27
3/13/81
240
T.S.
T.S.

(2)

UCC-ND

**NUCLEAR
DIVISION**

**UNION
CARBIDE**

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

P-2701

ORNL/CSD/TM-136

MASTER

ELMO Bumpy Torus Data Base

J. S. Stanton

REPRODUCTION OF THIS DOCUMENT IS UNLIMITED

TABLE OF CONTENTS

ACKNOWLEDGMENT	v
ABSTRACT	1
1. INTRODUCTION TO THE ELMO BUMPY TORUS DATA BASE	3
1.1. OVERVIEW	3
1.2. DEFINING AND ACCESSING THE INFORMATION IN THE DATA BASE	3
1.3. DATA BASE INDEX	4
1.4. THE OFF-LINE DATA ARCHIVE	5
2. DEFINING AND ACCESSING THE INFORMATION IN THE EBT DATA BASE	7
2.1. INTRODUCTION	7
2.2. DATA FILE STRUCTURE	7
2.2.1. Header Section	7
2.2.1.1. Format of the Header Section	8
2.2.2. Data Section	9
2.2.2.1. Data Attributes	10
2.3. BUILDING AND EDITING DESCRIPTOR FILES	13
2.3.1. Introduction	13
2.3.2. How to Build a Descriptor File	13
2.3.3. Example of Building a Descriptor File	13
2.3.3.1. Sample File Description	14
2.3.3.2. Sample File Format	14
2.3.3.3. Sample Execution of The BLDDFN Program	14
2.3.3.4. Sample Execution of the CNVDFN Command	20
2.3.3.5. Documentation of a Descriptor File as Produced by the System	21
2.4. ACCESSING DATA IN THE EBT DATA BASE	22
2.4.1. Introduction	22
2.4.2. Description of the Data Unpacking Routines	22
2.4.2.1. NEWFIL - Setup for a New Data File	22
2.4.2.1.1. NEWFIL Calling Specifications	23
2.4.2.2. GETDAT - Get Data Points	25
2.4.2.2.1. GETDAT Calling Specifications	25
2.4.3. Using the Data Unpacking Routines	26
2.4.3.1. Skeleton Program	26
2.4.3.2. Changes to User Programs	26
2.4.3.3. Loading the Data Unpacking Routines	27
2.4.4. Sample Program That Uses the Data Unpacking Routines	27
2.4.4.1. Program Listing	27
2.4.4.2. Execution of the Sample Program	29

3.	EBT DATA BASE INDEX	31
3.1.	INTRODUCTION	31
3.1.1.	EBT Data Base Index Structure	32
3.1.2.	System 1022 Introduction	34
3.1.3.	System 1022 Feature Overview	34
3.1.4.	System 1022 Data Base Structure	35
3.1.5.	EBT Index Data Set Formats	36
3.1.5.1.	Fields in the MCH Data Set	36
3.1.5.2.	Fields in the ACTD Data Set	39
3.1.5.3.	Fields in the DVAL Data Set	40
3.2.	USING THE SYSTEM	40
3.2.1.	Entering and Leaving System 1022	40
3.2.2.	System 1022 Command Structure	40
3.2.3.	Summary of System 1022 Commands	41
3.2.4.	System 1022 Examples	42
3.2.4.1.	System 1022 Example 1	42
3.2.4.2.	System 1022 Example 2	43
3.2.4.3.	System 1022 Example 3	44
3.2.5.	Descriptions of System 1022 Commands	45
3.2.5.1.	@OEBT	45
3.2.5.2.	FIND Command	45
3.2.5.2.1.	Basic Selection Conditions	46
3.2.5.2.2.	Logical Operators	47
3.2.5.2.3.	Special Selection Conditions	48
3.2.5.2.4.	Collapsing the Find Command	50
3.2.5.3.	SEARCH Command	52
3.2.5.4.	@DMPACT Command	53
3.2.5.5.	@DMPMCH Command	53
3.2.5.6.	PRINT Command	54
3.2.5.6.1.	Output List	55
3.2.5.6.2.	The ALL Option	57
3.2.5.6.3.	Totals List	57
3.2.5.6.4.	Output To Disk Files	59
3.2.5.7.	TYPE Command	60
3.2.5.8.	@MCHPRT Command	60
3.2.5.9.	DBSET Command	61
3.2.5.10.	MAP Command	62
3.2.5.11.	VALUES Command	64
3.2.5.12.	INFORM Command	65
4.	OFF-LINE DATA ARCHIVE	67
4.1.	INTRODUCTION	67
4.2.	END-USERS INTERFACE	68
4.3.	MAINTENANCE INTERFACE	71
4.3.1.	COPY TO OFF-LINE FUNCTION	72
4.3.2.	RESTORE TO ON-LINE FUNCTION	73
	APPENDIX A	75

ACKNOWLEDGMENT

This document was generated as a result of a project sponsored by the Computer Support Group of the Fusion Energy Division.

"Portions of the Data Base Index chapter of this document have been reprinted from the System 1022 (TM) User's Reference Manual with the permission of Software House. Use or disclosure of such material is subject to restrictions set forth in a license agreement relating to System 1022 (TM). System 1022 (TM) is a proprietary product of Software House, Cambridge, Massachusetts."

ABSTRACT

This document describes a set of computer programs developed to facilitate storage and retrieval of data generated by the ELMO Bumpy Torus (EBT) experiment.

The data is stored in a collection of files which contain either raw or analyzed data from diagnostics connected to the experiment.

An on-line index of steady-state machine conditions, diagnostic or analysis status information, and raw or analyzed data values unifies the file collection into a data base. The index is implemented under the System 1022 data base management system.

Access to the data is through either the standalone user language of System 1022 or a series of routines which allow a FORTRAN program to process the contents of an EBT file. Both access methods insulate the end user from the actual storage format of the data. The user refers to data by name and the system handles the problems associated with finding and unpacking the needed information.

The data processed by the EBT data base is generated by a series of DEC PDP8 and PDP11 computers. The data base programs reside on the Fusion Energy Division's DECSYSTEM-10 computer.

1. INTRODUCTION TO THE ELMO BUMPY TORUS DATA BASE

1.1. OVERVIEW

The ELMO BUMPY TORUS (EBT) data base is a collection of files which contain either raw or analyzed data from diagnostics connected to the EBT experiment. An on-line index of steady-state machine conditions, diagnostic or analysis status information, and raw or analyzed data values unifies the file collection into a data base. The data base automatically spools old files to and from off-line disk packs thus reducing the need for on-line storage.

1.2. DEFINING AND ACCESSING THE INFORMATION IN THE DATA BASE

The source and format of each EBT data file is documented in a set of descriptor files. One descriptor file may define the format of many data files. The BLDDFN (build definition) program generates descriptor files by interacting with the developer of a data file. The descriptor files are used by a set of data unpacking routines to transfer information from the data base to analysis programs. All access to the information in the data files is through a set of standard data unpacking routines. In order to process a data file, a program must execute a call to the NEWFIL (new file) routine followed by one or more calls to the GETDAT (get data) routine. NEWFIL opens a data file and processes its associated descriptor. GETDAT uses the format information set up by NEWFIL to locate and return data points from the file. The use of this technique to access data tends to shield analysis programs from changes in the manner in which data from a diagnostic is recorded. A further explanation of defining and accessing data is available in Chapter 2.

1.3. DATA BASE INDEX

The EBT experiment is typically run in a steady state while one or more diagnostic experiments are performed to either acquire information about or change the nature of the plasma in the torus. This mode of operation tends to establish a natural hierarchy of information in which the data that defines the steady-state condition of the EBT machine is the highest level and the information generated by diagnostics or analyses is the lowest level. The data acquired from the EBT experiment is stored in a collection of files in which each file contains either steady-state machine conditions, diagnostic data or analyzed information. The on-line index is the mechanism by which the collection of files is unified into a data base. The index can be viewed as simply a list of the files in the data base. The index reflects the natural structure of the EBT data by maintaining a hierarchy of indices. The highest index level holds the names of the files which contain machine condition information. This level also includes data which defines the nature of the machine condition described in the file, such as: 18 GHZ power level, generator currents, pressure, background radiation, etc. The next lower index level contains the names of the diagnostic and analyzed data files associated with a machine condition file. This level is used to determine which diagnostics or analyses were performed under a given machine condition file. The lowest index level holds selected data values from diagnostic or analyzed files.

The data base index may be used as a file selection or a data analysis tool. When used as a file selection tool, the main function of the data base index is to generate a list of file names. The list

may be based upon the time the file was generated, the state of the machine, the value of some data item(s) from the file, or any combination thereof. The list may then be used by an analysis program to select files for processing. As a data analysis tool the index is most useful for Boolean algebra selection and comparison of EBT runs. As an example it is possible to select all runs of the EBT machine where 18 GHz power was greater than 50 kw and pressure was less than $0.1 \cdot 10^{-6}$ atm and ICRH frequency was equal to 30 mhz, then calculate the average of WPERP and compare it to the value for some other set of EBT runs. The index is implemented as a group of System 1022 data sets (System 1022 is a general-purpose data base management system). The index and System 1022 are described in Chapter 3.

1.4. THE OFF-LINE DATA ARCHIVE

Since the amount of data generated by the EBT diagnostic and analysis programs is too voluminous to be maintained on-line, an archiving system is included in the data base. At the request of the data base coordinator, the archive system spools data files to off-line storage media. The location of each archived file is maintained in the on-line index. Using this information, the archiving system will return needed data files to an on-line disk pack. The archiving function is documented in Chapter 4.

6

Blank

2. DEFINING AND ACCESSING THE INFORMATION IN THE EBT DATA BASE

2.1. INTRODUCTION

The EBT data base is a loosely coupled collection of data files. The coupling is performed by the data base index. The files may be divided into three classes based upon the type of data in the file: machine condition, diagnostic, or analyzed. The data files consist of two parts: a header section and a data section. The header section contains file description information and the data section holds the actual experimental or analyzed data. The format of each file in the EBT data base is defined in a descriptor file. One descriptor file may be used to define many data files. The BLDDFN (build definition) program creates the descriptor files by interacting with the developer of a data file. All access to the data files is through a set of data unpacking routines. The unpacking routines use the descriptor file pointed to by the data file header section to locate and return requested information.

2.2. DATA FILE STRUCTURE

2.2.1. Header Section

The header section is used to identify the source, creation time, and format of a data file. The header points to the machine conditions in effect when the file was created. It also contains time and date stamps for the period during which the file was created. In addition the header points to the descriptor file which defines the format of the data section. The header section is encoded in seven-bit ASCII and contains 13 fields spanning 65 seven-bit bytes on the DECsystem-10.

2.2.1.1. Format of the Header Section

<u>BYTE #</u>	<u>FIELD NAME</u>	<u>DESCRIPTION</u>
1-2	HEADER FORMAT	TYPE OF HEADER SECTION THAT FOLLOWS. "00" FOR THIS FORMAT.
3-5	DIAGNOSTIC ID	NAME OF THE SOURCE OF THE DATA FILE.
6-10	DIAGNOSTIC SEQUENCE	UNIQUE IDENTIFIER FOR EACH RUN OF A DIAGNOSTIC.
11-13	DESCRIPTOR KEY	IDENTIFIES THE DESCRIPTION FILE NEEDED TO UNPACK THIS DATA FILE.
14-15	FILLER	NOT USED.
16-20	MACHINE CONDITION SEQUENCE AT START	IDENTIFIES THE MACHINE CONDITION FILE THAT WAS IN EFFECT WHEN THE DATA FILE WAS STARTED.
21-31	START DATE	THE DATE THE DATA FILE WAS BEGUN. dd-mm-yyyy
32	FILLER	NOT USED.
33-40	START TIME	THE TIME THE DATA FILE WAS BEGUN. hh:mm:ss
41-45	MACHINE CONDITION SEQUENCE AT END	IDENTIFIES THE MACHINE CONDITION FILE THAT WAS IN EFFECT WHEN THE DATA FILE WAS COMPLETED.
46-56	ENDING DATE	THE DATE THE DATA FILE WAS COMPLETED.
57	FILLER	NOT USED.
58-65	ENDING TIME	THE TIME THE DATA FILE WAS COMPLETED.

2.2.2. Data Section

The data section of a file is viewed as a collection of data points from one or more sources. A data source can be an A/D channel, an entry from a terminal, a calibration value, the result of an analysis, a switch setting, etc.. Each data source contained in a data file must be assigned a one- to five-letter name. This name is called the source id and is used for all references to the data from that source. Using names to identify the data from each source permits the user to change the way in which a data file is recorded without having to modify the programs which read that data file. Each data source may generate one or more data points. The number of points may be fixed when the description of the data file is created or it may be established when the data file is generated. In addition a data source may record data in either integer, floating point, double precision floating point, or ASCII. Data sources which develop data points in parallel may be related to each other by defining a repeating group structure. The group structure allows data points for related sources to be interleaved in the data file. This recording technique is especially useful where data sources are creating large numbers of data points in parallel and insufficient local storage is available to accumulate all of the data points.

As an example: if data sources SRCA and SRCB are defined as members of the repeating group GRP1, then the data generated by SRCA and SRCB could be recorded in the following manner:

<u>WORD #</u>	<u>DATA</u>
X	POINT 1 FROM SRCA
X+1	POINT 1 FROM SRCB
X+2	POINT 2 FROM SRC.
X+3	POINT 2 FROM SRCB
	.
	.
X+2N-2	POINT N FROM SRCA
X+2N-1	POINT N FROM SRCB

2.2.2.1. Data Attributes

Each data source may be assigned a number of attributes, including: source id, source or group flag, data type, cluster size, number of significant digits, constant multiplier, indexing mode, and comments. The source id, source or group flag, data type, and cluster size are required attributes; the others are optional. The attributes of each source in a data file are contained in a descriptor file. The BLDDFN (build definition) program is used to create the descriptor files. BLDDFN will prompt the developer of a data file through the specification of the attributes. The data sources should be described to BLDDFN in the same order as they are recorded in the data file. It is often useful to consider the use of a data file in the data base before its format is set.

A description of each attribute and its BLDDFN prompt follows. The BLDDFN prompt is underlined and precedes the description of the attribute.

SOURCE ID: Each generator of data points in a file must be assigned a name. The name is a one- to five-character code which uniquely identifies the data source. The name need only be unique within the

current descriptor file. The name assigned is called the source id and is used for all references to the data points created by the source.

CLUSTER FACTOR: Defines the number of data points that are recorded in the data file for each occurrence of a data source. If a data source is a member of a repeating group, it may occur more than once in the file. The cluster factor may be a fixed number such as 1 or 23, it may be defined in the data file, or it may be specified as repeating until the end-of-file. If the cluster factor is defined in the data file, each cluster of data values must be preceded by a one-word integer specifying the number of repetitions. For ASCII values a cluster factor of 1 means five characters.

<u>VALUE</u>	<u>MEANING</u>
n	WHERE n IS A POSITIVE INTEGER. SPECIFIES A FIXED NUMBER OF REPETITIONS.
0	ZERO. DEFINED IN THE DATA FILE.
-1	REPEATS TO END-OF-FILE.

SOURCE OR GROUP. (S/G): Each source id entered must be declared to be either a data source or a group. To select the class of an item, respond to the prompt with either "S" or "G".

DATA TYPE: This attribute defines the recording mode of the source. The recognized modes are: integer, floating point, double precision floating point, and ASCII. ASCII data values must be recorded in multiples of five characters. A mode is selected by responding to the prompt with one of the following codes:

<u>CODE</u>	<u>RESULTING DATA TYPE</u>
1	INTEGER
2	FLOATING POINT
3	DOUBLE PRECISION FLOATING POINT
4	ASCII

SIGNIFICANT DIGITS: Specifies the number of valid digits of data in each point generated by the data source. This attribute is not used by the data unpacking routines but is made available to user programs in a common area named SIGNIF.

INDEXING MODE: The EBT data base indexing routines will place values from selected data sources in an on-line index. The index may be used for analysis or as a file selection tool. The indexing mode specifies which data value from a source will be placed in the index.

<u>VALUE</u>	<u>INDEX MODE</u>
0	NONE
1	FIRST VALUE
2	AVERAGE VALUE
3	ALL VALUES

CONSTANT MULTIPLIER: The data unpacking routines of the EBT data base will optionally multiply each value obtained from a data source by a constant. The constant may be any floating point value. This feature is useful in those cases where the units of a data source have varied over time. By using the constant multiplier, all of the values from a source may be retrieved in the same units. A new descriptor file should be developed each time the units of a data source change.

COMMENTS: A comment may be associated with each data source. Up to 80 characters may be entered. End the comment with a (RETURN).

2.3. BUILDING AND EDITING DESCRIPTOR FILES

2.3.1. Introduction

Descriptor files are used by the data unpacking routines of the EBT data base to locate and return data points from files. The program named BLDDFN (build definition) creates descriptor files. System 1022 may be used to edit existing descriptors. All of the descriptor files used in the EBT data base are stored in account 210,2742.

2.3.2. How to Build a Descriptor File

To create a descriptor file, the following steps are necessary:

1. Identify the sources of the data recorded in the file,
2. Determine the order in which the data from the sources is recorded,
3. Ascertain the attributes of each data source such as: source id, data type, indexing mode, etc.,
4. Execute the BLDDFN program entering the data sources and their attributes in the same order as the sources are recorded in the data file,
5. Use the CNVDFN (convert definition) command to convert the description to the format used by the data unpacking routines.

2.3.3. Example of Building a Descriptor File

The following example shows the creation of a descriptor file using the BLDDFN program and the CNVDFN command.

2.3.3.1. Sample File Description

As an example consider a file created by a diagnostic named TST consisting of data from four sources.

1. S1 - a calibration value recorded as an integer.
2. S2 - a set of 100 data points from an A/D channel, recorded as floating point numbers.
3. S3 - a set of "N" data points from a source, recorded as integers, interleaved with data from source S4.
4. S4 - a set of "N" data points, recorded as double precision floating point numbers, intermixed with points from S3.

Where "N" is defined as the number of repetitions found before the end-of-file.

2.3.3.2. Sample File Format

<u>WORD #'S</u>	<u>SOURCE</u>	<u>TYPE</u>	<u>SAMPLE VALUE</u>
1-13	HEADER		
14	S1	INTEGER	112
15	S2	REAL	2.01
16	S2	REAL	2.02
.	.		
.	.		
114	S2	REAL	3.00
115	S3	INTEGER	1
116-117	S4	DBL REAL	1.0
118	S3	INTEGER	2
119-120	S4	DBL REAL	2.0
121	S3	INTEGER	3
122-123	S4	DBL REAL	3.0
124	END-OF-FILE		

2.3.3.3. Sample Execution of the BLDDFN Program

In this example the user's input is underlined.

.RUN BLDDFN

VERBOSE OR CONCISE PROMPTS? (V/C):V

INPUT THE THREE-LETTER CODE THAT IDENTIFIES THE DIAGNOSTIC, ANALYSIS, OR MACHINE CONDITION FILE THAT THIS SESSION DESCRIBES.

DIAGNOSTIC ID:TST

; the diagnostic id is assigned by R. D. Burris

THIS DESCRIPTION HAS BEEN ASSIGNED NAME:TST002

; the description name is used as the file-name of the descriptor file.

ENTER UP TO 80 CHARACTERS OF GENERAL COMMENTS:DEFN OF THE FORMAT OF THE DATA GENERATED BY THE TST DIAGNOSTIC.

AN ENTRY OF (CNTRL)Z DURING THE DESC. OF AN ITEM WILL CANCEL THAT DESC. AND START A NEW ITEM.

; (cntrl)Z means to enter the letter Z while depressing the cntrl key.

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:S1

INPUT	RESULTING CLUSTER FACTOR
-1	ITEM REPEATS UNTIL E-O-F,
0	REPEAT COUNT IS STORED IN THE DATA FILE,
n	ITEM REPEATS "n" TIMES, (WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY DEFINES THE NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR:1

SPECIFY IF THIS ITEM IS A DATA SOURCE OR A GROUP SOURCE OR GROUP.
(S/G):S

ENTRY	RESULTING DATA TYPE
1	INTEGER,
2	REAL,
3	DOUBLE REAL,
4	ASCII.

DATA TYPE:1

SIGNIFICANT DIGITS:5

VALUE RESULTING INDEXING MODE

0 NO VALUES INDEXED,
 1 INDEX FIRST VALUE,
 2 INDEX THE AVERAGE VALUE,
 3 INDEX ALL VALUES.

INDEXING MODE:0

CONSTANT MULTIPLIER:1.0.

INPUT UP TO 80 CHARACTERS OF DATA ITEM SPECIFIC
 COMMENTS:A CALIBRATION VALUE.

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP
 TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:S1

SOURCE ID S1 IS ALREADY IN USE.

; source ids must be unique within a descriptor file

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP
 TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:S2

INPUT RESULTING CLUSTER FACTOR

-1 ITEM REPEATS UNTIL E-O-F,
 0 REPEAT COUNT IS STORED IN THE DATA FILE,
 n ITEM REPEATS "n" TIMES,
 (WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY
 DEFINES THE NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR:100

SPECIFY IF THIS ITEM IS A DATA SOURCE OR A GROUP
 SOURCE OR GROUP. (S/G):G

COMMENTS:(CNTRL)Z

; the above is an example of aborting an item description

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED.
(UP TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:S2

INPUT RESULTING CLUSTER FACTOR

-1 ITEM REPEATS UNTIL E-O-F,
0 REPEAT COUNT IS STORED IN THE DATA FILE,
n ITEM REPEATS "n" TIMES,
(WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY DEFINES THE
NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR:100

SPECIFY IF THIS ITEM IS A SOURCE OR GROUP. (S/G):S

ENTRY RESULTING DATA TYPE

1 INTEGER,
2 REAL,
3 DOUBLE REAL,
4 ASCII.

DATA TYPE:2

SIGNIFICANT DIGITS:6

VALUE RESULTING INDEXING MODE

0 NO VALUES INDEXED,
1 INDEX FIRST VALUE,
2 INDEX THE AVERAGE VALUE,
3 INDEX ALL VALUES.

INDEXING MODE:2

CONSTANT MULTIPLIER:2.0E10

INPUT UP TO 80 CHARACTERS OF DATA ITEM SPECIFIC
COMMENTS:A/D OUTPUT CHANNEL

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP
TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:GRP34

INPUT RESULTING CLUSTER FACTOR

-1 ITEM REPEATS UNTIL E-O-F,
0 REPEAT COUNT IS STORED IN THE DATA FILE,
n ITEM REPEATS "n" TIMES,
 (WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY DEFINES THE
NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR: -1

SPECIFY IF THIS ITEM IS A DATA SOURCE OR A GROUP SOURCE OR GROUP.
(S/G): G

COMMENTS: GROUP DEFN TO ALLOW S3 & S4 TO INTERLEAVE

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP
TO FIVE CHARACTERS, BLANK TO STOP). SOURCE ID: S3

IS GROUP GRP34 THE OWNER OF THIS ITEM? (Y/N): Y

INPUT RESULTING CLUSTER FACTOR

-1 ITEM REPEATS UNTIL E-O-F,
0 REPEAT COUNT IS STORED IN THE DATA FILE,
n ITEM REPEATS "n" TIMES,
 (WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY DEFINES THE
NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR: 1

SPECIFY IF THIS ITEM IS A SOURCE OR GROUP. (S/G): S

ENTRY RESULTING DATA TYPE

1 INTEGER,
2 REAL,
3 DOUBLE REAL,
4 ASCII.

DATA TYPE: 1

SIGNIFICANT DIGITS: 10

VALUE RESULTING INDEXING MODE

0 NO VALUES INDEXED,
 1 INDEX FIRST VALUE,
 2 INDEX THE AVERAGE VALUE,
 3 INDEX ALL VALUES.

INDEXING MODE: 0

CONSTANT MULTIPLIER: 1.0

INPUT UP TO 80 CHARACTERS OF DATA ITEM SPECIFIC
 COMMENTS: A/D CHANNEL

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID: S4

IS GROUP GRP34 THE OWNER OF THIS ITEM? (Y/N): Y

INPUT RESULTING CLUSTER FACTOR

-1 ITEM REPEATS UNTIL E-O-F,
 0 REPEAT COUNT IS STORED IN THE DATA FILE,
 n ITEM REPEATS "n" TIMES,
 (WHERE n IS A POSITIVE INTEGER)

IF THE ITEM IS A MEMBER OF A GROUP, THEN THIS ENTRY DEFINES THE NUMBER OF REPEATS IN EACH OCCURRENCE OF THE GROUP.

CLUSTER FACTOR: 1

SPECIFY IF THIS ITEM IS A SOURCE OR GROUP. (S/G): S

ENTRY RESULTING DATA TYPE

1 INTEGER,
 2 REAL,
 3 DOUBLE REAL,
 4 ASCII.

DATA TYPE: 3

SIGNIFICANT DIGITS: 12

VALUE RESULTING INDEXING MODE

0 NO VALUES INDEXED,
 1 INDEX FIRST VALUE,

2 INDEX THE AVERAGE VALUE,
3 INDEX ALL VALUES.

INDEXING MODE:Q

CONSTANT MULTIPLIER:3.0

INPUT UP TO 80 CHARACTERS OF DATA ITEM SPECIFIC
COMMENTS:FLOATING POINT DATA ITEM.

STARTING A NEW ITEM DESCRIPTION.

INPUT THE NAME OF THE DATA SOURCE OR GROUP TO BE DESCRIBED. (UP
TO FIVE CHARACTERS, BLANK TO STOP).

SOURCE ID:

THIS DESCRIPTOR HAS BEEN NAMED TST002 5:TOTAL ITEMS DEFINED 4:DATA
SOURCES 1:GROUPS

INCLUDE A PARAMETER STATEMENT WITH VALUES GREATER THAN OR EQUAL TO THE
ONES ON THE FOLLOWING CARD IN ANY PROGRAM THAT USES THE DATA UNPACKING
ROUTINES TO PROCESS A FILE WHOSE FORMAT IS DEFINED BY THIS DESCRIPTION.

PARAMETER ARSIZ= 5,HARSIZ= 2,GRPSIZ= 1,NUMBUF=1

TO FINISH THE DESCRIPTION FILE, ENTER THE FOLLOWING COMMAND AFTER THIS
PROGRAM EXITS:

DO CNVDPN TST,002

STOP

END OF EXECUTION

2.3.3.4. Sample Execution of the CNVDPN Command

The following example shows how to complete the creation of an EBT
data base descriptor file. The CNVDPN command is entered when the
BLDDPN program exits. A comma must be entered between the diagnostic
id and descriptor key fields. The user's input is underlined.

.DO CNVDFN TST.002

.COPY DSKB:TST002.DFN[210,2742]<117>=DSK:TST002.DAT

.Q TST002.LPT

TOTAL OF 3 BLOCKS IN 1 FILE IN LPT REQUEST

2.3.3.5. Documentation of a Descriptor File as Produced by the System

The CNVDFN command automatically generates a printer listing that documents each descriptor file. If additional copies of the documentation are needed, a command named PRIDFN may be used. To execute the PRIDFN command, enter:

DO PRIDFN xxx,yyy

where xxx equals the diagnostic id and yyy is the descriptor key. An example of such a listing follows.

THIS DESCRIPTOR HAS BEEN NAMED TST002
 5:TOTAL ITEMS DEFINED
 4:DATA SOURCES
 1:GROUPS

INCLUDE A PARAMETER STATEMENT WITH VALUES GREATER THAN OR EQUAL TO THE ONES ON THE FOLLOWING IN ANY PROGRAM THAT USES THE DATA UNPACKING ROUTINES TO PROCESS A FILE WHOSE FORMAT IS DEFINED BY THIS DESCRIPTION.

PARAMETER ARSIZ= 5,HARSIZ= 2,GRPSIZ= 1,NUMBUF=1

TST 002 JUN-20-1980				DEFN OF THE FORMAT OF THE DATA GENERATED BY THE TST DIAGNOSTIC.						
OWNER	SEQ	SRCID	S/G	CLUSTER	TYPE	SIGNF	INDEX	MULT.	COMMENTS	
	00001	S1	0	1	1	5	0	.10000E+01	A CALIBRATION VALUE.	
	00003	S2	0	100	2	6	2	.20000E+11	A/D OUTPUT CHANNEL	
	00004	GRP34	1	-1	0	0	0	.00000E+00	GROUP DEFN TO ALLOW S3 & S4 TO INTERLEAVE	
GRP34	00001	S3	0	1	1	10	0	.10E+01	A/D CHANNEL	
GRP34	00002	S4	0	1	3	12	0	.30000E+01	FLOATING POINT DATA ITEM.	

2.4. ACCESSING DATA IN THE EBT DATA BASE

2.4.1. Introduction

All access to information contained in the files of the EBT data base is through a set of data unpacking routines. The unpacking routines use the descriptor file pointed to by the header of each data file to locate and unpack requested data points. A user of the EBT data base need be concerned with only two data unpacking routines: NEWFIL, and GETDAT. NEWFIL accepts a data file name as input, processes the header of the file, and then interprets the descriptor file pointed to by the header. NEWFIL must be called once for each new file the data unpacking routines are to process. GETDAT accepts as input the source id of the data to be unpacked and returns an array of data points from that source. Both routines are FORTRAN callable and use I/O units 61, 62, and 63 to access the data and descriptor files.

2.4.2. Description of the Data Unpacking Routines

2.4.2.1. NEWFIL - Setup for a New Data File

The NEWFIL routine is called once for each data file to be processed. The arguments to NEWFIL are the name of the data file, the machine condition sequence during which the file was created, and a status flag. NEWFIL opens the data file and inputs the header information. The header is stored in the named common DDHDR and may be referenced by the calling program. Using the descriptor key in the header NEWFIL builds the name of the descriptor file needed to unpack the data file. NEWFIL then processes the descriptor file setting up

the data unpacking arrays used by GETDAT. The FORTRAN definition of the data unpacking arrays is contained in the source file EBTCOM.FOR which must be included in any program that uses the data unpacking routines. A listing of EBTCOM.FOR appears in Appendix A. NEWFIL also places a copy of the machine conditions in effect when the file was created in the common area named VALMC. The definition of VALMC is contained in EBTCOM.FOR.

If a binary zero is specified as the name of the data file to be processed, NEWFIL will read a file called DMPACT.DAT to find the name(s) of the data files. DMPACT.DAT should be an ASCII file recorded in the format used by the @DMPACT command of the data base index. The name of the data file being processed is returned in the file-name argument. Additional calls with a zero file-name cause NEWFIL to process the next data file listed in the DMPACT.DAT file.

2.4.2.1.1. NEWFIL Calling Specifications.

CALL NEWFIL(file-name,mach-cond,status-flag)

where:

file-name is the name of the data file to process encoded as "filename.extension" in a double-precision variable. If the value of the file-name variable is a binary zero, then NEWFIL will read a file named DMPACT.DAT to find the file-name(s) to be processed. The name of the file(s) will be returned in file-name.

mach-cond the sequence number of the machine condition in effect when the file was created. If a binary zero is specified, NEWFIL will use the index to determine the machine condition and the value will be returned in this field. Do not use a constant for this argument.

status is an integer value returned by NEWFIL to indicate the success or failure of the call. The following codes are returned in status.

<u>STATUS</u>	<u>MEANING</u>
0	SUCCESS
1	OWNER OF AN ITEM NOT FOUND
2	DEFN. FILE OUT OF ORDER
3	ARSIZ PARAMETER IS TOO SMALL
4	DEFN. FILE IS EMPTY
5	DATA FILE IS EMPTY
6	HARSIZ PARAMETER IS TOO SMALL
7	DATA FILE NOT FOUND
8	DEFN. FILE NOT FOUND

2.4.2.2. GETDAT - Get Data Points

The GETDAT routine is called to access data points from a source. GETDAT uses the data unpacking arrays setup by NEWFIL to find and unpack a group of data points. Repeated calls to GETDAT for the same source result in the sequential reading of the data file for that source. It is legal to intermix requests for different sources; GETDAT will still correctly process each source. The index of the current data source in the data unpacking arrays is stored in a common variable named ICURR. The data unpacking arrays and ICURR are defined in the EBTOM.DAT file. If a negative word count is specified, GETDAT will return the data in clusters, one cluster per call.

2.4.2.2.1. GETDAT Calling Specifications.

CALL GETDAT(source-id,array-size,status,data-array)

where:

source-id	is the name of the data source to be processed.
array-size	on entry this argument should be set to the size of the data array in words; on return the argument contains the number of data items returned. Do not use a constant for this argument.
status	is an integer value returned by GETDAT that indicates the success or failure of the call.

<u>VALUE</u>	<u>STATUS</u>
0	SUCCESS
-1	END OF A SOURCE ENCOUNTERED
1	NO DATA FOUND FOR REQUESTED SOURCE
2	NO DATA FOUND FOR REQUESTED SOURCE
3	NO DATA FOUND FOR REQUESTED SOURCE
4	THE REQUESTED SOURCE DOES NOT EXIST
5	INVALID REQUEST
6	LESS THAN A FULL CLUSTER RETURNED

data-array is a single-dimensioned array in which
GETDAT is to return data.

2.4.3. Using the Data Unpacking Routines

2.4.3.1. Skeleton Program

C set the size of the data unpacking arrays

```
PARAMETER ARSIZ=x,HARSIZ=y,GRPSIZ=z,NUMBUF=a
```

C load the FORTRAN definitions of the unpacking arrays

```
INCLUDE 'DSKB:EBTCOM.FOR[210,2740]'/NOLIST
INCLUDE 'DSKB:EBTDAT.FOR[210,2740]'/NOLIST
```

C setup to process a new data file

```
CALL NEWFIL(file-name,mach-cond,status)
```

C get some data

```
CALL GETDAT(source-id,array-size,status,data-array)
```

```
STOP
END
```

2.4.3.2. Changes to User Programs

In order to use the data unpacking routines, the following changes must be made to a program.

- 1) Insert the following parameter statement at the beginning of the program.

```
PARAMETER ARSIZ=x,HARSIZ=y,GRPSIZ=z,NUMBUF=a
```

where:

x,y, and z

are greater than or equal to the values specified by the BLDDFN program or the CNVDFN command for the files to be processed.

- a is an integer equal to 1 if the data files will be processed randomly and 10 if sequentially. Use 1 if you do not know.

- 2) Insert the following line before any DATA statements, statement function definitions, or executable statements.

```
INCLUDE 'DSKB:EBTCOM.FOR[210,2740]'/NOLIST
```

- 3) Enter this line before any executable statements.

```
INCLUDE 'DSKB:EBTDAT.FOR[210,2740]'/NOLIST
```

- 4) Then insert the calls to NEWFIL, and GETDAT at the appropriate places in the program.

2.4.3.3. Loading the Data Unpacking Routines

To load the data unpacking routines with a program, enter the following command.

```
LOAD your-pgm-name,@DSKB:EBTUNP[210,2740]
```

2.4.4. Sample Program That Uses the Data Unpacking Routines

2.4.4.1. Program Listing

```
C
C A PROGRAM TO EXERCISE THE DATA UNPACKING ROUTINES
C
C PARAMETER STATEMENT THAT ALLOWS UP TO TEN SOURCE AND
C GROUP DEFINITIONS, TEN NESTED GROUP COUNTERS, TWO
C LEVELS OF GROUP NESTING, AND ONE DATA BUFFER
C
C   PARAMETER ARSIZ=10,HARSIZ=10,GRPSIZ=2,NUMBUF=1
C
C DEFINITION OF THE DATA FILE NAME VARIABLE
C   DOUBLE PRECISION DFILNM
C
C DEFINITION OF THE DATA ARRAYS
C   DOUBLE PRECISION DARRAY
C   DIMENSION IARRAY(50),ARRAY(50),DARRAY(25)
C   EQUIVALENCE (IARRAY(1),ARRAY(1),DARRAY(1))
C
C DEFINITION OF OTHER COMMUNICATION STUFF
C   INTEGER ICOUNT,ISTAT,ISRC,MCSEQ
C
C
C EBT DATA UNPACKING COMMONS
C
C   INCLUDE 'DSKB:EBTCOM.FOR[210,2740]'/NOLIST
```

```

      INCLUDE 'DSKB:EBTDAT.FOR[210,2740]'/NOLIST
C
C GET A FILE NAME
C
100 WRITE(5,110)
110 FORMAT(' ENTER NAME OF DATA FILE:',$,)
      READ(5,120,END=999)DFILNM
120 FORMAT(A10)
C
C GET A MACHINE CONDITION SEQUENCE
C
      WRITE(5,130)
130 FORMAT(' ENTER THE MACHINE CONDITION SEQUENCE:',$,)
      READ(5,140)MCSEQ
140 FORMAT(A5)
C
C PROCESS THE BEGINNING OF A NEW FILE
C
C
      CALL NEWFIL(DFILNM,MCSEQ,ISTAT)
C
C CHECK IF SUCCESSFUL
C
      IF(ISTAT.EQ.0)GOTO 200
C NOPE
      WRITE(5,150)ISTAT
150 FORMAT(' ERROR ON NEWFIL:',I5)
      GOTO 100
C
C ASK FOR A DATA SOURCE
C
200 WRITE(5,210)
210 FORMAT(' ENTER SOURCE:',$,)
      READ(5,220,END=100)ISRC
220 FORMAT(A5,I5)
C
C GET ICOUNT DATA POINTS FROM SOURCE
C
300 ICOUNT=50
      CALL GETDAT(ISRC,ICOUNT,ISTAT,IARRAY)
C
C CHECK IF SUCCESSFUL
C
      IF(ISTAT.LE.0)GOTO 320
C NOPE
      WRITE(5,310)ISTAT,ISRC
310 FORMAT(' ERROR:',I5,' UNPACKING SOURCE:',A5)
      GOTO 200
C
C TEST IF ANY DATA POINTS FOUND
C
320 IF(ICOUNT.GT.0)GOTO 350
C NOPE
      WRITE(5,330)ISRC
330 FORMAT(' NO DATA FOR SOURCE:',A5)

```

```

      GOTO 500
C
C OUTPUT DATA POINTS ON TERMINAL. USE FORMAT BASED
C ON DATA TYPE OF REQUESTED SOURCE.
C
350 GOTO(400,420,440,460)DATYPE(ICURR)
400 WRITE(5,410)(IARRAY(I),I=1,ICOUNT)
410 FORMAT(' ',7(I10,1X))
      GOTO 500
420 WRITE(5,430)(ARRAY(I),I=1,ICOUNT)
430 FORMAT(' ',7(E11.4,1X))
      GOTO 500
440 WRITE(5,450)(DARRAY(I),I=1,ICOUNT)
450 FORMAT(' ',7(E11.4,1X))
      GOTO 500
460 WRITE(5,470)(IARRAY(I),I=1,ICOUNT)
470 FORMAT(' ',14(A5))
      GOTO 500
C
C CHECK IF END OF A SOURCE FOUND
C
500 IF(ISTAT.EQ.0)GOTO 200
C YEP
      WRITE(5,510)ISRC
510 FORMAT(' END OF SOURCE:',A5,' ENCOUNTERED.')
      GOTO 200
999 STOP
      END

```

2.4.4.2. Execution of the Sample Program

This execution uses the file defined in the example on building definition files.

```

.EX EXAMPL.@DSKB:EBTUNP[210,2740]
LINK: Loading
[LNKXCT EXAMPL Execution]

```

```

ENTER NAME OF DATA FILE: B0001.TST
ENTER THE MACHINE CONDITION SEQUENCE: A0002
ENTER SOURCE: S4
  0.3000E+01 0.6000E+01 0.9000E+01
END OF SOURCE: S4      ENCOUNTERED.
ENTER SOURCE: S1
  112
END OF SOURCE: S1      ENCOUNTERED.
ENTER SOURCE: S2

```


0.4020E+11 0.4040E+11 0.4060E+11 0.4080E+11 0.4100E+11 0.4120E+11 0.4140E+11
 0.4160E+11 0.4180E+11 0.4200E+11 0.4220E+11 0.4240E+11 0.4260E+11 0.4280E+11
 0.4300E+11 0.4320E+11 0.4340E+11 0.4360E+11 0.4380E+11 0.4400E+11 0.4420E+11
 0.4440E+11 0.4460E+11 0.4480E+11 0.4500E+11 0.4520E+11 0.4540E+11 0.4560E+11
 0.4580E+11 0.4600E+11 0.4620E+11 0.4640E+11 0.4660E+11 0.4680E+11 0.4700E+11
 0.4720E+11 0.4740E+11 0.4760E+11 0.4780E+11 0.4800E+11 0.4820E+11 0.4840E+11
 0.4860E+11 0.4880E+11 0.4900E+11 0.4920E+11 0.4940E+11 0.4960E+11 0.4980E+11
 0.5000E+11

ENTER SOURCE:CH B 1

COULD NOT FIND SOURCE CH B IN DEFW. ERROR: -1

ENTER SOURCE:S3

END OF SOURCE:S3 ENCOUNTERED.

1 2 3

ENTER SOURCE:S2

0.5020E+11 0.5040E+11 0.5060E+11 0.5080E+11 0.5100E+11 0.5120E+11 0.5140E+11
 0.5160E+11 0.5180E+11 0.5200E+11 0.5220E+11 0.5240E+11 0.5260E+11 0.5280E+11
 0.5300E+11 0.5320E+11 0.5340E+11 0.5360E+11 0.5380E+11 0.5400E+11 0.5420E+11
 0.5440E+11 0.5460E+11 0.5480E+11 0.5500E+11 0.5520E+11 0.5540E+11 0.5560E+11
 0.5580E+11 0.5600E+11 0.5620E+11 0.5640E+11 0.5660E+11 0.5680E+11 0.5700E+11
 0.5720E+11 0.5740E+11 0.5760E+11 0.5780E+11 0.5800E+11 0.5820E+11 0.5840E+11
 0.5860E+11 0.5880E+11 0.5900E+11 0.5920E+11 0.5940E+11 0.5960E+11 0.5980E+11
 0.6000E+11

END OF SOURCE:S2 ENCOUNTERED.

ENTER SOURCE:GRP34100

CANNOT RETURN DATA FOR A GROUP:GRP34

ERROR: 11 UNPACKING SOURCE:GRP34

ENTER SOURCE:

3. EBT DATA BASE INDEX

3.1. INTRODUCTION

The on-line index facility of the EBT data base acts as both a file selection and data analysis tool. The index consists of three data sets related in a hierarchical manner. The highest level data set is named "MCH". MCH contains information about steady-state machine conditions. This includes the time and date the machine condition started and ended, a sequence number for each machine condition, and a number of values which define the machine condition. The next level in the hierarchy is the active diagnostic data set which is named "ACTD". Each record in the ACTD data set contains information about the status of a diagnostic or analysis that produced data during a specific machine condition. Every record in ACTD is owned by one and only one record in the MCH data set. One record in MCH may own none to many records in ACTD. The lowest level data set in the index contains data values and is named "DVAL". Each record in DVAL is owned by one record in the ACTD data set. Each DVAL record contains one data value from a diagnostic or analyzed data file.

The on-line index is implemented under the System 1022 (TM) data base management system. System 1022 may be viewed as a very high level interactive programming language. Many data processing functions such as record selection, sorting, updating, and printing are implemented as one-line commands in System 1022. This means that many computer tasks which might require the writing of a FORTRAN program can instead be performed by entering a few lines of commands to System 1022. The reduction in programming time makes it easier to develop new

applications and simpler to modify existing systems. Another advantage to System 1022 is that it allows fast and efficient access to data through the use of inverted indices. An inverted index (called a KEY in System 1022) can be viewed as an array of the unique values found in a data field for all of the records in a data set. The array also contains direct access pointers to the records which contain each value. When one needs to select a set of records based upon the contents of one or more data fields, System 1022 scans the index arrays for the values instead of performing the I/O that would be needed to test the contents of each record. The commands to System 1022 are entered in a simple English-like language where a verb such as FIND, SEARCH, PRINT, or SORT specifies the basic data processing function to be performed. As an example, to select the runs of the EBT experiment where 28 GHZ power was used, the following command would be entered:

FIND P28 GT 0

3.1.1. EBT Data Base Index Structure

The EBT data base index is composed of three System 1022 data sets: MCH machine conditions, ACTD diagnostic status, and DVAL diagnostic values. The data sets are related in a hierarchical manner as shown below.

3.1.2. System 1022 Introduction

System 1022 is a general-purpose data management software system which operates on DECsystem-10 computers. It may be used interactively and in application programs. Related data sets are easily linked for combined management and reporting of information. The System creates and updates large (and small) data collections. A fast retrieval capability gives access to data in large collections of records. A comprehensive report writer displays this data.

3.1.3. System 1022 Feature Overview

1. Data Retrieval: There are no complex rules about retrieving data from a System 1022 data set. The FIND command locates records through indexes which are built and maintained entirely by the System. Many indexes are easily combined in a FIND command to select subsets of the data without anticipating what these subsets will be. Using the FIND command does not take much study; conditions are specified naturally in a free-format statement without complex constructions. The SEARCH command complements the FIND command. It searches sequentially through records and selects those which match the specified conditions. A SEARCH starts from a group of records selected by FIND and narrows the selection.

2. Data Analysis and Reporting: The user displays data with the PRINT and TYPE commands. Arithmetic expressions may be used freely to display data and may use System functions which total, average, compute standard deviations, take square roots, and return minimum and maximum values in the data. A complete output formatter controls the

presentation of data. Fields are quickly specified with formats as in FORTRAN, or they are specified in detail with PICTURE formats similar to ones in COBOL. Scrolled output handles large fields of comments or lists of data.

3. Multiple Datasets: Many data sets may be processed simultaneously to relate their data. The relationships between them are determined by simple, visible matchings between common indexed fields. The MAP command selects records in another data set based on indexed values in common with currently selected records. One or more fields in a record can be the basis for selecting associated records. Data may be retrieved and displayed from more than one data set at the same time.

3.1.4. System 1022 Data Base Structure

The smallest unit of data is an attribute, item, or field. Attributes may contain text information or may be numeric. The current from each of the generators is an example of an attribute in the EBT index. All of the values in System 1022 data sets have already had the constant multiplier specified in their descriptor applied.

A record is a collection of related attributes which are stored as a unit. The EBT index contains three types of records for machine conditions, diagnostic status, and data values.

A data set is a collection of records having the same attributes. The structure of all records in a data set is the same; the order, length and type of attributes do not vary from record to record. A

data set is a file. The EBT index consists of three data sets: MCH machine conditions, ACTD diagnostic status, and WAL data values.

A data base is one or more data sets. It strictly means the data sets which hold information about the related parts of a larger entity, i.e., those data sets which are commonly processed together because changes in one imply changes in the others. It loosely means all data stored in data sets which relate in any way to the activities of the user. A data base has different records for different parts of its structure. One record might relate to many records in another part of the data base. System 1022 relates records in different data sets by matching them on common values in the data. System 1022 considers the EBT index a data base because it is a collection of related data sets.

3.1.5. EBT Index Data Set Formats

In order for System 1022 to process a data set, its format must be defined. The definition consists of assigning a name to each field in a record, and then specifying such things as the location of the field, its data type, if an index should be created for the field, and an optional abbreviation for the name. The definitions entered for the EBT on-line index data sets follow.

3.1.5.1. Fields in the MCH Data Set

<u>NAME</u>	<u>ABBR</u>	<u>INDEXED</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
MC_SEQ	MS	YES	TEXT	The machine condition sequence number of the file the information in this record describes. This field is used as the link to the ACTD data set.

MC_START_DATE	MSD	YES	DATE	The date this machine condition started.
MC_START_TIME	MST	NO	TEXT	The time this machine condition started. Stored as hh:mm:ss.
MC_END_DATE	MED	NO	DATE	The date this machine condition ended.
MC_END_TIME	MET	NO	TEXT	The time this machine condition ended. Stored as hh:mm:ss.
MC_ARCH_VOL	MAV	YES	TEXT	The name of the off-line volume that holds this machine condition. If blank, then the machine condition SPD is on-line.
DENSITY	DENSI	YES	REAL	Interferometer density.
ICRH_FREQUENCY	FREQI	YES	REAL	ICRH frequency.
TOROIDAL_ERR_FLD_CURRENT	TORCU	YES	REAL	Toroidal error field current.
VERTICAL_ERR_FLD_CURRENT	VCUR	YES	REAL	Vertical error field current.
HORIZONTAL_ERR_FLD_CURREN	HCUR	YES	REAL	Horizontal error field current.
PRESSURE_NORTH_1	PN1	YES	REAL	Pressure by cavity.
PRESSURE_NORTH_2	PN2	YES	REAL	
PRESSURE_NORTH_3	PN3	YES	REAL	
PRESSURE_NORTH_4	PN4	YES	REAL	
PRESSURE_NORTH_5	PN5	YES	REAL	
PRESSURE_NORTH_6	PN6	YES	REAL	
PRESSURE_EAST_1	PE1	YES	REAL	
PRESSURE_EAST_2	PE2	YES	REAL	
PRESSURE_EAST_3	PE3	YES	REAL	
PRESSURE_EAST_4	PE4	YES	REAL	
PRESSURE_EAST_5	PE5	YES	REAL	
PRESSURE_EAST_6	PE6	YES	REAL	
PRESSURE_SOUTH_1	PS1	YES	REAL	
PRESSURE_SOUTH_2	PS2	YES	REAL	
PRESSURE_SOUTH_3	PS3	YES	REAL	
PRESSURE_SOUTH_4	PS4	YES	REAL	
PRESSURE_SOUTH_5	PS5	YES	REAL	
PRESSURE_SOUTH_6	PS6	YES	REAL	
PRESSURE_WEST_1	PW1	YES	REAL	
PRESSURE_WEST_2	PW2	YES	REAL	
PRESSURE_WEST_3	PW3	YES	REAL	
PRESSURE_WEST_4	PW4	YES	REAL	

PRESSURE_WEST_5	PW5	YES	REAL	
PRESSURE_WEST_6	PW6	YES	REAL	
WPERP_NORTH_1	WPN1	YES	REAL	WPERP by cavity.
WPERP_NORTH_2	WPN2	YES	REAL	
WPERP_NORTH_3	WPN3	YES	REAL	
WPERP_NORTH_4	WPN4	YES	REAL	
WPERP_NORTH_5	WPN5	YES	REAL	
WPERP_NORTH_6	WPN6	YES	REAL	
WPERP_EAST_1	WPE1	YES	REAL	
WPERP_EAST_2	WPE2	YES	REAL	
WPERP_EAST_3	WPE3	YES	REAL	
WPERP_EAST_4	WPE4	YES	REAL	
WPERP_EAST_5	WPE5	YES	REAL	
WPERP_EAST_6	WPE6	YES	REAL	
WPERP_SOUTH_1	WPS1	YES	REAL	
WPERP_SOUTH_2	WPS2	YES	REAL	
WPERP_SOUTH_3	WPS3	YES	REAL	
WPERP_SOUTH_4	WPS4	YES	REAL	
WPERP_SOUTH_5	WPS5	YES	REAL	
WPERP_SOUTH_6	WPS6	YES	REAL	
WPERP_WEST_1	WPW1	YES	REAL	
WPERP_WEST_2	WPW2	YES	REAL	
WPERP_WEST_3	WPW3	YES	REAL	
WPERP_WEST_4	WPW4	YES	REAL	
WPERP_WEST_5	WPW5	YES	REAL	
WPERP_WEST_6	WPW6	YES	REAL	
GEN_7_CURRENT	GEN7	YES	REAL	Current from generator 7.
GEN_8_CURRENT	GEN8	YES	REAL	Current from generator 8.
GEN_9_CURRENT	GEN9	YES	REAL	Current from generator 9.
GEN_10_CURRENT	GEN10	YES	REAL	Current from generator 10.
ARE_CURRENT	ARE	YES	REAL	Aspect ratio enhancement coil current.
PERCENT_HYDROGEN_GAS	PHG	YES	REAL	Hydrogen gas percentage.
PERCENT_DEUTERIUM_GAS	PDG	YES	REAL	Deuterium gas percentage.
POWER_10_6_GHZ	P10	YES	REAL	10.6 ghz power.
POWER_18_GHZ	P18	YES	REAL	18 ghz power.
POWER_28_GHZ	P28	YES	REAL	28 ghz power.
POWER_ICRH	PICRH	YES	REAL	ICRH power.
POWER_18GHZ_NORTH_1	P18N1	YES	REAL	18 ghz power by cavity.
POWER_18GHZ_NORTH_2	P18N2	YES	REAL	
POWER_18GHZ_NORTH_3	P18N3	YES	REAL	
POWER_18GHZ_NORTH_4	P18N4	YES	REAL	

POWER_18GHZ_NORTH_5	P18N5	YES	REAL
POWER_18GHZ_NORTH_6	P18N6	YES	REAL
POWER_18GHZ_EAST_1	P18E1	YES	REAL
POWER_18GHZ_EAST_2	P18E2	YES	REAL
POWER_18GHZ_EAST_3	P18E3	YES	REAL
POWER_18GHZ_EAST_4	P18E4	YES	REAL
POWER_18GHZ_EAST_5	P18E5	YES	REAL
POWER_18GHZ_EAST_6	P18E6	YES	REAL
POWER_18GHZ_SOUTH_1	P18S1	YES	REAL
POWER_18GHZ_SOUTH_2	P18S2	YES	REAL
POWER_18GHZ_SOUTH_3	P18S3	YES	REAL
POWER_18GHZ_SOUTH_4	P18S4	YES	REAL
POWER_18GHZ_SOUTH_5	P18S5	YES	REAL
POWER_18GHZ_SOUTH_6	P18S6	YES	REAL
POWER_18GHZ_WEST_1	P18W1	YES	REAL
POWER_18GHZ_WEST_2	P18W2	YES	REAL
POWER_18GHZ_WEST_3	P18W3	YES	REAL
POWER_18GHZ_WEST_4	P18W4	YES	REAL
POWER_18GHZ_WEST_5	P18W5	YES	REAL
POWER_18GHZ_WEST_6	P18W6	YES	REAL

3.1.5.2. Fields in the ACTD Data Set

<u>NAME</u>	<u>ABBR</u>	<u>KEYED</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
MC_SEQ	MS	YES	TEXT	The machine condition during which the data file this record describes was developed. This field is used as the link to the MCH data set.
DIAG_ID_SEQ	DIS	YES	TEXT	The diagnostic id and sequence of the file this record describes. This field is used as the link to the DVAL data set.
DIAG_START_DATE	DSD	YES	DATE	The date the file was created.
DIAG_START_TIME	DST	NO	TEXT	The time the file was created.

3.1.5.3. Fields in the DVAL Data Set

<u>NAME</u>	<u>ABBR</u>	<u>KEYED</u>	<u>TYPE</u>	<u>DESCRIPTION</u>
DIAG_ID_SEQ	DIS	YES	TEXT	The diagnostic id and sequence of the file that contains this value. This field is the link to the ACTD data set.
NAME_OF_VALUE	NV	YES	TEXT	The source id from which the value in this record was obtained.
VALUE	V	YES	REAL	The indexed data point.

3.2. USING THE SYSTEM

3.2.1. Entering and Leaving System 1022

To enter System 1022, type the DECsystem-10 command:

R 1022

The System responds with an asterisk (*) when it is ready to accept a command.

@OEBT

Should be the first command entered to System 1022. This opens the EBT index data sets. To leave System 1022, type the command QUIT. System 1022 closes any output files and data sets, and returns the user to monitor level.

3.2.2. System 1022 Command Structure

System 1022 accepts commands in a word-oriented format. The basic form of a System 1022 command is:

COMMAND word word word ...

Each word is separated from the next by a space, tab, or comma.

Multiple commands may be typed on a single line if each is terminated by a period and a space, i.e.,

COMMAND word word. COMMAND word word.

A command may be continued to a new line by ending the current line with an ampersand (&) and a carriage return.

COMMAND word word word & <return>

word word <return>

3.2.3. Summary of System 1022 Commands

The commands in System 1022 used by the EBT data base index are summarized below. The underlined portion of each command is the minimum acceptable abbreviation for the command.

<u>Command</u>	<u>Description</u>
<u>@OEBT</u>	Opens the three EBT index data sets.
<u>FIND</u>	Selects a group of active records for examination, using keyed attributes.
<u>SEARCH</u>	Selects records using attributes' values.
<u>@DMPACT</u>	Writes the names of the files currently selected in the ACTD data set to a disk file named "DMPACT.DAT". This file may then be used to build the call arguments to the NEWFIL routine.
<u>@DMPMCH</u>	Writes the names of the files currently selected in the MCH data set to a disk file named "DMPMCH.DAT". This file may then be used to build the call arguments to the NEWFIL routine.
<u>PRINT</u>	Writes output data to the terminal or a disk file.
<u>TYPE</u>	Displays information to the terminal.
<u>@MCHPRT</u>	Displays the contents of the currently selected records in the MCH index data set.

DBSET Selects a data set for use from among several already open. For the EBT data base index the sets are: 1=MCH, 2=ACTD, 3=DVAL. The numbers and names may be used interchangeably ex: DBSET 3 is the same as DBSET DVAL.

MAP Locates records in a second data set based on attribute values in the current group of records. A number of map commands have been setup for the EBT data base.

@M12 Maps from the machine condition data set to the active diagnostic data set.

@M23 Maps from the active diagnostic set to the diagnostic value set.

@M32 Maps from the diagnostic value set to the active diagnostic set.

@M21 Maps from the active diagnostic set to the machine condition set.

VALUES Displays the data distributions of keyed attributes for selected records.

INFORM Displays information about the data set or about the status of System 1022.

QUIT Ends a System 1022 session and returns to the operating system.

SAVE Saves a binary file containing record numbers for the last selected group of records.

3.2.4. System 1022 Examples

3.2.4.1. System 1022 Example 1

This example shows how to run System 1022, open the EBT index data sets, select all machine condition files made between February 1, and February 27, 1980, and then dump the machine condition sequence numbers to a file for processing by an analysis program.

.R 1022
*@OEBT

;start System 1022
;open the EBT on-line index

```

*FIND MC START DATE BET 2/1/80 2/27/80 ;select EBT runs performed
;on and through Feb. 1 to
; Feb. 27, 1980
10 RECORDS FOUND ;1022's response
*DMPMCH ;copy the identifiers of the
;selected machine runs to a disk
;file named DMPMCH.DAT

```

Listing of the file named "DMPMCH.DAT" produced by the preceding example.

```

A0101
A0103
A0104
A0102
A0110
A0111
A0109
A0105
A0106
A0107

```

3.2.4.2. System 1022 Example 2

The purpose of this example is to list any values indexed for the diagnostic "TST" when the machine condition value MR1 was greater than 1.234 and value M3 was less than 41.

```

.R 1022 ;start System 1022
*CEBT ;open the EBT index
*FIND MR1 GT 1.234 AND M3 LT 41 ;select EBT runs
x RECORDS FOUND ;1022's response
*M12 ;find the records in the
;ACTD data set that were generated
;during the selected EBT runs
;1022's response
y RECORDS FOUND ;now select the records
*FIND LAST AND DIAG ID SEQ CT TST ;created by the "TST" diagnostic
;1022's response
z RECORDS FOUND ;choose the records in the DVAL
*M23 ;data set which contain values
;from the selected runs of the
;TST diagnostic
q RECORDS FOUND ;1022's response

```

```

*TYPE NAME OF VALUE VALUE           ;list the data values found
CHAN1  1.02
CHAN2  3.33
-
-
-
CHAN1  1.999

```

3.2.4.3. System 1022 Example 3

This example finds the set of machine condition files in effect when the TST diagnostic produced a value of between 23.11 and 27.0 from A/D channel number 1.

```

.R 1022                               ;start System 1022
*@OEBT                               ;open the EBT index data
*DBSET DVAL1                         ;process the data value index
*FIND DIS CT TST NV EQ ADC1 AND V BET 23.11 27.0
                                         ;select the records in the DVAL
                                         ;data set generated by the TST
                                         ;diagnostic, which contain a
                                         ;value of between 23.11 and
                                         ;27.0 for the source id ADC1
x RECORDS FOUND                         ;1022's response
*@M32                               ;map up in the index hierarchy to
                                         ;the executions of the TST
                                         ;diagnostic which produced
                                         ;the specified values
y RECORDS FOUND                         ;1022's response
*@M21                               ;finally map up to the runs of the
                                         ;EBT experiment during which the
                                         ;selected executions of the TST
                                         ;diagnostic occurred
z RECORDS FOUND                         ;1022's response
*TYPE MS MSD MST                   ;output the machine condition
                                         ;sequence number, the starting date
                                         ;and time for the selected EBT runs

A0001 2/1/1980 11:22:00
B1012 3/21/1980 10:01:56

.
.
.
A0101 2/27/1980 08:05:11
.
```

3.2.5. Descriptions of System 1022 Commands

3.2.5.1. @OEBT

The @OEBT command opens the three data sets in the EBT data base index. MCH is the active data set after the open. This command should be the first response to System 1022's "##" prompt.

```
.R 1022  
*@OEBT
```

3.2.5.2. FIND Command

The FIND command gives very fast retrieval of information from the data set. This is made possible by the key tables built and maintained for keyed access attributes. The FIND command is restricted to selection conditions on KEYED attributes. Retrieval of information using nonkeyed attributes requires the SEARCH command, which searches the data records themselves.

The form of the FIND command is:

```
FIND <select-condition> [<logical-relation><select-condition>]...
```

which is illustrated by the following examples:

```
FIND P28 30  
FIND WPERP LT 30000 AND P28 GT 50  
FIND GEN_1 GT 5000 OR PRES GT 6.0
```

In these examples the intent of the FIND command is clear. The selection conditions are given in a direct English-like form, which needs no special explanation. Advanced users may prefer to use the abbreviated form of the selection conditions (see the System 1022 manual, Collapsing the FIND Command).

The System normally responds to the FIND Command (and also to SEARCH and MAP) with one of three messages, depending on the number of records found which satisfy the command:

n RECS FOUND
 FOUND
 NOT FOUND

3.2.5.2.1. Basic Selection Conditions. The basic selection condition used in the FIND command has the form

attribute	relational	value
P28	EQ	30

Where:

attribute is the name or abbreviation of a keyed attribute.

relational is one of the following relational operators:

<u>Operator</u>	<u>Meaning</u>
EQ	Is equal to the value
NE	Is not equal to the value
GT	Is greater than the value
GE	Is greater than or equal to the value
LT	Is less than the value
LE	Is less than or equal to the value
CT	Contains the text string in its value (TEXT attributes only)
NCT	Does not contain the text string in its value (TEXT attributes only)

Two additional operators, BETWEEN (also BET) and NBETWEEN (also NBET), specify a range of values, for example:

FIND P28 BET 21 65

finds all records for which P28 is between 21 and 65, inclusive. The

operator NBET is the opposite of BET; it accepts records where the attribute value is outside of the range.

value is the value to which the attribute is compared. The values can be numeric or text constants, date items, or expressions made of these. The value is converted, if necessary, to agree in type with the attribute in the comparison.

The user may type text and date values in the FIND command without quote marks, if they are single words without included spaces. Use care not to enter an unquoted value which is the same as an attribute name; this will confuse the system.

3.2.5.2.2. Logical Operators. Basic selection conditions may be linked by any of the logical operators listed below to form more complex selection criteria:

<u>OPERATOR</u>	<u>DESCRIPTION</u>
AND	Records satisfy both selection conditions.
OR	Records satisfy either selection condition.
XOR	Records satisfy one, but not both selection conditions.
EQV	Records satisfy both conditions or neither.
NOT	Records do not satisfy the selection condition.

The logical operators, when used without parentheses, are evaluated according to the following precedence:

1. XOR and EQV FIRST...
2. AND

3. OR ...LAST

NOT applies only to the relator, selection, or parenthetical group immediately following it.

EXAMPLES:

FIND P28 GE 30 AND P28 LE 35 OR P28 EQ 40

This finds first every machine condition where 28 GHZ power is between 30 and 35, then adds every machine condition where 28 GHZ power is 40.

FIND NOT P28 GE 30 AND P28 LE 35

This finds every machine condition where 28 GHZ power is not 30 or over (LT 30) and at the same time LE 35. We end up with every machine condition where 28 GHZ power is under 30.

FIND P28 GE 30 OR P28 LE 35 AND P28 EQ 40

This finds first every machine condition whose 28 GHZ power is simultaneously LE 35 and equal to 40 (none) and adds every machine condition where 28 GHZ power is 30 or over. We end up with every machine condition where 28 GHZ power is 30 or over.

FIND NOT (P28 GE 30 AND P28 LE 35)

This finds every machine condition where 28 GHZ power is not between 30 and 35: i.e., every machine condition where 28 GHZ power is 29 or less, or 36 or more.

3.2.5.2.3. Special Selection Conditions

The FIND command can also be used with the four following special selection conditions.

1) ALL

The keyword ALL used as a condition in the FIND command selects all active (not deleted) records in the data set and is the preferred way to do this. The resulting message gives the number of active records. FIND ALL is the best way to set up the entire data set for a

following SEARCH on unkeyed attributes through all the records. FIND ALL works even when there are no keyed attributes.

2) LAST

The keyword LAST represents the records in the current selection group just before executing the FIND command containing it. This selection group is usually the result of the last FIND, SEARCH, or MAP command. LAST may be combined logically with other selection groups, and thus may be used either to add records to those already found or to eliminate some previously found.

Examples:

```
FIND P28 35
FIND LAST AND WPERP GT 234
```

The second FIND selects all machine conditions where 28 GHZ power equals 35 and WPERP is greater than 234.

```
FIND NOT LAST
```

This FIND then selects all machine conditions where 28 GHZ power is not equal to 35 and WPERP is not greater than 234.

```
FIND LAST OR MSD 2/1/80.
```

This adds all records taken on Feb. 1, 1980 to the previous selection group.

3) FILE file-descriptor

The SAVE command saves a system-readable copy of the retrieval information for the current selection group. The copy is stored as a disk file, using the default extension of .DMV. (See the SAVE command.)

The file is referenced by the FILE selector condition. This makes it possible to retain different selection groups and make them available for a later operation.

Example:

```
FIND P28 EQ 30
SAVE GRP30
```

```
FIND WPERP EQ 123 AND FILE GRP30
```

The retrieval information for records with P28 equal to 30 is saved in the file called GRP30.DMV. The clause "FILE GRP30" stands for the records in that group.

The file used in a FILE clause must have been created with a System 1022 SAVE command. The system gives an error message for any other kind of file.

3.2.5.2.4. Collapsing the Find Command. The FIND command allows many shortcuts in typing which make sense to the System and which make the entire FIND command logical and natural.

First, the System does not require that the relational EQ be typed in most cases.

FIND P28 EQ 30	Can be typed as
FIND P28 30	
FIND P28 EQ 30 AND WPERP EQ 123	Can be typed as
FIND P28 30 AND WPERP 123	

Second, the attribute name need not be repeated if it is the same as the last one used.

FIND P28 30 OR P28 31	Can be typed as
FIND P28 30 OR 31	

Third, the logical connective OR is not required between values occurring under the same attribute name.

FIND P28 30 OR 31	Can be typed as
FIND P28 30 31	

Fourth, the logical connective AND is not required between different attribute groups.

FIND P28 31 32 AND WPERP 123 234 Can be typed as

FIND P28 31 32 WPERP 123 234

These rules can be stated a different way, allowing for a few exceptions:

1. When the user omits a relational (like EQ, LT, GT), the System supplies one which is the same as the last one typed. The System supplies EQUALS if no relational is typed. The System fills in the last relational until a new selection condition begins, where the relational defaults again to EQ.
2. A logical (like AND, OR) which is omitted between values of the same attribute is filled in as OR.
3. A logical which is omitted between separate selection conditions is filled in as AND.

Thus,

FIND P28 31 32 P28 30 31 29 WPERP 123 234

is the same as typing

FIND (P28 EQ 31 OR P28 EQ 32) AND (P28 EQ 30 OR P28 EQ 31 OR
P28 EQ 29) AND (WPERP EQ 123 OR WPERP EQ 234)

An exception occurs for the arithmetic relational NE, and similarly for NBET and NCT. An expression such as P28 NE 30 31 means P28 NE 30 AND NE 31. The default logical operator connecting the two conditions is changed from OR to AND whenever NE is used. If any logical operator other than AND is used after a NE, then the effect of the NE is cancelled and the normal rules apply. For example:

P28 NE 31 32 OR GT 25 LT 10 is interpreted to mean

P28 NE 31 AND NE 32 OR GT 25 OR LT 10

3.2.5.3. SEARCH Command

The SEARCH command has the same logical structure and format as the FIND command. However, the SEARCH command has greater generality and locates records in a different physical way.

The form of the search command is:

```
SEARCH <selection-condition>[<logical-relation>
<selection-condition>]...
```

The FIND command selects a subset of the entire data set using only KEYED attributes. The selection is rapid due to the special structure that the system builds for KEYED attributes.

The SEARCH command searches only among the records currently selected, usually by the last FIND, SEARCH, or MAP command. SEARCH can perform attribute-to-attribute comparisons on the values found in the records and is not limited to KEYED attributes.

The purpose of the SEARCH command is to reduce the selection of a FIND command. Because the SEARCH command reads through the data records, and not special tables, it usually takes longer than an equivalent FIND command. An efficient strategy is to use a FIND command if possible to narrow down the possibilities and then use a SEARCH on those records found.

A SEARCH command can test a condition between attributes in each record.

For example the following are valid SEARCH commands:

```
SEARCH P28 EG 30
```

```
SEARCH P28 EQ P18-2
```

```
SEARCH RATING1 GT (RATING2+RATING3)*(RATING4-38*9)
```

The CONTAINS relator (or CT) may be used in SEARCH commands. This specifies searching for text attributes which contain the value as a substring rather than requiring the whole attribute to match. "SEARCH CONTAINS" may be more efficient than "FIND CONTAINS" in the case of a small subset of records since FIND CONTAINS requires that the entire key table be searched.

3.2.5.4. @DMPACT Command

The DMPACT command outputs the file names contained in the currently selected records of the ACTD data set. The resulting file is named "DMPACT.DAT" and may be used by analysis programs to build the arguments to the NEWFIL data unpacking routine. The format of the command is:

```
*@DMPACT
```

Listing of a sample "DMPACT.DAT" file.

```
A1101TSTB0123001
B2010RPIN0002
A0001TSTA1645000
```

Where the file consists of:

<u>COLUMNS</u>	<u>VALUE</u>
1-5	MACHINE CONDITION SEQUENCE.
6-8	THE DIAGNOSTIC ID.
9-13	THE DIAGNOSTIC SEQUENCE.
14-16	THE VOLUME THE FILE IS ARCHIVED ON. IF BLANK, THEN THE FILE IS ON-LINE.

3.2.5.5. @DMPMCH Command

The DMPMCH command outputs the machine condition sequence numbers of the currently selected records in the MCH data set. The output file is named "DMPMCH.DAT". This file may then be used by analysis programs

to select machine condition files to process. The form of the command is:

*@DMPMCH

Listing of a sample "DMPMCH.DAT" file.

```
A1101      001
B2010
A0001      000
```

Where the file consists of:

<u>COLUMNS</u>	<u>VALUE</u>
1-5	MACHINE CONDITION SEQUENCE.
6-13	BLANKS.
14-16	THE VOLUME THE FILE IS ARCHIVED ON. IF BLANK, THEN THE FILE IS ON-LINE.

3.2.5.6. PRINT Command

The PRINT command displays record values either to the terminal or to a disk file. It prints expressions formed from attribute values in the records being retrieved and also expressions containing attributes from other than the current data set.

The PRINT command displays the records in the current data set selected by the last FIND, SEARCH, or MAP command. Every record selected is retrieved and displayed by the PRINT command. For example:

.R 1022

7/2/80
System 1022A (114B)

* @OFBI
FILES OPEN. MCH IS CURRENT DB,
FOUND.

* DSET DVAL

* FIND NV GEN

22 RECS FOUND.

* PRINT NV VALUE

```
GEN      5000.0000
GEN      9999.0000
GEN      5000.0000
```

```

GEN      5500.0000
GEN      5500.0000
GEN      .00000000

```

The form of the PRINT command is:

```
PRINT [ ON { <file-descriptor> } ] <output-list>
```

For example: PRINT DIAG_ID_SEQ NAME_OF_VALUE VALUE

```
PRINT ON NAMES.OUT DIAG_ID_SEQ NAME_OF_VALUE VALUE
```

```
PRINT ON 3 DIAG_ID_SEQ NAME_OF_VALUE VALUE
```

The ON clause directs the output to a disk file specified by a file descriptor. If the ON clause is omitted, output is sent to the terminal.

3.2.5.6.1. Output List

The output list has the general form:

```
[<print-list> [FORMAT----END]] [<totals-list> [FORMAT----END]]
```

The print-list is the keyword ALL and/or a list of expressions and attributes which are evaluated and printed for each record selected. The keyword ALL prints the value of each attribute in a record as if the user had typed the name of every attribute.

Example:

```
PRINT ALL
```

```
PRINT MR1+MR3 MC_SEQ M5+1 -<return>
MR2+(MR4*MR5)
```

Each expression is delimited from the next one by at least one space or comma. Spaces may be typed freely within and around expressions.

The FORMAT clause is optional. If it is omitted, the PRINT command uses a default format to print all expressions and attributes.

This default format is:

1. Each attribute value is followed by a space.
2. TEXT values print in the full length of the item as specified in the data set; the values are padded with blanks on the right, if necessary.
3. INTEGER values print in the minimum number of spaces required. This means that a column of values may not align on the right. A minus sign precedes the value if it is negative; no sign appears if it is positive.
4. REAL values print in 10 positions. They have eight significant digits, with decimal point and preceding sign. The sign is "space" for plus and "-" for minus. Should the value be too large or small to print in 10 positions, then scientific (E-field) notation is used. The number prints as a preceding sign, decimal point, 4-digit fraction, and 4-character power of 10, like -.4368E+25.
5. DATE VALUES print in the format month/day/year, as in 10/25/1984.

If the FORMAT clause is included, the System follows the FORMAT exactly to print the data. FORMAT gives the user complete control over how the data looks on the output. See the System 1022 manual for details on PRINT format specifications.

Example:

```
* PRINT NV V DIS FORMAT A5 3X F8.2 3X A8 END
```

The above command prints the following:

GEN	5000.00	OXADNQAA
GEN	9999.00	OXAIQQAA
GEN	5000.00	OXAEQQAA
.		
.		
.		
GEN	5500.00	OXAAQQAA
GEN	5500.00	OXABQQAA
GEN	.00	OXACQQAA

Elements of a print list can be literals, for example:

```
PRINT "HOW ARE YOU" 3 6.7
```

Prints:

```
HOW ARE YOU 3 6.7000000
```

3.2.5.6.2. The ALL Option

The keyword ALL in the print-list causes the values of all the attributes in the data set description to be printed in the order they appear in the data set. ALL stands for a list of all attribute names in the record. The user can determine the names and order of attributes in a PRINT ALL with the INFORM NAMES command.

Example:

```
# DBSET ACTD
# FIND DIS CT RPI
4 RECS FOUND.
# INFORM NAMES
MC_SEQ
DIAG_ID_SEQ
DIAG_START_DATE
DIAG_START_TIME
# PRINT ALL
A0000 RPIA0010 6/03/1980 14:00:30
A0000 RPIA0011 6/03/1980 14:02:32
A0000 RPIA0012 6/03/1980 14:10:55
A0000 RPIA0013 6/03/1980 14:14:47
```

3.2.5.6.3. Totals List

The form of the totals list is:

```
<totals-list> [FORMAT ----- END]
```

The totals-list prints just once after all selected records have been processed by the print-list. This allows the accumulation of totals and other functions which take values from all the records.

The available functions are:

MIN	Prints the minimum attained value of the expression.
MAX	Prints the maximum attained value of the expression.
MEAN	Prints the mean value over all records of the expression.
TOT	Prints the sum of the expression over all records.
TOT1	Prints the number of records in the electron group.
STDEV	Prints the standard deviation of the expression over all records.

The totals-list usually starts by default just after the print-list format clause, or when the first function word is encountered. This allows typing without confusion:

```
PRINT MC_SEQ PRES TOT WPERP
```

This prints the MC_SEQ and PRES from each record, and then the TOTAL WPERP at the end. The print-list is MC_SEQ PRES and the totals-list is TOT WPERP. If the totals-list is to start before the first function is encountered, the word TOTALS may be used as a separator, if no print-list format statement already acts as a default separator. Example:

```
PRINT MC_SEQ PRES TOTALS SYSDATE TOT WPERP
```

In this case, the totals line has today's date followed by the total WPERP.

Sometimes the user wants only a totals line. If so, leave out the print-list entirely, as in:

```
PRINT TOT WPERP
```

```
PRINT TOTALS SYSDATE TOT WPERP
```

The quantities printed as totals are not at all related to those appearing in the print-list. In particular, WPERP need not appear in the print-list to generate TOT WPERP in the totals-list.

The totals-list functions may apply to expressions by following the function name by the expression to be evaluated. They themselves may be combined into expressions.

3.2.5.6.4. Output To Disk Files

The PRINT command can direct output to disk files. These files are specified by a file descriptor. For example:

```
PRINT ON DATA.DAT WPERP MC_SEQ
```

This sets up a correspondence between the output file DATA.DAT and the output generated by PRINT commands. A previous file by the same name is overwritten, not extended, by ON <filename>.

A PRINT command without the ON clause directs its output to the file associated with the last PRINT command in this System 1022 session. If no file has been established, output is to the terminal.

Once an output file is set up, later PRINT commands send output to that file, provided the ON clause is not used. For example:

```
* FIND PRES BET 30 40
10 RECS FOUND
* PRINT ON DATA.DAT WPERP MC_SEQ
* FIND PRES BET 41 50
25 RECS FOUND
* PRINT WPERP MC_SEQ
```

Output is sent to the file until one of the following commands is issued:

```
PRINT ON <file-descriptor>
QUIT
```

The disk file is closed when another ON <file-descriptor> clause is used.

Be careful not to mistakenly overwrite a previous file by using the ON <file-descriptor> clause. For example:

```
* FIND PRES BET 30 40
400 RECS FOUND
* PRINT ON DATA.DAT WPERP MC SEQ
* FIND PRES BET 41 50
25 RECS FOUND
* PRINT ON DATA.DAT WPERP MC SEQ
```

In this example the second PRINT command reinitializes the file NAME.LST, superseding the data first printed.

3.2.5.7. TYPE Command

The TYPE command acts like the PRINT command, except that output is sent only to the user's terminal. The form of the command is:

```
TYPE <output list>
```

3.2.5.8. @MCHPRT Command

The @MCHPRT command lists the contents of the currently selected records in the MCH data set on the user's terminal. An example of the command follows.

```
.R 1022
8/19/80
System 1022A (114B)
* @OEBT
FILES OPEN. MCH IS CURRENT DB,
2 RECS FOUND.
* @MCHPRT

*** NEW MACHINE CONDITION ***
SEQ:A0000 START:JAN-01-1980 00:00:00 END:
DENSITY: .00000000 ICRH FREQ: .00000000 ICRH POWER: .00000000
*** ERROR FIELD CURRENTS ***
TOROIDAL: .00000000 VERTICAL: .00000000 HORIZONTAL: .00000000
*** PRESSURES ***
NORTH .00000000 .00000000 .00000000 .00000000 .00000000 .00000000
EAST .00000000 .00000000 .00000000 .00000000 .00000000 .00000000
SOUTH .00000000 .00000000 .00000000 .00000000 .00000000 .00000000
WEST .00000000 .00000000 .00000000 .00000000 .00000000 .00000000
*** WPERP ***
```

NORTH	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
EAST	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
SOUTH	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
WEST	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000

*** GENERATOR CURRENTS ***

GEN 7: .00000000 GEN 8: .00000000 GEN 9: .00000000 GEN 10: .00000000
ARE COIL CURRENT: .00000000

*** GAS PERCENTAGES *** HYDROGEN: .00000000 DEUTERIUM: .00000000

*** MICROWAVE POWER ***

10.6 GHZ: .00000000 18 GHZ: .00000000 28 GHZ: .00000000

*** 18 GHZ POWER ***

NORTH	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
EAST	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
SOUTH	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000
WEST	.00000000	.00000000	.00000000	.00000000	.00000000	.00000000

*** NEW MACHINE CONDITION ***

SEQ:A0021 START:AUG-15-1980 14:56:49 END:AUG-15-1980 14:59:30 ARCHIVE PACK:
DENSITY: 19.000000 ICRH FREQ: 19.000000 ICRH POWER: 19.000000

*** ERROR FIELD CURRENTS ***

TOROIDAL: 19.000000 VERTICAL: 19.000000 HORIZONTAL: 19.000000

*** PRESSURES ***

NORTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
EAST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
SOUTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
WEST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000

*** WPERP ***

NORTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
EAST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
SOUTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
WEST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000

*** GENERATOR CURRENTS ***

GEN 7: 19.000000 GEN 8: 19.000000 GEN 9: 19.000000 GEN 10: 19.000000
ARE COIL CURRENT: 19.000000

*** GAS PERCENTAGES *** HYDROGEN: 19.000000 DEUTERIUM: 19.000000

*** MICROWAVE POWER ***

10.6 GHZ: 19.000000 18 GHZ: 19.000000 28 GHZ: 19.000000

*** 18 GHZ POWER ***

NORTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
EAST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
SOUTH	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000
WEST	19.000000	19.000000	19.000000	19.000000	19.000000	19.000000

* Q
EXIT

3.2.5.9. DBSET Command

When more than one data set is open, the DBSET command establishes the current data set. Most System 1022 commands operate on the current data set.

The DBSET command has the form:

```
DBSET      {<data set-descriptor>}
           {
           {<n>
           }
```

The <data set-descriptor> selects a data set by giving its name and whatever additional filename and device information is necessary to distinguish it from the other open data sets. For the EBT data base use MCH for the machine condition data set, ACTD for the active diagnostic data set, and DVAL for the diagnostic data set.

The parameter <n> identifies the current data set by its number. MCH=1,ACTD=2,DVAL=3.

All later commands following a DBSET command operate on the selected data set until another DBSET, or MAP.

3.2.5.10. MAP Command

The MAP command locates records in another data set (the second data set) which share common attribute values with the selected records of the 1st, or current data set. The form of the command is

```
MAP [LOGICAL <log>] [T0] <data set-descriptor>
    [VIA] <attr-1> [ [T0] <attr-2> ]
```

Where:

<log> Tells how to combine the result of this MAP with the previously selected records in the result data set. Either AND or OR may be specified. Without this clause, the previous selection group in the result data set has no effect on the result of the MAP.

<data set-descriptor> Is either the data set name or data set number of the second data set, where new records are selected.

<attr-1> Is the name of the attribute in the current data set used as the basis for the mapping. MAP uses

the values of this attribute in the current group of records.

<attr-2>

Is the name of a keyed attribute in the second data set whose values are compared with the values of attr-1 as the basis for the mapping. If attr-2 is omitted, it has the same name as attr-1.

The MAP finds all the records in the second data set with attr-2 equal to any value of attr-1 appearing in selected records of the 1st data set. The values of attr-1 point to the records in the second data set.

Also, the MAP leaves the second data set as the current one. MAP does a DBSET to the second data set and has all the effects of a DBSET command in command files and programs.

The first data set in a MAP is not changed at all by the MAP. The selection group is unchanged when the user DBSETs back to make it the current data set.

For example, consider the two data sets MCH and ACTD, one contains information about steady-state machine conditions and the other information about active diagnostics. The data sets are linked by the common attribute machine condition sequence (MC_SEQ). Suppose we want to select all of the diagnostic files which were generated when machine condition value M1 was greater than 4. The following command sequence could be used:

```
*E
*F 11 GT 4
x RECORDS FOUND
*MAP TO ACTD VIA MC_SEQ
y RECORDS FOUND
*
```

As an example of the use of the LOGICAL clause of the MAP command, consider the case where we want to select all of the data files created by the diagnostic "TST", when the machine condition value MR2 was equal to 4.11 and the diagnostic "TST" generated a value of 4 for source id "SRCA". The following commands could be used:

```
*@OEBT
*FIND MR2 EQ 4.11
x RECORDS FOUND
*MAP TO ACTD VIA MC_SEQ
y RECORDS FOUND
*DBSET DVAL
*FIND DIS CT TST AND NV EQ SRCA AND V EQ 4
z RECORDS FOUND
*MAP LOGICAL AND TO ACTD VIA DIS
q RECORDS FOUND
*
```

A number of commands have been developed to make mapping through the EBT index data sets simple.

<u>STARTING DATA SET</u>	<u>COMMAND</u>	<u>EQUIVALENT MAP COMMAND</u>
MCH	@M12	MAP TO ACTD VIA MC_SEQ
ACTD	@M23	MAP TO DVAL VIA DIAG_ID_SEQ
DVAL	@M32	MAP TO ACTD VIA DIAG_ID_SEQ
ACTD	@M21	MAP TO MCH VIA MC_SEQ

3.2.5.11. VALUES Command

The VALUES command displays information about the value distribution of a keyed attribute. The form of the command is:

```
VALUES [ON <file-descriptor>] <attribute>
```

Data is displayed for just the currently selected records. The data appears in ascending order sorted by the values of the attribute.

To write the information to a disk file, the ON clause may be included with an output file descriptor. Use of this feature does not involve any of the assignable System output channels.

Example:

```
* DBSET 3
* FIND NV GEN V BET 5000 5500
17 RECS FOUND.
* VAL V
5000.00000 (9)
5010.00000 (6)
5500.00000 (2)
```

This indicates 9 records with GEN equal to 5000, 6 records with GEN equal to 5010, etc.

3.2.5.12. INFORM Command

The INFORM command displays information about the structure and condition of a data set. The form of the command is:

```
INFORM { ATTRIBUTE <attribute-name> }
      { NAMES }
      { SET }
      { STRUCTURE }
```

Option: ATTRIBUTE <attribute-name>

This prints just the line from the data set structure which describes this attribute. The attribute abbreviation can be substituted for the attribute name.

```
* INFORM ATTRIBUTE DIS
ATTRIBUTE DIAG_ID_SEQ ABBREV DIS TEXT KEYED COL 1 8
```

Option: NAMES

This prints the full attribute names in the data set.

```
* INFORM NAMES
DIAG_ID_SEQ
NAME_OF_VALUE
VALUE
```

Option: SET

This prints the full data set name of the current data set.

* INFORM SET

DVAL in DVAL.DMS[210,2740]! is set 3

Option: STRUCTURE

This generates a list of all attributes in the data set.

* INFORM STRUCTURE

ATTRIBUTE DIAG_ID_SEQ ABBREV DIS TEXT KEYED COL 1 8

ATTRIBUTE NAME_OF_VALUE ABBREV NV TEXT KEYED COL 9 13

ATTRIBUTE VALUE ABBREV V REAL KEYED COL 14 23

4. OFF-LINE DATA ARCHIVE

4.1. INTRODUCTION

The off-line data archive stores infrequently referenced EBT data files on mountable disk packs. A set of programs has been developed to automatically transfer files to and from the off-line storage. The archive has two basic interfaces, one for the end-user of the EBT data base and the other for the data base coordinator. The end-user interface is through a program named EBTRCL (recall files). EBTRCL accepts requests for files to be recalled to on-line storage. The requested files are copied to the user's on-line disk area after an operator has mounted the necessary disk packs. The recall delay will typically be overnight. The data base coordinator (maintenance) interface consists of three programs: EBTAGE (file aging), EBTMJB (copy to offline), and EBTPAK (copy to on-line). EBTAGE builds a list of the files to be copied off-line based upon the date the files were last accessed and the location of the machine condition the file is to be stored under. The list produced by EBTAGE may be edited using TECO to add or remove files. The EBTMJB program then processes the list and creates a batch job that actually performs the copy of the specified files. EBTPAK scans the file recall requests entered by end-users and builds a batch job to mount the necessary disk packs and copy the requested files to the user's on-line disk area.

4.2. END-USERS INTERFACE

The end-user's interface should be used when a user needs to access a data file that has been moved to the off-line disk packs. This facility will place a request to recall the needed file in a queue of such requests. The file will then be restored to the user's on-line disk area when an operator is available to mount the required disk pack. The recall request queue is stored in a System 1022 data set. The data set may be accessed by entering the following command to System 1022:

```
OPEN DSKB:EBTRCL[210,2740]
```

The status of the queue may then be checked using standard System 1022 commands. The EBTRCL program is used to place recall requests in the queue. EBTRCL will accept the names of the files to be recalled from a file such as the one created by the DMPACT command of the data base index or from the user's terminal.

A description of the prompts output by EBTRCL and the recognized responses follows. The EBTRCL prompt is underlined.

ENTER AN ID FOR THIS GROUP OF FILES: This prompt expects a three- to ten-character code beginning with the user's initials. The code is used to identify this recall request. The status of a request may be found by specifying this code as the argument to the RCLST command. RCLST is described later in this chapter.

RECALL MACH. COND. FILES ALSO? (N/Y): Responding with a "y" to this prompt directs EBTRCL to copy both the files explicitly requested and any associated machine condition files. Any other response results in the copying of only explicitly requested files.

IS THE RECALL LIST IN A FILE? (Y/N): The response to this prompt directs EBTRCL to either read the names of the files to be recalled from a list created in the format used by the DMPACT command or to accept the names from the user's terminal.

A response of "Y" or <return> results in the following output.

```
Unit=2 DSK:DMPACT.DAT/ACCESS=SEQIN/MODE=ASCII
Enter new file specs. and with $(ALT)
#
—
```

If the list is in DMPACT.DAT, respond with \$(ALT). Otherwise, enter the name of the list followed by \$(ALT).

A response of "N" to the "... IN A FILE" prompt causes EBTRCL to input the file names from the user's terminal. The following prompts are used to input the list.

DIAGNOSTIC ID ((CNTRL)Z TO STOP):XXX:

Enter the three-letter code assigned to the diagnostic or analysis which created the file to be recalled. This code is assigned by the data base coordinator. An entry of return causes EBTRCL to default to the value output between the colons in the prompt. Inputting (cntrl)Z results in an exit from the file name input loop.

DIAGNOSTIC SEQUENCE: Input the five letter file name of the file to be recalled. An entry of (cntrl)Z or a

blank line results in an exit from the file name input loop.

ANY MORE FILES? (N/Y): A response by "Y" to this prompt results in a return to the ". . . IN A FILE" prompt so that more file names may be entered. Any other response causes EBTRCL to stop.

An example of an execution of the EBTRCL program follows:

EXAMPLE:

```
.RUN EBTRCL
ENTER AN ID FOR THIS GROUP OF FILES: TSTABC
RECALL THE MACH. COND. FILES ALSO? (N/Y): Y
IS THE RECALL LIST IN A FILE? (Y/N): N
DIAGNOSTIC ID (CNTRL Z TO STOP): TST
DIAGNOSTIC SEQUENCE: B0001
DIAGNOSTIC ID (CNTRL Z TO STOP): TST:OXA
DIAGNOSTIC SEQUENCE: ANQAA
DIAGNOSTIC ID (CNTRL Z TO STOP): OXA:TST
DIAGNOSTIC SEQUENCE: B3333
FILE B3333.TST NOT FOUND.
DIAGNOSTIC ID (CNTRL Z TO STOP): TST:RPI
DIAGNOSTIC SEQUENCE: A0032
DIAGNOSTIC ID (CNTRL Z TO STOP): RPI:_Z
3:INPUT
ANY MORE FILES? (N/Y): Y
IS THE RECALL LIST IN A FILE? (Y/N): Y
Unit=2 DSK:DMPACT.DAT/ACCESS=SEQIN/MODE=ASCII
Enter new file specs. End with an $(ALT)
*$
3:INPUT
ANY MORE FILES? (N/Y): N
```

The following example shows the commands used to create the DMPACT.DAT file used in the preceding example. The user's input is underlined.

EXAMPLE:

```
.R 1022
7/31/80
System 1022A (114B)
* @OEBT
FILES OPEN. MCH IS CURRENT DB,
FOUND.
* DEJET ACTD
* FIND DIS CT NAX AND DSD EQ 5/9/80
```

```

3 RECS FOUND.
* TYPE ALL
A0000 NAXA0010  5/09/1980 14:19:25
A0000 NAXA0012  5/09/1980 14:42:27
A0000 NAXA0013  5/09/1980 16:07:16
* DMPACT
* FILE TYPE DMPACT.DAT
A0000NAXA0010000
A0000NAXA0012000
A0000NAXA0013000
* Q

```

EXIT

The RCLST (recall list) command may be used to determine the status of one or more recall requests. The form of the RCLST command is:

```
.DO RCLST xxx
```

where XXX is either an identifier entered in response to the EBTRCL program's "ENTER AN ID..." prompt or the user's initials.

The action of the RCLST command is to generate a list of all of the unsatisfied recall requests for the identifier or user specified.

4.3. MAINTENANCE INTERFACE

The maintenance interface has two basic functions: to copy infrequently referenced data files off-line and to restore off-line files to an end-user's on-line disk area when needed. The copy to off-line function is implemented through two programs, EBTAGE and EBTMJB. EBTAGE scans the EBT data base's on-line file directory and selects data files to be copied off-line based upon the date the files were last referenced. EBTMJB processes the list of files generated by EBTAGE and creates a batch job to perform the copy. The restore to on-line function is performed by the EBTPAK program. EBTPAK scans the list of files requested by end-users and builds a batch job to copy the files to the requesting user's disk area.

4.3.1. COPY TO OFF-LINE FUNCTION

In order to copy files to the EBT off-line archive, the following steps must be performed.

- 1) LOGIN to the data base account [210,2740],
- 2) Run the EBTAGE program,
- 3) Specify a cutoff date to EBTAGE. All machine conditions that have not been referenced since that date will be selected for archiving,
- 4) Optionally, edit the list generated by EBTAGE using an editor such as TECO. The list is in a file named ARCLST.DAT.
- 5) Run the EBTMJB program.
- 6) Specify the name of the archive disk pack to be used for machine conditions which have not been previously archived.
- 7) Submit the batch control file generated by EBTMJB. The file is named ARCLST.CTL.
- 8) Mount the disk packs requested by the batch job.

Example:

```
.RUN EBTAGE
ENTER THE CUTOFF DATE:8/20/80
EXIT
.TYPE ARCLST.DAT
A0000 0000277
A0021 0000017
.RUN EBTMJB
PACK BLOCKS SFDS
      294      2
      294:TOTAL BLOCKS IN ARCHIVE LIST
ENTER THE PACK TO SEND NEW SFDS TO:001
MC SFDS INPUT:      2
SFDS TO COPY:      2
BAD SFDS:      0
PACKS TO MOUNT:      1
STOP
END OF EXECUTION
CPU TIME: 0.31      ELAPSED TIME: 4.52
EXIT
.TYPE ARCLST.CTL
.MOUNT EBTA:/VID:"EB:001"/SINGLE/WAIT
.COPY EBTA:*.#[210,2740,MA0000]=DSKB:*.#[210,2740,MA0000]
.DELETE DSKB:*.#[210,2740,MA0000]
.DELETE DSKB:MA0000.SFD[210,2740]
```

```
.COPY EBTA:*. *[210,2740,MA0021]=DSKB:*. *[210,2740,MA0021]
.DELETE DSKB:*. *[210,2740,MA0021]
.DELETE DSKB:MA0021.SFD[210,2740]
.SUBMIT ARCLST/TIME
```

4.3.2. RESTORE TO ON-LINE FUNCTION

In order to restore files to the on-line disk area of the end-user, the following steps must be performed.

- 1) Run the EBTPAK program.
- 2) Submit the batch control file created by EBTPAK. The file is called EBTPAK.CTL.
- 3) Mount the disk packs the batch job requests.

Example:

```
.LOGIN 210/2737
JOB 58 ORNL Fusion Energy 603a TTY162
1236 19-Aug-80 Tue
.RUN EBTPAK
STOP
END OF EXECUTION
CPU TIME: 0.77 ELAPSED TIME: 15.38
EXIT
.TYPE EBTPAK.CTL
.MOUNT EBTA:/VID:"EBT000"/SINGLE/WAIT
.NOERROR
.COPY DSKD:A0000.MCH[000210,002737]=EBTA:A0000.MCH[210,2740,MA0000]
.COPY DSKD:A0001.NAX[000210,002737]=EBTA:A0001.NAX[210,2740,MA0000]
.COPY DSKD:A0010.RPI[000210,002737]=EBTA:A0010.RPI[210,2740,MA0000]
.COPY DSKD:A0011.RPI[000210,002737]=EBTA:A0011.RPI[210,2740,MA0000]
.COPY DSKD:A0012.RPI[000210,002737]=EBTA:A0012.RPI[210,2740,MA0000]
.COPY DSKD:A0013.RPI[000210,002737]=EBTA:A0013.RPI[210,2740,MA0000]
.ERROR ?
.DISMOUNT EBTA:/VID:"EBT000"/WAIT
.MOUNT EBTA:/VID:"EBT001"/SINGLE/WAIT
.NOERROR
.COPY DSKD:A0032.RPI[000210,002737]=EBTA:A0032.RPI[210,2740,MA1111]
.COPY DSKD:A1111.MCH[000210,002737]=EBTA:A1111.MCH[210,2740,MA1111]
.COPY DSKD:B0001.TST[000210,002737]=EBTA:B0001.TST[210,2740,MB2222]
.COPY DSKD:B2222.MCH[000210,002737]=EBTA:B2222.MCH[210,2740,MB2222]
.COPY DSKB:A0032.RPI[000210,004111]=EBTA:A0032.RPI[210,2740,MA1111]
.COPY DSKB:A1111.MCH[000210,004111]=EBTA:A1111.MCH[210,2740,MA1111]
.COPY DSKB:B0001.TST[000210,004111]=EBTA:B0001.TST[210,2740,MB2222]
.COPY DSKB:B2222.MCH[000210,004111]=EBTA:B2222.MCH[210,2740,MB2222]
.COPY DSKB:B0001.TST[000210,111111]=EBTA:B0001.TST[210,2740,MB2222]
.COPY DSKB:B2222.MCH[000210,111111]=EBTA:B2222.MCH[210,2740,MB2222]
.COPY DSKA:A0032.RPI[000210,444444]=EBTA:A0032.RPI[210,2740,MA1111]
.COPY DSKA:A1111.MCH[000210,444444]=EBTA:A1111.MCH[210,2740,MA1111]
```

.COPY DSKA:B0001.TST[000210,444444]=EBTA:B0001.TST[210,2740,MB2222]
.COPY DSKA:B2222.MCH[000210,444444]=EBTA:B2222.MCH[210,2740,MB2222]
.ERROR ?
.DISMOUNT EBTA:/VID:"EBT001"/WAIT
.MOUNT EBTA:/VID:"EBT002"/SINGLE/WAIT
.NOERROR
.COPY DSKD:ANQAA.OXA[000210,002737]=EBTA:ANQAA.OXA[210,2740,MC0000]
.COPY DSKD:C0000.MCH[000210,002737]=EBTA:C0000.MCH[210,2740,MC0000]
.COPY DSKB:ANQAA.OXA[000210,004111]=EBTA:ANQAA.OXA[210,2740,MC0000]
.COPY DSKB:C0000.MCH[000210,004111]=EBTA:C0000.MCH[210,2740,MC0000]
.COPY DSKA:ANQAA.OXA[000210,444444]=EBTA:ANQAA.OXA[210,2740,MC0000]
.COPY DSKA:C0000.MCH[000210,444444]=EBTA:C0000.MCH[210,2740,MC0000]
.ERROR ?
.DISMOUNT EBTA:/VID:"EBT002"/WAIT

.SUBMIT EBTPAK/TIME

APPENDIX A

C COMMON AREAS USED BY EBT DATA UNPACKING ROUTINES
 C PARAMETEPS ARSIZ,HARSIZ,GRPSIZ, AND NUMBUF MUST BE DEFINED IN THE
 C MAIN CALLING PROGRAM.
 C ARSIZ SETS THE MAXIMUM NUMBER OF DATA SOURCES
 C AND GROUPS WHICH MAY BE DEFINED FOR A DATA
 C FILE.
 C HARSIZ SETS THE SIZE OF THE ARRAYS WHICH KEEP
 C TRACK OF THE OCCURRENCE COUNTS OF THE OWNERS
 C OF SOURCES. HARSIZ SHOULD BE SET TO GREATER
 C THAN THE SUM OF THE NUMBER OF DATA SOURCES IN
 C A GROUP TIMES THE NESTING LEVEL OF THE SOURCES,
 C FOR ALL GROUPS IN THE DATA FILE.
 C GRPSIZ ESTABLISHES THE SIZE OF THE STACK USED FOR
 C PROCESSING NESTED GROUPS
 C NUMBUF CONTROLS THE NUMBER OF BUFFERS ALLOCATED FOR THE
 C DATA FILE. EACH BUFFER CONTAINS 128 WORDS. IF
 C PROCESSING OF THE FILE IS RANDOM, THEN NUMBUF
 C SHOULD BE SET TO 1. IF THE FILE IS PROCESSED
 C SEQUENTIALLY, THEN NUMBUF SHOULD BE SET TO 10.
 C

INTEGER ICURR
 INTEGER BUFCNT,GCP
 INTEGER OWNER,SEQ,SRCID,SRCGRP,FRSTWD,LSTUSD,DATYPE
 INTEGER WLEN,CLUSTER,CRCLST,GRPHID,GRPCLS,GRPFIR,GRPEND
 INTEGER SIGNIF,INDEX,SRCMAX,SRCCNT
 INTEGER DFDI,DFDK,DFDATE,DFCOM
 INTEGER DDDI,DDSEQ,DDDK,DDSMC,DDSDT,DDEMC,DDEDI
 REAL*4 UNITS

C THE FOLLOWING COMMON AREAS ARE USED TO HOLD
 C DEFINITIONS OF DATA SOURCES AND GROUPS
 C
 C ICURR POINTS TO THE CURRENT ENTRY IN THE DATA UNPACKING
 C ARRAYS
 C COMMON /ICURR/ICURR
 C OWNER HOLDS THE INDEX OF THE OWNER OF AN ITEM
 C COMMON /OWNER/OWNER(ARSIZ)
 C SEQ HOLDS THE SEQUENCE NUMBER OF AN ITEM IN ITS OWNER
 C COMMON /SEQ/SEQ(ARSIZ)
 C SRCID HOLDS THE NAME OF AN ITEM
 C COMMON /SRCID/SRCID(ARSIZ)
 C SRCGRP IS 0 IF AN ITEM IS A SOURCE, 1 IF A GROUP
 C COMMON /SRCGRP/SRCGRP(ARSIZ)
 C FRSTWD POINTS TO THE FIRST WORD OF AN ITEM IN THE
 C DATA FILE
 C COMMON /FRSTWD/FRSTWD(ARSIZ)
 C LSTUSD POINTS TO THE WORD LAST REFERENCED IN AN ITEM
 C COMMON /LSTUSD/LSTUSD(ARSIZ)
 C DATYPE HOLDS THE DATA TYPE OF AN ITEM
 C 1=INT.,2=REAL,3=DOUBLE,4=A5 ASCII
 C COMMON /DATYPE/DATYPE(ARSIZ)
 C WLEN CONTAINS THE WORD LENGTH OF ONE OCCURRENCE OF
 C AN ITEM

```

COMMON /WDLEN/WDLEN(ARSIZ)
C   CLUSTER HOLDS THE CLUSTER FACTOR FOR AN ITEM.
C   -1=EOF,0=DEFINED IN DATA,+NUMBER=NUMBER OF OCCURRENCES
COMMON /CLUSTER/CLUSTER(ARSIZ)
C   CRCLST HOLDS THE NUMBER OF OCCURRENCES OF AN ITEM
C   REMAINING TO BE PROCESSED
COMMON /CRCLST/CRCLST(ARSIZ)
C   GRPFIR AND GRPEND ARE USED DIFFERENTLY DEPENDING ON WHETHER
C   AN ITEM IS A SOURCE OR A GROUP.
C
C   IF A GROUP GRPFIR POINTS TO THE DEFN OF THE FIRST MEMBER
C   OF THE GROUP AND GRPEND POINTS TO THE LAST MEMBER.
C
C   IF A SOURCE AND THE SOURCE IS A MEMBER OF A GROUP THEN
C   GRPFIR POINTS TO THE FIRST ENTRY IN GROUP CLUSTER ARRAYS
C   BELONGING TO A SOURCE AND GRPEND POINTS TO THE LAST
C
C   OTHERWISE GRPFIR AND GRPEND ARE NOT USED.
C
COMMON /GRPFIR/GRPFIR(ARSIZ)
COMMON /GRPEND/GRPEND(ARSIZ)
C   SIGNIF HOLDS THE NUMBER OF SIGNIFICANT BITS OF DATA
C   FOR AN ITEM
COMMON /SIGNIF/SIGNIF(ARSIZ)
C   INDEX HOLDS THE TYPE OF INDEXING TO BE PROVIDED OF AN ITEM
COMMON /INDEX/INDEX(ARSIZ)
C   UNITS IS A NUMBER WHICH WHEN MULTIPLIED BY THE DATA IN THE FILE
C   FOR A SOURCE RESULTS IN SOME STANDARD UNIT.
COMMON /UNITS/UNITS(ARSIZ)
C   SRCMAX=THE VALUE OF ARSIZ, SRCCNT=THE NUMBER OF DEFN INPUT
COMMON /SRCMAX/SRCMAX,SRCCNT
C
C   THE FOLLOWING COMMON AREAS ARE USED TO KEEP TRACK OF
C   THE OCCURRENCE COUNTS OF THE OWNERS OF A SOURCE.
C
C   GRPHID HOLDS A POINTER TO THE OWNER
COMMON /GRPHID/GRPHID(0:HARSIZ)
C   GRPCLS HOLDS THE ACTIVE CLUSTER COUNT FOR AN OWNER
COMMON /GRPCLS/GRPCLS(HARSIZ)
C   GCASIZ=HARSIZ GCP=NUMBER OF ENTRIES USED
COMMON /GCASIZ/GCASIZ,GCP
C   DEFNHDR HOLDS THE CONTENTS OF THE HEADER OF THE CURRENT
C   DEFINITION FILE IN ASCII
COMMON /DEFNHDR/DEFDI,DEFDK,DEFDATE(3),DEFCON(16)
C   DDHDR HOLDS THE CONTENTS OF THE CURRENT DATA FILE HEADER
COMMON /DDHDR/DDDI,DDSEQ,DDDK,DDSMC,DDSDT(4),
1DDMC,DDDET(4)
C   THE FOLLOWING COMMON AREAS ARE USED TO PROCESS NESTED GROUPS
COMMON /IGPA/IGPA(GRPSIZ)
COMMON /IGTCA/IGTCA(GRPSIZ)
COMMON /IGPTR/IGPTR(GRPSIZ)
C   THIS COMMON IS USED TO SET THE BUFFER COUNT OF THE DATA FILE
COMMON /BUFCNT/BUFCNT
C   1022 DATA BASE MC INDEX RECORD SIZE
COMMON /I1022F/MCSIZ

```

```

C      MACHINE CONDITION INDEXED DATA VALUES
C      COMMON /VALMC/VALMC(110)
C      VALMC(1)      MC SEQ # 'ASCII'
C      VALMC(2)      MC START DATE IN 1022 BINARY FORM
C      VALMC(3&4)    MC START TIME 'ASCII' HH:MM:SS
C      VALMC(5)      MC END DATE IN 1022 BINARY FORM
C      VALMC(6&7)    MC END TIME 'ASCII' HH:MM:SS
C      VALMC(8)      MC ARCHIVE DISK VOLUME 'ASCII'
C      VALMC(9)      DENSITY 'REAL'
C      VALMC(10)     ICRH FREQUENCY 'REAL'
C      VALMC(11)     TOROIDAL ERROR FIELD CURRENT 'REAL'
C      VALMC(12)     VERTICAL ERROR FIELD CURRENT 'REAL'
C      VALMC(13)     HORIZONTAL ERROR FIELD CURRENT 'REAL'
C      VALMC(14)     AVERAGE PRESSURE 'REAL'
C      VALMC(15)     AVERAGE WPERP 'REAL'
C      VALMC(16-39)  PRESSURE BY CAVITY 'REAL'
C      VALMC(40-63)  WPERP BY CAVITY 'REAL'
C      VALMC(64-68)  GENERATOR CURRENTS 'REAL'
C      VALMC(69-70)  GAS PERCENTAGES 'REAL'
C      VALMC(71-74)  POWER 'REAL'
C      VALMC(75-98)  18 G/Z POWER BY CAVITY 'REAL'
C
C
C      *****END OF EBTCOM*****
C
C

```