (>) appears at the end of the first line and another at the beginning of the next line, like this:

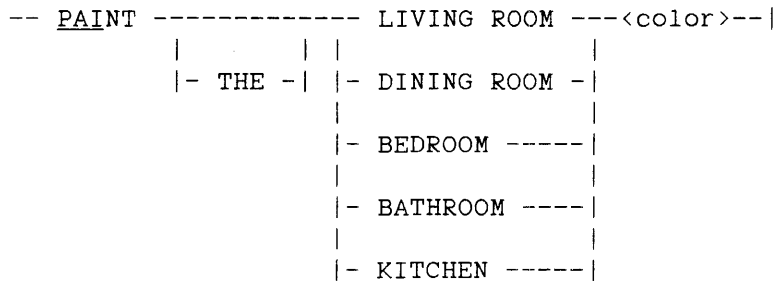
```
-------------------------------------------------------------------->

>-----------
```

The end of a railroad diagram is denoted by a vertical bar (|) or percent sign (%). The vertical bar means the command or statement can be followed by a semicolon and another command or statement. The percent sign means the command or statement must be on a line by itself.

CONSTANTS AND VARIABLES

Consider a hypothetical command for giving instructions to a house painter:



This command tells the painter to paint a designated room in the color you specify.

The example introduces two important features of railroad diagrams:

- Constants
- Variables

Constants

Constants are items that you cannot vary. You must enter a constant as it appears in the diagram, either in full or abbreviated. If you abbreviate a constant, you must enter everything that is underlined in the railroad diagram, optionally followed by one or more of the remaining characters.

You can recognize constants in railroad diagrams by the fact that they are never enclosed in angle brackets.

In the example, the word PAINT is a constant. You could enter PAINT in full or abbreviate it to PAI or PAIN, but not to PA or PAN. If no part of the constant is underlined, you cannot abbreviate it at all.

Variables

Variables are items that you can replace with other data to suit a particular situation; that is, you can vary the information you enter in place of the variable, subject to rules defined for the particular command or statement.

Variables appear in a railroad diagram enclosed in angle brackets (<>).

In the example, <color> is a variable item. If the description of the PAINT command defines the allowable colors as BLUE, GREEN, PINK, and YELLOW, you can enter any one of these in your command.

FOLLOWING THE PATHS OF A RAILROAD DIAGRAM

The paths of a railroad diagram lead you through the diagram from beginning to end. They are represented by horizontal and vertical lines.

A path shows the allowable syntax. Some diagrams have just one path that goes from the beginning to the end of the diagram. Others contain several paths, each covering a part of the diagram. A path shows which items you can include in a command or statement, which you can omit, and the number of times you can include a particular item or group of items.

To follow a path through a railroad diagram, you need to understand the items you may encounter along the way. These items are

- Required items and punctuation
- Optional items
- Loops

A description of each item follows.

Required Items and Punctuation

Required items and punctuation must be entered in the command or statement; you cannot omit them. A required item appears by itself in a path (horizontal line). A required item can be either a constant or a variable. For example, if a railroad diagram indicates

```
-- PAINT -- BEDROOM --<color>--|
```

the words PAINT and BEDROOM are required constants, and <color> is a required variable. You could correctly enter

```
PAINT BEDROOM BLUE
```

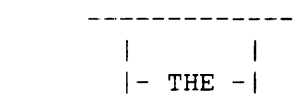
but not

```
PAINT BEDROOM
```

because the required item <color> would be missing.

Optional Items

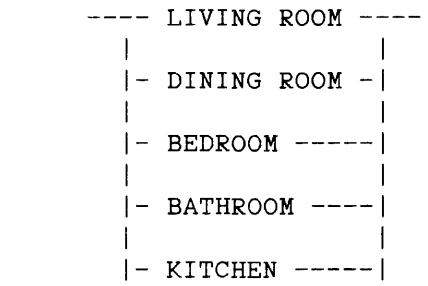
Optional items appear one below another in a vertical list. You can choose any one of the items in the list. If the list also contains an empty path (all dashes), you can omit the item entirely. An optional item can be either a variable or a constant. The PAINT command in the example contains two lists. The first is



which gives you two options:

- Enter the constant THE
- Omit it (because there is an empty path)

The second list has five optional constants:

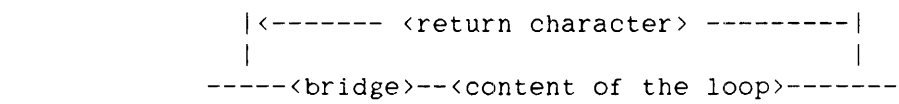


You must enter one of the optional items (LIVING ROOM, DINING ROOM, BEDROOM, BATHROOM, or KITCHEN) because there is no empty path in this list.

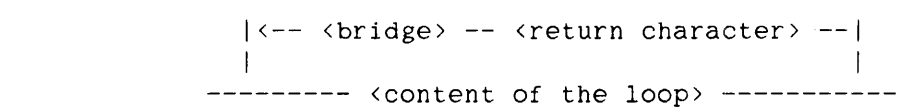
Loops

A loop is an item or group of items that you can repeat. The number of repetitions allowed is controlled by an item called the bridge.

A loop can span all or part of a railroad diagram. It always consists of at least two horizontal lines, one below the other, like this:



or



The bridge shows the maximum number of times you can go through the loop. The bridge can precede the contents of the loop, or it can precede the return character on the upper line of the loop to specify the number of times the right-to-left path can be traversed. The bridge is an integer enclosed in sloping lines, / \, for example, /4\. Not all loops have bridges. Those that do not can be repeated any number of times.

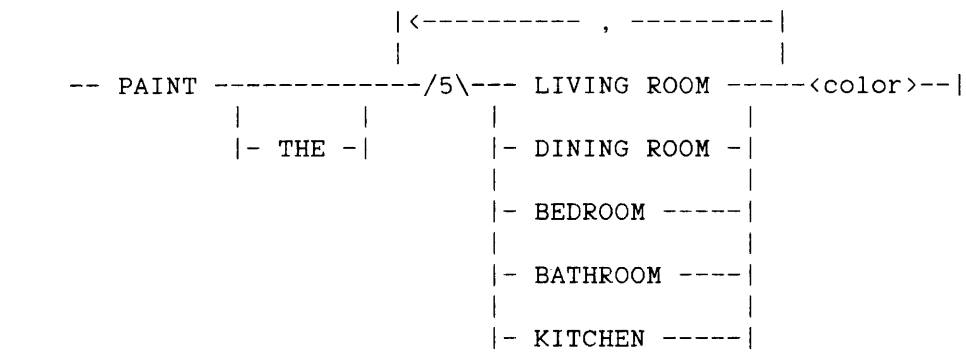
The top line is a right-to-left path that contains information about repeating the loop. The return character is the character to use before each repetition of the loop (often, a comma). Not all loops contain a return character; if none is shown, just enter one or more spaces before repeating the loop.

The other lines show the content of the loop (the data you enter each time you go through the loop). This can be any combination of optional items, required items, lists, and even other loops. The content of a loop can range from simple (one item), to very complex (many items, lists, and loops).

Example 1. A Simple Loop

The PAINT command as first shown is of limited usefulness. To tell the painter to do several rooms, you need a separate command for each room. It would be much easier if you could tell him to do several rooms in one command.

You can do that by making the list of rooms into a loop. The command would then look like this:



The bridge has a value of 5, so you can go through the loop up to five times, for a total of five rooms. The return character is a comma, which you must enter before repeating the loop content.

You can now enter

```
PAINT THE LIVING ROOM, BEDROOM, KITCHEN YELLOW
```

or

```
PAINT DINING ROOM, BEDROOM, BATHROOM BLUE
```

or

```
PAINT BEDROOM PINK
```

or

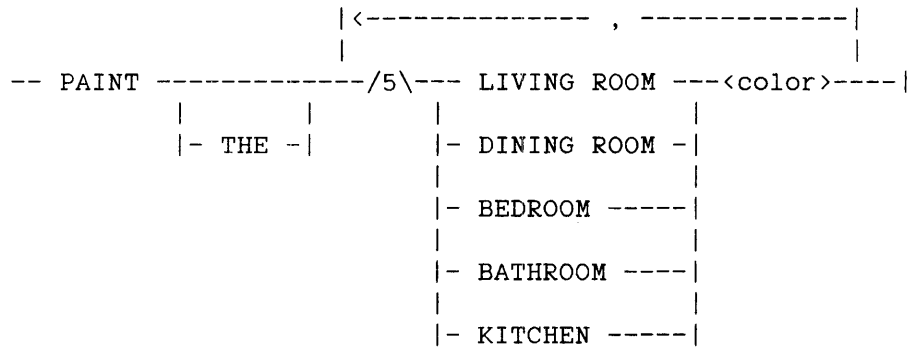
```
PAINT BEDROOM, BATHROOM, BEDROOM, BEDROOM BLUE
```

or any other valid combination.

This simple loop makes the PAINT command more versatile, but a significant drawback remains. Although you can include up to five rooms in a command, you cannot specify different colors.

Example 2. A More Complex Loop

If the content of the loop were to include the color, you could specify a different color for each room.



The content of the loop now consists of the

- List of optional constants that indicate rooms
- Required variable <color>

The bridge value is 5, and the return character is a comma. Given this railroad diagram, some valid PAINT commands would be

PAINT THE BEDROOM PINK

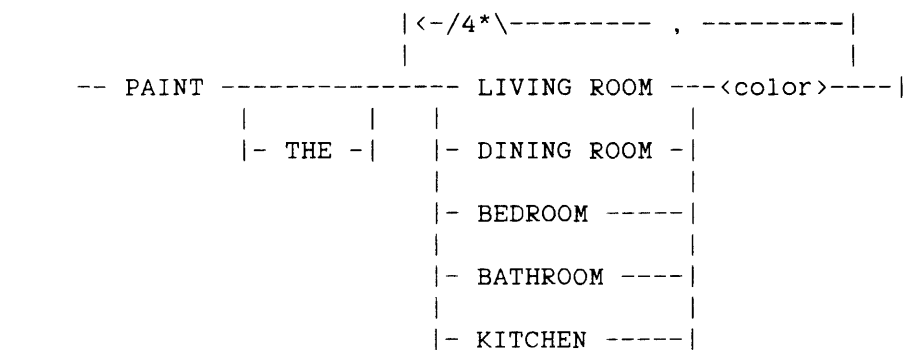
PAINT THE LIVING ROOM BLUE, DINING ROOM GREEN, KITCHEN YELLOW

PAINT BEDROOM GREEN, KITCHEN BLUE

and so on.

Example 3. Another Loop

In some bridges an asterisk follows the number. For example,



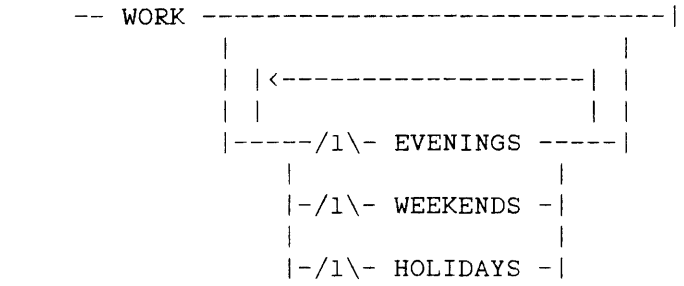
The asterisk means you must take the right-to-left path at least once. You cannot, for example, enter PAINT BEDROOM BLUE; you must tell the painter at least two rooms to paint. The maximum number of rooms to be painted is still five: the first time through the loop with up to four repetitions.

A valid form of the command would be

```
PAINT BEDROOM BLUE, KITCHEN YELLOW
```

Example 4. Another Use of the Bridge

A bridge can also control the number of times you take an individual path within a loop. For example, another command to the painter might be:



Each bridge /1\ indicates you can take that path once or not at all. That is, you can enter each of the items EVENINGS, WEEKENDS, and HOLIDAYS once at most. Some valid commands are

```
WORK EVENINGS WEEKENDS HOLIDAYS

WORK WEEKENDS

WORK HOLIDAYS EVENINGS
```

but

```
WORK EVENINGS EVENINGS
```

is invalid.

A FINAL WORD

To familiarize you with railroad diagrams, this explanation describes the elements of the diagrams and gives a few simplified examples. Some of the actual diagrams you will encounter in a book may be considerably more complex.

However, the principles are the same no matter how complex the diagram. The more you work with railroad notation, the more easily you will understand even the most complex diagrams.