NUREG/CR-4639
EGG-2458
Volume 2

Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR)

Programmer's Guide

Prepared by O. J. Call, J. A. Jacobson

Idaho National Engineering Laboratory
EG&G Idaho, Inc.

Prepared for
U.S. Nuclear Regulatory
Commission

NUREG/CR-4639
EGG-2458
Volume 2
RX

Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR)

Programmer's Guide

Manuscript Completed: September 1988
Date Published: September 1988

Prepared by
O. J. Call, J. A. Jacobson

Idaho National Engineering Laboratory
Managed by the U.S. Department of Energy

EG&G Idaho, Inc.
Idaho Falls, ID 83415

Prepared for
Division of Systems Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555
NRC FIN A6850
Under DOE Contract No. DE-AC07-761D01570

ABSTRACT

The Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR) is an automated data base management system for processing and storing human error probability and hardware component failure data. The NUCLARR system software resides on an IBM (or compatible) personal micro-computer and can be used to furnish data inputs for both human and hardware reliability analysis in support of a variety of risk assessment activities.

The NUCLARR system is documented in a five-volume series of reports. Volume II of this series is the Programmer's Guide for maintaining the NUCLARR system software. This Programmer's Guide provides, for the software engineer, an orientation to the software elements involved, discusses maintenance methods, and presents useful aids and examples.

SUMMARY

The Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR) is documented in a series of five volumes. Volume I: Summary Description is a general overview of the NUCLARR system. Volume I provides the background of the NUCLARR program, including a description of methods for data collection, system specification, data structures, and taxonomies. Volume II: Programmer's Guide provides information for maintaining the software for the NUCLARR system. Volume III: Data Base Management Guide for Processing Data and Revising the Data Manual contains the procedures for processing human error probability and hardware component failure data and entering the data values into the NUCLARR system. Volume IV: User's Guide instructs the end user in operating the NUCLARR software. Volume V: Data Manual is a hard-copy report of the data residing in the NUCLARR system.

This report, Volume II, is addressed to experienced software engineers who will be responsible for maintenance of the NUCLARR system software. The narrative sections of this Programmer's Guide present discussions of the resources and methods used for system development and maintenance. Besides providing in-depth explanatory material here, the reader is directed in obtaining more detailed technical information by identification of specific tools and other products used. Also included are several appendices which provide examples and other maintenance aids.

Requests for information or for obtaining the NUCLARR software and/or documentation should be directed to either of the following:

Thomas G. Ryan
U.S. Nuclear Regulatory Commission - RES
Reliability and Human Factors Branch
5640 Nicholson Lane, NL/N-316
Rockville, MD 20852 USA
(Phone) 301-492-3550

David I. Gertman
NUCLARR Data Clearinghouse
Idaho National Engineering Laboratory
P. O. Box 1625
Idaho Falls, ID 83415 USA
(Phone) 208-526-0652

ACKNOWLEDGMENTS

We are grateful to Dr. T. G. Ryan, of the U.S. Nuclear Regulatory Commission (NRC), for his continued contributions as Technical Monitor for this program. We appreciate the technical direction and recommendations contributed by D. I. Gertman and W. E. Gilmore from the Human Factors Research Unit at the Idaho National Engineering Laboratory (INEL). The authors would also like to thank G. H. Beers, D. J. Fink, P. M. McGuire, and T. H. Tucker, from the Special Applications Unit at the INEL, for their efforts in software development and for their insight and willing response to questions about the software during document preparation.

In addition, we owe special appreciation to N. L. Wade, also of the INEL, for her assistance as technical editor in the preparation of this report.

Finally, we thank R. N. Hagen of the Special Applications Unit at the INEL for his support and guidance in the mechanics of producing this report.

CONTENTS

FIGURES

TABLES

ACRONYMS

ASCII	American Standard Code for Information Interchange
DOS	disk operating system
HEP	human error probability
HCF	hardware component failure
IBM	International Business Machines Corporation
INEL	Idaho National Engineering Laboratory
NRC	U.S. Nuclear Regulatory Commission
NUCLARR	Nuclear Computerized Library for Assessing Reactor Reliability
PC	personal computer

NUCLEAR COMPUTERIZED LIBRARY FOR ASSESSING REACTOR RELIABILITY (NUCLARR) VOLUME II: PROGRAMMER'S GUIDE

INTRODUCTION

The Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR) is a data base management system used to process, store, and retrieve human and hardware reliability data in a ready-to-use format. The NUCLARR system was developed by the U.S. Nuclear Regulatory Commission (NRC) to provide the risk analysis community a repository of data that can be used to support a variety of risk assessment activities. The system provides a broad range of data base management functions, computational algorithms for aggregating the source data, and mechanisms for report and plot generation. The system software is designed for operation on an IBM* personal computer (PC) or compatible microcomputer.

The purpose of this particular volume is to provide information necessary for maintaining and modifying the NUCLARR system software. The material presented is intended for use by software engineers with experience in using the programming language, specialized data base and system development tools, procedure libraries, and disk operating system (DOS) used in the development and implementation of NUCLARR. For more detailed information than is provided herein, the software engineer will need to access documentation on the tools and other products identified in subsequent sections of this volume. The remainder of this introduction reviews the presentation format used for the rest of this document.

The next three sections describe the environment, structures, and organization of the software elements used in the NUCLARR system, providing insights into the design and functions of NUCLARR. Further details on the design and implementation of the NUCLARR system may be found in the other volumes of this five-volume series. Of special interest would be Volume I: Summary Description[1] and Volume IV: User's Guide[2].

The subsequent two sections describe the processes involved with generating an updated form of the NUCLARR software and of data base records and forms.

The following section provides descriptions of NUCLARR system programs and libraries to assist in locating specific procedures while resolving software problems. These descriptions also contain some information regarding methodology and techniques involved in coding.

The final narrative section presents a summary regarding use of this volume.

In order to provide further support to users of this volume, a series of appendices, A through H, is available with either examples or detailed maintenance aids mentioned previously in this volume.

---

*Mention of specific products and/or manufacturers in this document implies neither endorsement of preference nor disapproval by the U.S. Government, any of its agencies, or EG&G Idaho, Inc., of the use of a specific product for any purpose.

PROGRAMMING ENVIRONMENT

Software for the NUCLARR system was developed and is to be maintained using the Modula-2 programming language on an IBM (or compatible) micro-computer. The design and implementation of NUCLARR was done using Modula-2/86 from Logitech, Inc. of Redwood City, California. The Modula-2 language environment includes a compiler, linker, debugger, and libraries. A set of libraries, referred to as the Ad Hoc Standard Library, is used to augment the Logitech-supplied libraries. The Ad Hoc Standard Library was originally supplied by Pacific Systems Group of Coos Bay, Oregon. Further, the Logitech-supplied utility LOD2EXE is used to transform load files (generated by the linker) into executable files.

The SAGE[3] application development system, developed at the Idaho National Engineering Laboratory (INEL), is used to increase programmer productivity and to greatly reduce software maintenance costs. SAGE provides efficient and versatile relational data base tools, graphics, formatted screens and menus, online helps, editing, and many other capabilities used in NUCLARR. The SAGE features used were also written using the Modula-2 programming language.

Within the SAGE system, a new ad hoc library was developed to augment the Logitech-supplied libraries. This library set has replaced the one supplied by Pacific Systems Group.

SAGE system utilities used in developing and maintaining the NUCLARR system are THOR, REBUILD, and DOCUPROC. The THOR utility is used to create and edit forms, develop and maintain data base record structures, generate reports of forms and data base structures, and compile the data base definition file. The REBUILD utility is used to rebuild lost or damaged data base index files and to restructure data base data files when record structures are added or revised. The DOCUPROC utility is used to generate ASCII file reports such as Modula-2 module listings. It may be of interest to note that the names SAGE and THOR are not acronyms but simply names.

Plotting/graphics capabilities are provided through the CRYSTAL Graphics[4] set of libraries developed at INEL. These libraries make calls to Graphics Kernel System (GKS) modules which in turn use the HALO Graphics interface which is an efficient set of assembly language graphic primitives. HALO is supplied by Media Cybernetics, Inc. of Silver Spring, Maryland. CRYSTAL Graphics was specifically designed to function within the SAGE system.

Appendix A identifies where to locate documentation for the libraries accessed from the above described software environment of the NUCLARR system. In the high level flow charts of Appendices B1, B2, C1, C2, and D, references are made to procedures from libraries covered by Appendix A as well as to procedures coded explicitly for NUCLARR, some of which are found charted in Appendices B1 and C1.

Of course, a text editor is also necessary for developing and editing program source code. So far, the text editor of choice has been Kedit from Mansfield Software Group of Storrs, Connecticut.

The micro-computer used for NUCLARR system maintenance must be IBM or IBM compatible and should be at the IBM PC-AT level or above in capability with disk operating system at DOS 3.0 or above capabilities. This computer needs to have at least

640 k bytes of memory, an enhanced graphics card, a math 8087 coprocessor card, a color monitor, at least 30 megabytes of fixed disk space, and a disk drive for 5 1/4 inch floppy disks. A printer which has the IBM graphics font should be accessible from this computer.

To give an idea of the size of the NUCLARR system, it is here noted that the number of lines of code in the current software developed for NUCLARR is 66,583.

A summary of the items addressed in this section is shown in Table 1.

Table 1. Environment Summary for NUCLARR System Maintenance

Major Elements	Key Subunits or Associated Units
Modula-2 language environment	Ad Hoc Standard Library LOD2EXE utility
SAGE system	Ad Hoc Standard Library THOR utility REBUILD utility DOCUPROC utility
CRYSTAL Graphics Library	Graphics Kernel System (GKS) HALO Graphics interface
Text editor	
Hardware	IBM PC-AT or compatible computer Printer with IBM graphics font

DATA STRUCTURE

NUCLARR data are structured and managed in relational data base form. In a relational data base, there are multiple relations, with each relation consisting of data records. Each data record has one or more fields defined for it, and that field definition applies to every record in a given relation. These fields are the references to the data stored in the data base. Each record type resides in its own relation.

In the data base structure used for NUCLARR, there are up to three files associated with each relation: a data file, an index file, and a blockdata file. A data file contains fixed-length data values. A blockdata file contains variable-length data values, with each data value managed as a block of connected data rather than as a single piece of data. Index files keep track of where the data is stored on the data files and the blockdata files. See Appendices E and F for the data base file names used in the NUCLARR system.

The definition of the structure of the fields in data records for a given relation constitutes what is called the 'schema' for that relation. This schema defines the type, format, and value range allowed for data entered into each field of the data record type assigned to the given relation. Also in this schema a name and description are specified for

each field. How fields are used in search processing is also prescribed in this schema. Such a schema provides the uniformity of data handling within each relation; i.e., every record in a given relation will have data organized using the same field structure. The set of all these relation schemas constitutes the schema for the data base.

Also associated with a data base is a collection of forms. These forms may be used for various purposes such as on-screen helps, selection menus, and data presentation for user data handling and for reporting mechanisms.

When schemas or forms are created or modified using THOR, the data base source file needs to be compiled and the resulting definition file made available for use in the NUCLARR system before these alterations will be effective in NUCLARR. In this volume, see the section titled DATA BASE COMPILATION for more details concerning this process.

In the NUCLARR system, three data bases are used. Data base records and forms are reviewed by using THOR to either view them on-screen or to generate a hard copy report. DATABANK is the name of the data base used for human error probability (HEP) data. HARDWARE is the name of the data base used for hardware component failure (HCF) data. MASTER is the name of the data base used to select which of the other two data bases will be accessed for data retrieval. Appendix D shows a THOR report of the only record and form in MASTER. Appendices E and F provide sample THOR reports showing some records and some forms from DATABANK and HARDWARE, respectively.

SOFTWARE ORGANIZATION

The NUCLARR system is organized as a series of Modula-2 programming language modules. These modules consist of definition blocks and procedures. Procedures can also have their own definition blocks. All procedures begin with the reserved word PROCEDURE. Two special types of modules are used in combination to define a library. A module beginning with the reserved words DEFINITION MODULE defines what is exportable from the library whose name follows these reserved words. A module beginning with the reserved words IMPLEMENTATION MODULE contains the executable code of the library whose name follows these reserved words. Such a module pair will be what is referenced when the word 'library' is used in the remainder of Volume II. Modules other than those used to create libraries begin with the reserved word MODULE and will be what is referenced when the word 'module' is used in the remainder of Volume II.

Through use of the SAGE system, modules and libraries access the appropriate elements of the three data bases, MASTER, DATABANK, and HARDWARE. Such interfacing and other processing actions make use of library sets which are provided by the Modula-2 vendor and the SAGE system or by associated products. See Appendix A for the key to locating documentation on these library sets.

The modules and libraries programmed for NUCLARR are segregated into three categories. There are modules and libraries used for HEP data processing, modules and libraries used for HCF data processing, and modules used to set up retrieval access to the selected data type. Figure 1 shows the relationships involved between batch files (.BAT suffixed) and module executable files (.EXE suffixed) in the file structure used to set up retrieval access.

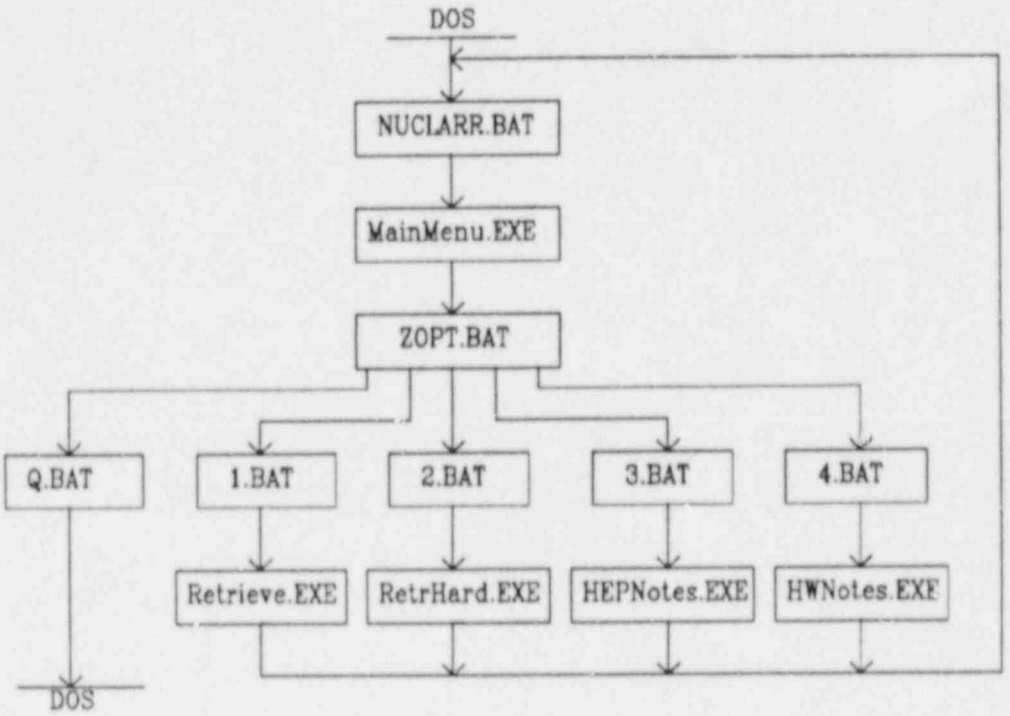


Figure 1. File structure used for retrieval access.

The modules programmed for NUCLARR data processing can be further classified as to whether they are used for maintaining a data base or for retrieving data from a data base. Only the modules used to maintain DATABANK and those used to set up retrieval access do not become involved with overlay structuring.

An overlay structure provides for holding in memory only those modules of the system which are needed for current processing activities. Such a structure consists of a base overlay and one or more additional levels of overlay with one or more modules in each overlay. However, there can be only one module in the base overlay. Once a base overlay is invoked, it remains in memory until its associated overlay structure is released. One, and only one, module from as many overlay levels as desired may be memory resident at one time. Each module is kept on a separate file. Overlay structure is established at the time compiled modules are linked. Then the LOD2EXE utility is used to create .EXE-suffixed files for base overlay modules.

There is one more term used in association with modules. This term, 'program', is a reference to a functionally complete unit of software. In general, 'program' is a reference to a module together with all the software units it imports. However, when overlay structures are used, the term 'program' refers to all the modules, and their imported units, included in the structure.

A series of five figures are presented to show the executable file structure of those modules involved in overlay structures. A base overlay file has a suffix of .EXE. Files used in other overlay levels have a suffix of .LOD. Program entry occurs at the base overlay level. As the program executes, modules are accessed based on options selected by the user.

Figure 2 shows the file structure used in the retrieval of data from the data base

named DATABANK. Figure 3 shows the file structure used in maintaining the data base named HARDWARE.

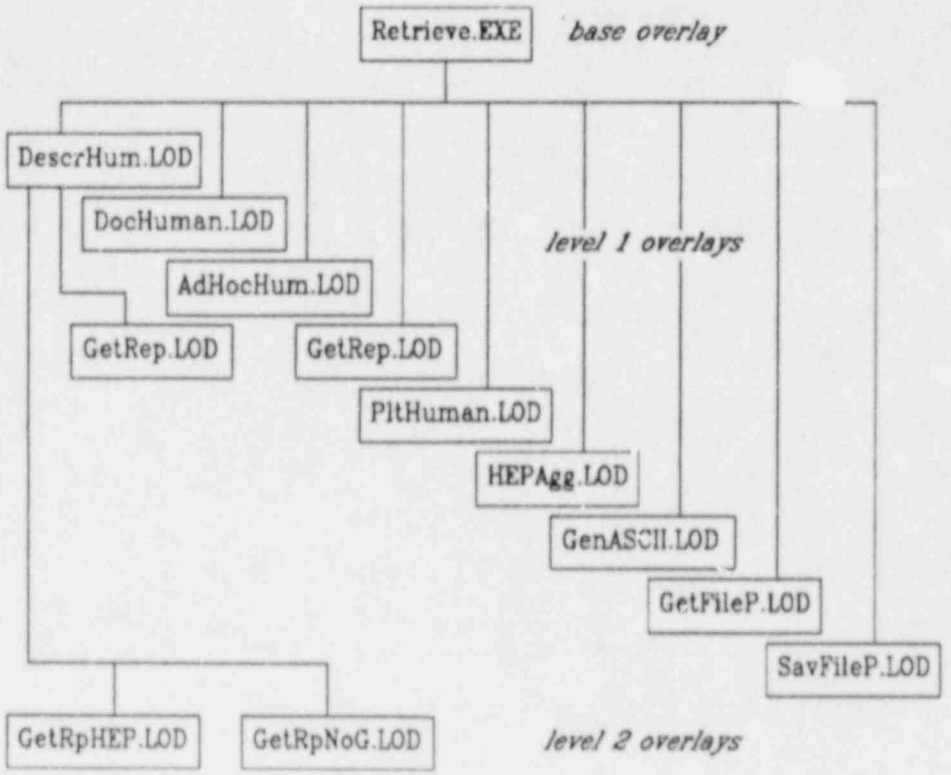


Figure 2. Overlay structure for the program Retrieve.

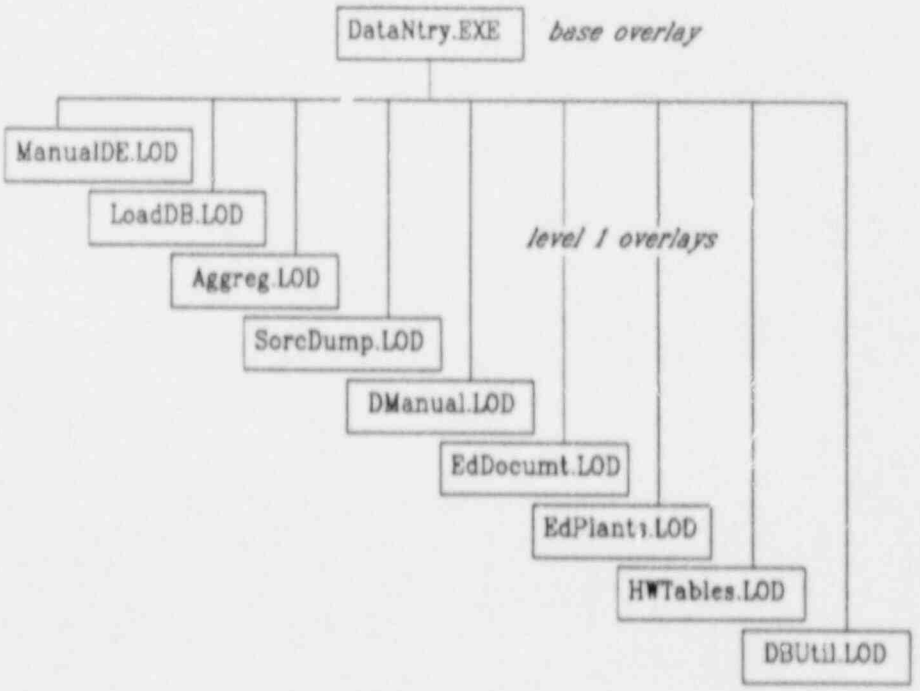


Figure 3. Overlay structure for the program DataNtry.

Figures 4 through 6 show the file structure used in the retrieval of data from the data base named HARDWARE. Figure 4 shows the base overlay, level 1 overlays, and access to level 2 overlays. Figure 5 shows the descriptive search level 1 overlay and its associated level 2 overlays. Figure 6 shows the ad hoc search level 1 overlay and its associated level 2 overlays.

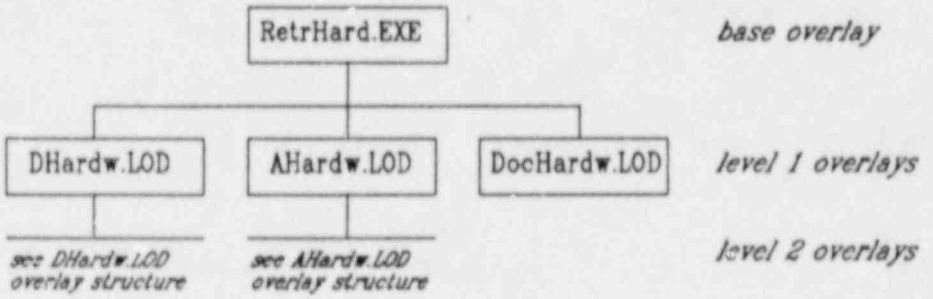


Figure 4. Overlay structure for the program RetrHard.

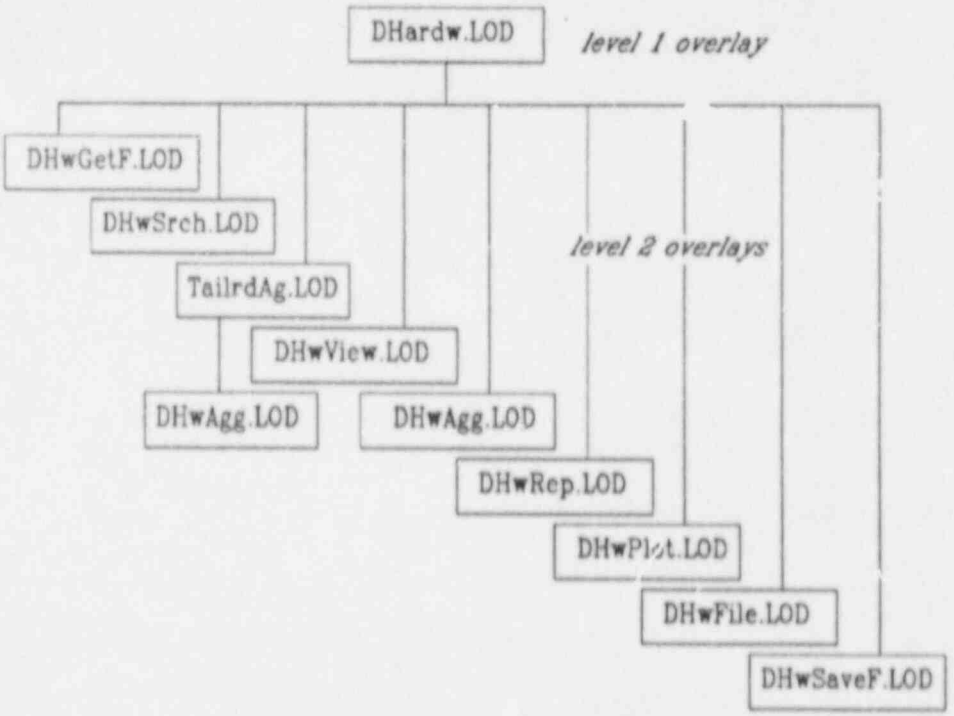


Figure 5. Overlay structure for the module DHardw in program RetrHard.

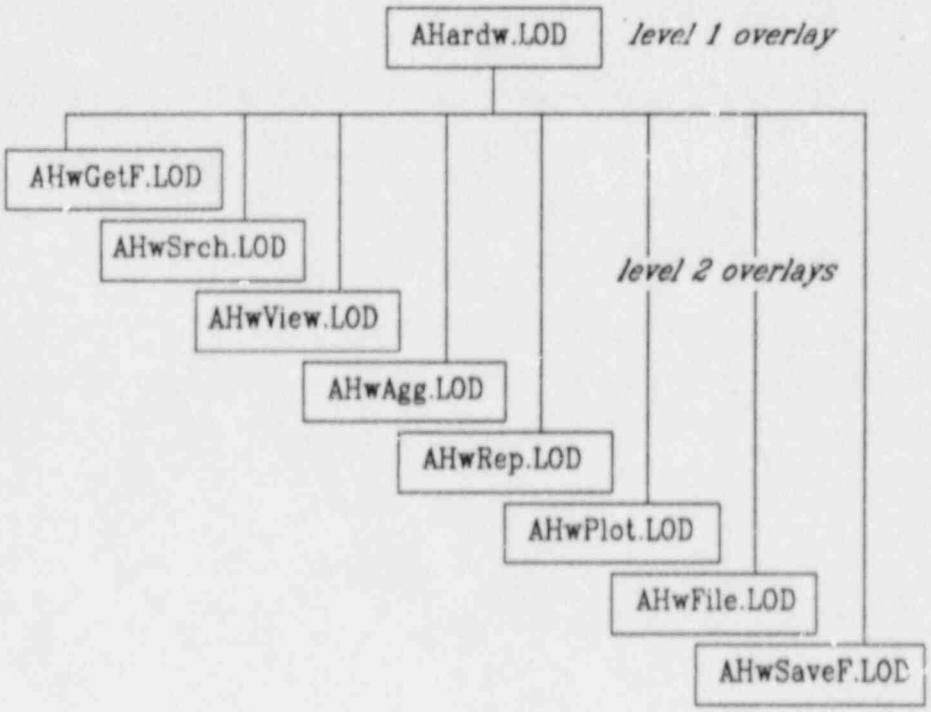


Figure 6. Overlay structure for the module AHardw in program RetrHard.

PROGRAM COMPILATION AND LINKING

The Sage system tools used are all compiled using the math coprocessor option for more efficient real number arithmetic. Therefore, this option must be selected when compiling NUCLARR modules.

Version checking exists in the Modula-2 language environment to prevent version mismatches. This version checking enforces the following rule: if file 1 is to be used by file 2, then file 1 must have a version time-stamp which is earlier than that of file 2. The two following statements are a consequence of this rule.

1. For library files, a DEFINITION MODULE (file name suffixed with .DEF) must have a time stamp which is earlier than that of its corresponding IMPLEMENTATION MODULE (file name suffixed with .MOD).

2. Any library files accessed must have a time stamp which is earlier than that of the file which makes the import request.

After modules and libraries are created or modified, it is necessary to be sure that the version checking rule is complied with while compiling modules and libraries and while linking programs. Compilation and/or linking will not be successful if the version checking rule is violated.

After a library's DEFINITION MODULE is compiled, its IMPLEMENTATION MODULE must be compiled before the library can be used and any software unit importing from that library must then be compiled. After a module is compiled, it must then be

linked before that version can be executed. Also, any module importing from a library must be linked after that library's compilation in order to access the current version of that library.

A module which is to be executed directly from DOS should be processed by the LOD2EXE utility after its .LOD suffixed file is generated by the link process. Using LOD2EXE generates a .EXE suffixed file, e.g. RETRIEVE.EXE. The .EXE-suffixed file is used when execution is requested by a DOS command which is simply a program name, e.g. RETRIEVE.

The remainder of this section shows the normal ordering of commands used to compile libraries and modules and to link modules. These command lists are grouped into usage categories and should be used in the order given within a category. All of the commands listed would be used if recompiling all libraries and programs of the NUCLARR system. Such a universal recompilation would be needed, for example, whenever an upgrade to Modula-2 or to the SAGE system needs to be implemented.

Retrieval Access

Note that the libraries ClrHouse, NUCGEN, and General must be available before these commands are used.

These modules can be compiled singly or in any order, and they can be linked singly or in any order.

Commands to compile modules:

```
c:\m2lod\m2c mainmenu.mod/c
c:\m2lod\m2c hepnotes.mod/c
c:\m2lod\m2c hwnotes.mod/c
```

Commands to link modules (use after compilation):

```
c:\m2lod\m2l mainmenu
c:\m2lod\m2l hepnotes
c:\m2lod\m2l hwnotes
```

HEP Data Processing

If compilation is needed for a single library's DEFINITION MODULE, not only should it be compiled but each DEFINITION MODULE following in the list should also be compiled because of import dependencies among the libraries. Alternatively, one may review the import dependencies (by looking in the library source listings) for all libraries following the library of interest, compile the library of interest, and then selectively compile those following DEFINITION MODULES dependent on the library of interest. Having compiled DEFINITION MODULES, all corresponding IMPLEMENTATION MODULES would then need to be compiled in the same sequence; and then all modules which import from these compiled libraries would need to be compiled. Finally, linking should be done

for all modules which import from these compiled libraries.

If compilation is needed for a single library's IMPLEMENTATION MODULE without having compiled the corresponding DEFINITION MODULE, that IMPLEMENTATION MODULE may be compiled without regard to others in the list. In this same circumstance, it would not be necessary to compile modules which import from the library whose IMPLEMENTATION MODULE was just compiled; but linking should be done for these modules.

In each command list for compiling modules in the HEP data processing category, the modules can be compiled singly or in any order as needed.

If linking is needed for a single module and that module is to be linked as a base overlay, not only should that module be linked but each module following in the list should also be linked because of overlay structuring.

If linking is needed for a single module and that module is to be linked as a level 1 overlay node, not only should that module be linked but each module following in the list for that particular node should also be linked because of overlay structuring.

If linking is needed for a single module and that module is to be linked simply as a level 1 or level 2 overlay, or without overlay structuring, then that module may be linked without regard to others in the list.

Commands to compile libraries:

(the DEFINITION MODULE part of libraries)

```
c:\m2lod\m2c grf.def/c
c:\m2lod\m2c nucgen.def/c
c:\m2lod\m2c nucprint.def/c
c:\m2lod\m2c storagem.def/c
c:\m2lod\m2c retrieve.def/c
c:\m2lod\m2c breports.def/c
c:\m2lod\m2c calc.def/c
c:\m2lod\m2c nucfile.def/c
```

(the IMPLEMENTATION MODULE part of libraries)

```
c:\m2lod\m2c grf.mod/c
c:\m2lod\m2c nucgen.mod/c
c:\m2lod\m2c nucprint.mod/c
c:\m2lod\m2c storagem.mod/c
c:\m2lod\m2c retrieve.mod/c
c:\m2lod\m2c breports.mod/c
c:\m2lod\m2c calc.mod/c
c:\m2lod\m2c nucfile.mod/c
```

Commands to compile modules for maintaining DATABANK (use after compiling libraries):

```
c:\m2lod\m2c setup.mod/c
```

c:\m2lod\m2c nuclarr.mod/c

Commands to link modules for maintaining DATABANK (use after compilation):

c:\m2lod\m2l setup
c:\m2lod\m2l nuclarr

Commands to compile modules for retrieving DATABANK data (use after compiling libraries):

c:\m2lod\m2c descrhum.mod/c
c:\m2lod\m2c dochuman.mod/c
c:\m2lod\m2c adhochum.mod/c
c:\m2lod\m2c getrep.mod/c
c:\m2lod\m2c plthuman.mod/c
c:\m2lod\m2c hepagg.mod/c
c:\m2lod\m2c genascii.mod/c
c:\m2lod\m2c getfilep.mod/c
c:\m2lod\m2c savfilep.mod/c
c:\m2lod\m2c getrphep.mod
c:\m2lod\m2c getrpnog.mod

Commands to link modules for retrieving DATABANK data (use after compilation):

(linking the Retrieve library/module as a base overlay)

c:\m2lod\m2l retrieve/m

(linking level 1 overlays to the Retrieve module)

c:\m2lod\m2l dochuman/b retrieve
c:\m2lod\m2l adhochum/b retrieve
c:\m2lod\m2l getrep/b retrieve
c:\m2lod\m2l plthuman/b retrieve
c:\m2lod\m2l hepagg/b retrieve
c:\m2lod\m2l genascii/b retrieve
c:\m2lod\m2l getfilep/b retrieve
c:\m2lod\m2l savfilep/b retrieve

(linking DescrHum, as a level 1 overlay node, to the Retrieve module)

c:\m2lod\m2l descrhum/m/b retrieve

(linking level 2 overlays to the DescrHum module)

c:\m2lod\m2l getrphep/b descrhum
c:\m2lod\m2l getrpnog/b descrhum

If compilation is needed for a single library's DEFINITION MODULE, not only should it be compiled but each DEFINITION MODULE following in the list should also be compiled because of import dependencies among the libraries. Alternatively, one may review the import dependencies (by looking in the library source listings) for all libraries following the library of interest, compile the library of interest, and then selectively compile those following DEFINITION MODULES dependent on the library of interest. Having compiled DEFINITION MODULES, all corresponding IMPLEMENTATION MODULES would then need to be compiled in the same sequence; and then all modules which import from these compiled libraries would need to be compiled. Finally, linking should be done for all modules which import from these compiled libraries.

If compilation is needed for a single library's IMPLEMENTATION MODULE without having compiled the corresponding DEFINITION MODULE, that IMPLEMENTATION MODULE may be compiled without regard to others in the list. In this same circumstance, it would not be necessary to compile modules which import from the library whose IMPLEMENTATION MODULE was just compiled; but linking should be done for these modules.

In each command list for compiling modules in the HCF data processing category, the modules can be compiled singly or in any order as needed.

If linking is needed for a single module and that module is to be linked as a base overlay, not only should that module be linked but each module following in the list should also be linked because of overlay structuring.

If linking is needed for a single module and that module is to be linked as a level 1 overlay node, not only should that module be linked but each module following in the list for that particular node should also be linked because of overlay structuring.

If linking is needed for a single module and that module is to be linked simply as a level 1 or level 2 overlay, then that module may be linked without regard to others in the list.

Commands to compile libraries:

(the DEFINITION MODULE part of libraries)

```
c:\m2lod\m2c statlib.def/c
c:\m2lod\m2c hardfile.def/c
c:\m2lod\m2c clrhouse.def/c
c:\m2lod\m2c general.def/c
c:\m2lod\m2c storeman.def/c
c:\m2lod\m2c olayhw.def/c
c:\m2lod\m2c hwdispla.def/c
c:\m2lod\m2c hwreport.def/c
c:\m2lod\m2c hwfile.def/c
c:\m2lod\m2c graphics.def/c
c:\m2lod\m2c hwplot.def/c
c:\m2lod\m2c hardag.def/c
c:\m2lod\m2c adhochw.def/c
```

```
c:\m2lod\m2c descrhw.def/c
c:\m2lod\m2c sfileio.def/c
```

(the IMPLEMENTATION MODULE part of libraries)

```
c:\m2lod\m2c statlib.mod/c
c:\m2lod\m2c hardfile.mod/c
c:\m2lod\m2c clrhouse.mod/c
c:\m2lod\m2c general.mod/c
c:\m2lod\m2c storeman.mod/c
c:\m2lod\m2c olayhw.mod/c
c:\m2lod\m2c hwdispla.mod/c
c:\m2lod\m2c hwreport.mod/c
c:\m2lod\m2c hwfile.mod/c
c:\m2lod\m2c graphics.mod/c
c:\m2lod\m2c hwplot.mod/c
c:\m2lod\m2c hardag.mod/c
c:\m2lod\m2c adhochw.mod/c
c:\m2lod\m2c descrhw.mod/c
c:\m2lod\m2c sfileio.mod/c
```

Commands to compile modules for maintaining HARDWARE (use after compiling libraries):

```
c:\m2lod\m2c datantry.mod/c
c:\m2lod\m2c manualde.mod/c
c:\m2lod\m2c loaddb.mod/c
c:\m2lod\m2c sorcdump.mod/c
c:\m2lod\m2c dmanual.mod/c
c:\m2lod\m2c aggreg.mod/c
c:\m2lod\m2c edplants.mod/c
c:\m2lod\m2c eddocumt.mod/c
c:\m2lod\m2c hwtables.mod/c
c:\m2lod\m2c dbutil.mod/c
```

Commands to link modules for maintaining HARDWARE (use after compilation):

(linking the DataNtry module as a base overlay)

```
c:\m2lod\m2l datantry/m
```

(linking level 1 overlays to the DataNtry module)

```
c:\m2lod\m2l manualde/b datantry
c:\m2lod\m2l loaddb/b datantry
c:\m2lod\m2l sorcdump/b datantry
c:\m2lod\m2l dmanual/b datantry
c:\m2lod\m2l aggreg/b datantry
c:\m2lod\m2l edplants/b datantry
c:\m2lod\m2l eddocumt/b datantry
c:\m2lod\m2l hwtables/b datantry
c:\m2lod\m2l dbutil/b datantry
```

Commands to compile modules for retrieving HARDWARE data (use after compiling libraries):

```
c:\m2lod\m2c dhwsrch/c
c:\m2lod\m2c dhwrep/c
c:\m2lod\m2c dhwagg/c
c:\m2lod\m2c dhwfile/c
c:\m2lod\m2c dhwview/c
c:\m2lod\m2c dhwplot/c
c:\m2lod\m2c dhwsavef/c
c:\m2lod\m2c dhwgetf/c
c:\m2lod\m2c tailrdag/c

c:\m2lod\m2c dhardw/c

c:\m2lod\m2c ahwsrch/c
c:\m2lod\m2c ahwrep/c
c:\m2lod\m2c ahwagg/c
c:\m2lod\m2c ahwfile/c
c:\m2lod\m2c ahwview/c
c:\m2lod\m2c ahwplot/c
c:\m2lod\m2c ahwsavef/c
c:\m2lod\m2c ahwgetf/c

c:\m2lod\m2c ahardw/c

c:\m2lod\m2c dochardw/c

c:\m2lod\m2c retrhard/c
```

Commands to link modules for retrieving HARDWARE data (use after compilation):

(linking the RetrHard module as a base overlay)

```
c:\m2lod\m2l retrhard/m
```

(linking DHardw, as a level 1 overlay node, to the RetrHard module)

```
c:\m2lod\m2l dhardw/m/b retrhard
```

(linking level 2 overlays to the DHardw module)

```
c:\m2lod\m2l dhwsrch/b dhardw
c:\m2lod\m2l dhwrep/b dhardw
c:\m2lod\m2l dhwagg/b dhardw
c:\m2lod\m2l dhwfile/b dhardw
c:\m2lod\m2l dhwview/b dhardw
c:\m2lod\m2l dhwplot/b dhardw
c:\m2lod\m2l dhwsavef/b dhardw
c:\m2lod\m2l dhwgetf/b dhardw
c:\m2lod\m2l tailrdag/b dhardw
```

(linking AHardw, as a level 1 overlay node, to the RetrHard module)

```
c:\m2lod\m2l ahardw/m/b retrhard
```

(linking level 2 overlays to the AHardw module)

```
c:\m2lod\m2l ahwsrch/b ahardw
c:\m2lod\m2l ahwrep/b ahardw
c:\m2lod\m2l ahwagg/b ahardw
c:\m2lod\m2l ahwfile/b ahardw
c:\m2lod\m2l ahwview/b ahardw
c:\m2lod\m2l ahwplot/b ahardw
c:\m2lod\m2l ahwsavef/b ahardw
c:\m2lod\m2l ahwgetf/b ahardw
```

(linking a level 1 overlay to the RetrHard module)

```
c:\m2lod\m2l dochardw/b retrhard
```

DATA BASE COMPILATION

Through an option within THOR, the definition file for a data base is created by compiling its source file. The file name for a data base source file consists of the data base name with the suffix .SRC appended. The file name for a data base definition file consists of the data base name with the suffix .DFL appended. For example, 'DATABANK.SRC' is the source file and DATABANK.DFL is the definition file for the data base named DATABANK.

When relation schemas or data base forms are created or modified in THOR, these changes are made in the data base source file. But, the Modula-2 programs in the NUCLARR system access a data base by referencing its definition file. Therefore, the data base source file needs to be compiled and the resulting definition file made available for use in the NUCLARR system before such changes will be effective in NUCLARR.

DESCRIPTION OF SYSTEM PROGRAMS AND LIBRARIES

The programs and libraries of the NUCLARR system are briefly described in this section. The descriptions are organized according to previously defined categories: retrieval access, HEP data processing, and HCF data processing. A series of figures are presented to show the library dependencies associated with each of the NUCLARR system modules and libraries.

Sample charts showing high level flow structure for libraries and modules are available in Appendices B1, B2, C1, C2, and D. Listings of source code can be reviewed on screen by using a text editor such as Kedit. Use DOCUPROC to print source listings with a table of contents and indexing for ready reference. Samples of source code listings generated by DOCUPROC can be found in Appendices D, G, and H.

Appendix A states where to obtain descriptions of libraries accessed by the NUCLARR system, but which are not part of NUCLARR.

Retrieval Access Programs

The three programs described here are used to set up retrieval access to data resident in the NUCLARR system. This access may be to the HEP data in the data base DATABANK or to the HCF data in the data base HARDWARE. Batch files are used in concert with program files to accomplish this processing.

See Figure 1 to review the relationships involved between those batch files and program executable files used to provide this retrieval access. See the batch file listings in Appendix D to review the DOS commands used for this access processing.

Retrieval access is initiated from the DOS root directory by requesting execution of the batch file NUCLARR.BAT. When retrieval access is terminated, the user is returned to the DOS root directory.

The program MainMenu presents the menu for selecting retrieval access options. Access to data in one of the data bases may be requested. Access to information on aggregation processing for either HEP data or HCF data may be requested. Based on the option selected, an appropriate batch file is created to provide the desired access or to leave the NUCLARR system. NUCLARR library dependencies are shown in Figure 7.

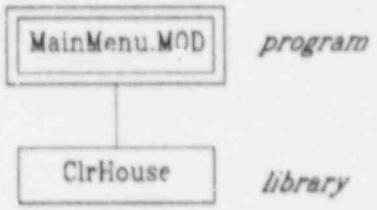


Figure 7. Library dependencies for the program MainMenu.

The program HEPNotes presents information on aggregation processing for HEP data. This information may be viewed, one screen at a time, until the option to return to the previous selection menu is taken. NUCLARR library dependencies are shown in Figure 8.

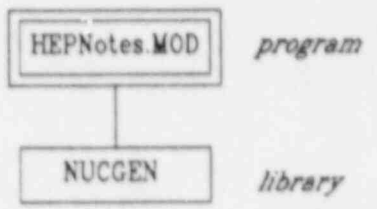


Figure 8. Library dependencies for the program HEPNotes.

The program HWNotes presents information on aggregation processing for HCF data. This information may be viewed, one screen at a time, until the option to return to the

previous selection menu is taken. NUCLARR library dependencies are shown in Figure 9.

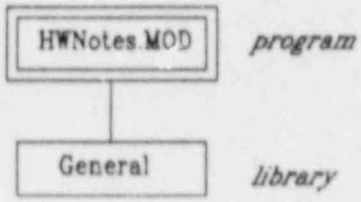


Figure 9. Library dependencies for the program HWNotes.

HEP Data Processing Libraries

This set of libraries provides procedures used by the programs which process HEP data. Each library consists of procedures which are related to one another in terms of functionality or other features. Some procedures from these libraries are used many times by various programs, modules, or other libraries importing them. In some cases, a procedure is imported and used only once. Regardless of how much they are used, procedures are grouped into these libraries for convenience of maintenance, to provide for needed overlay structuring, to minimize the amount of source code, and to minimize memory required during execution.

The library NUCGEN provides general purpose procedures for accessing and combining data elements. There are no NUCLARR library dependencies, as shown in Figure 10.

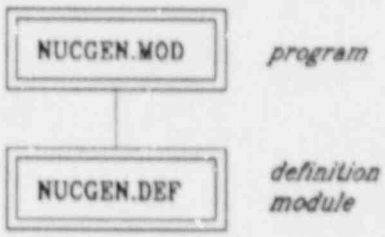


Figure 10. No library dependencies for the program NUCGEN.

The library NUCPRINT provides procedures for displaying data and its associated descriptions. NUCLARR library dependencies are shown in Figure 11.

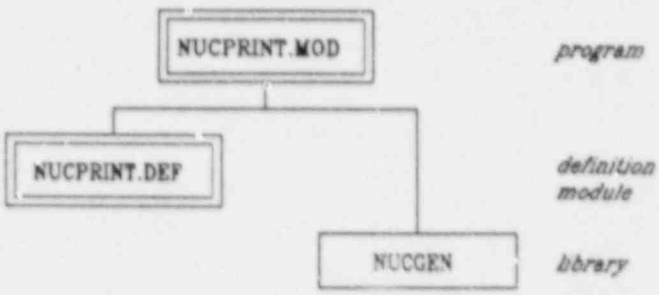


Figure 11. Library dependencies for the program NUCPRINT.

The library StorageM provides for storage buffer manipulation. StorageM procedures are used to add values to, retrieve values from, and clear a predefined storage buffer. NUCLARR library dependencies are shown in Figure 12.

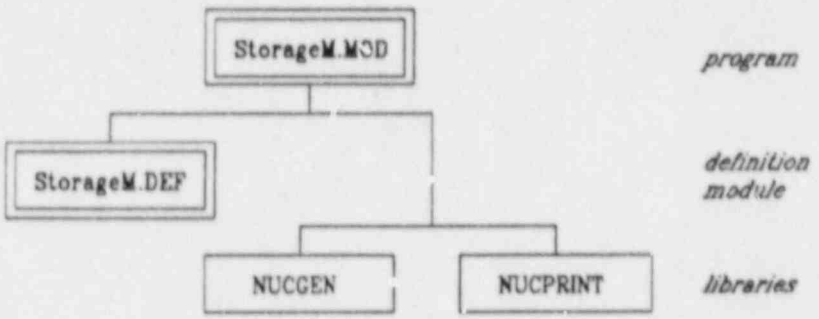


Figure 12. Library dependencies for the program StorageM.

The library Retrieve is used both as a program and as a library. The reason for this double duty is to allow for the passing of needed variables into the overlay structure associated with the program Retrieve. See the description of the program Retrieve for more information.

The library BReports provides procedures for generating reports on records referenced in the storage buffer. NUCLARR library dependencies are shown in Figure 13.

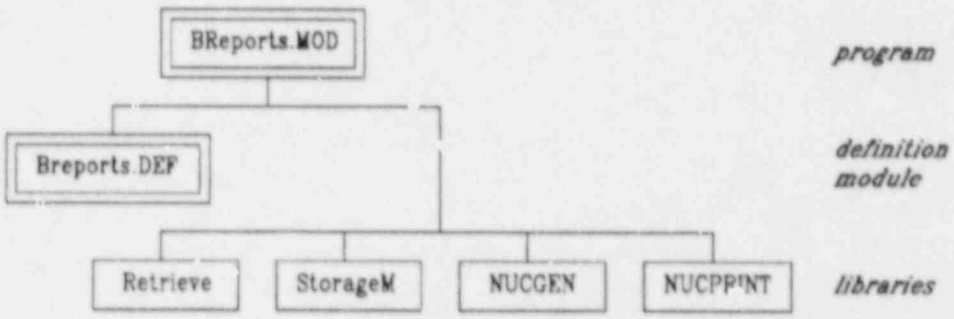


Figure 13. Library dependencies for the program BReports.

The library Calc provides procedures for calculating HEP and associated values. NUCLARR library dependencies are shown in Figure 14.

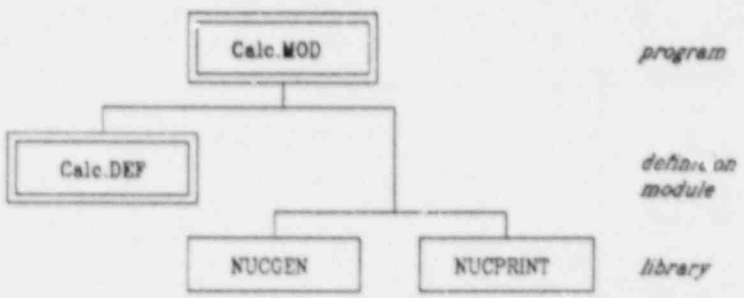


Figure 14. Library dependencies for the program Calc.

The library Grf provides procedures for plotting. There are no NUCLARR library dependencies, as shown in Figure 15.

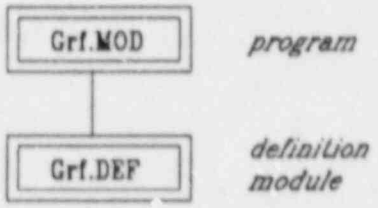


Figure 15. No library dependencies for the program Grf.

The library NUCFILE provides procedures for loading specified data from the data base to an ASCII file. NUCLARR library dependencies are shown in Figure 16.

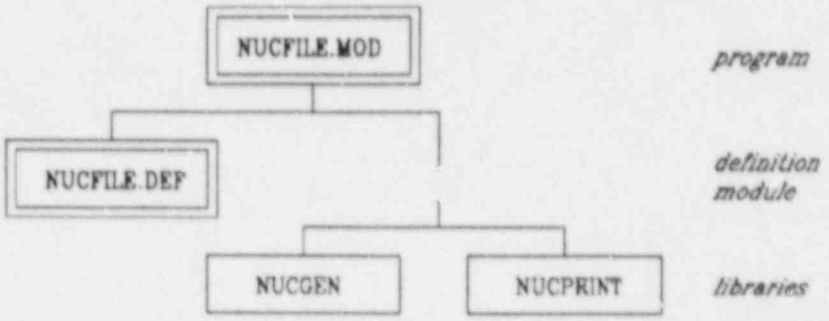


Figure 16. Library dependencies for the program NUCFILE.

HEP Data Processing Programs

There are three programs used for HEP data processing. The two programs Setup and Nuclarr are used to maintain the DATABANK data base, while the program Retrieve is for retrieving data from DATABANK.

The program Setup is used to maintain the qualifying records of the DATABANK data base. Qualifying records are those records used to evaluate the validity of data being entered. Examples of qualifying records are plant code (facility identification) records and matrix definition records. NUCLARR library dependencies are shown in Figure 17.

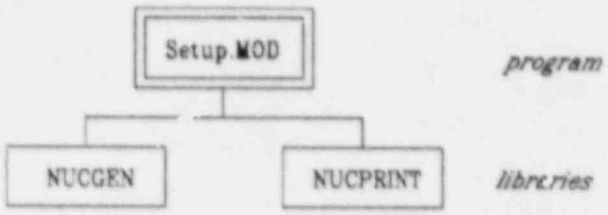


Figure 17. Library dependencies for the program Setup.

The program Nuclarr is used to enter and modify data in the DATABANK data base as well as to perform other maintenance chores such as HEP calculations and generation of various data reports. NUCLARR library dependencies are shown in Figure 18.

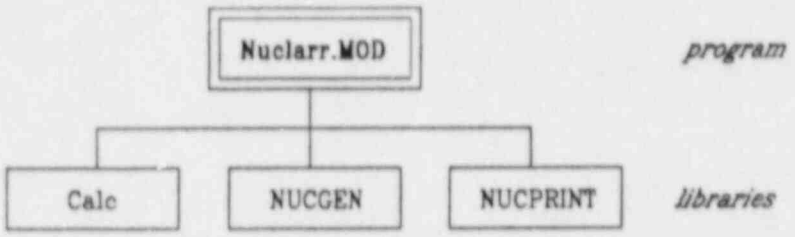


Figure 18. Library dependencies for the program Nuclarr.

The program Retrieve makes DATABANK data available for review but not for alteration. This program uses an overlay structure (see Figure 2) and is also written as a library so that needed variables can be passed into the overlays. The following module descriptions are for those modules included in Retrieve's overlay structure. All these modules, except for GetRpHEP and GetRpNoG, are accessed by selection from the menu presented in Retrieve. The module DocHuman is the only one which does not make use of the search storage buffer for its processing. The search storage buffer is used to accumulate HEP data record references located by one of the two search processes. This buffer is cleared by explicit user request or by exiting from Retrieve. NUCLARR library dependencies are shown in Figure 19.

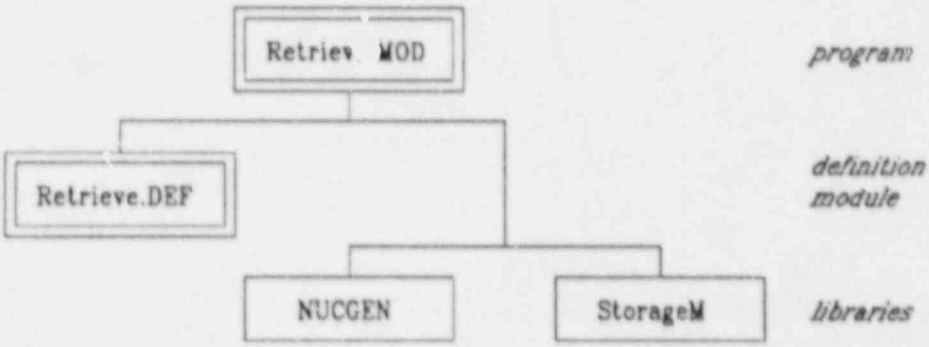


Figure 19. Library dependencies for the program Retrieve.

The module DescrHum performs a descriptive search for HEP data. To perform this search processing, the user selects a taxonomy level, job classification, and plant ID; these selections determine the matrix number needed. Next, the matrix row and column are chosen by the user. Now, records which meet the criteria determined by the user's selections are located and then referenced in the search storage buffer. Through the overlay structure, DescrHum provides access to the three modules GetRep, GetRpHEP, and GetRpNoG, for reporting on data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 20.

The module GetRpHEP provides for reporting on HEP records referenced in the search storage buffer. Aggregated HEP values from the data base will be included in any report

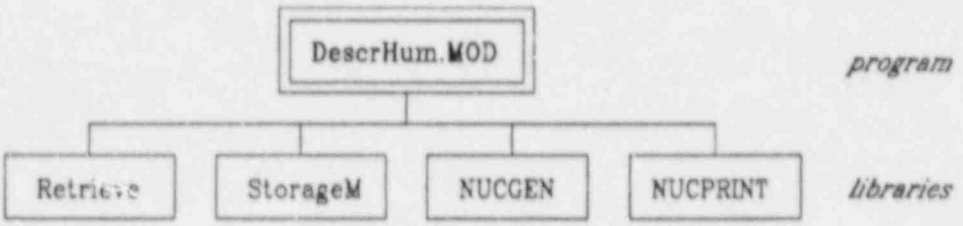


Figure 20. Library dependencies for the program DescrHum.

from this module. These data base values will be at the task level, cell level, and functional group level. GetRpHEP is accessed from DescrHum. NUCLARR library dependencies are shown in Figure 21.

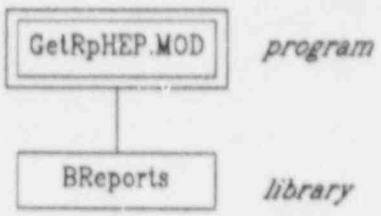


Figure 21. Library dependencies for the program GetRpHEP.

The module GetRpNoG provides for reporting on HEP records referenced in the search storage buffer. Aggregated HEP values from the data base will be included in any report from this module. These data base values will be at the task level and cell level, but not at the functional group level. GetRpNoG is accessed from DescrHum. NUCLARR library dependencies are shown in Figure 22.

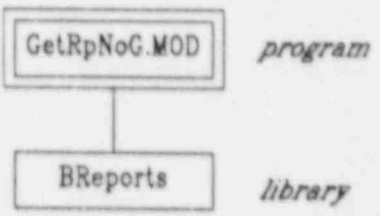


Figure 22. Library dependencies for the program GetRpNoG.

The module DocHuman allows the user to review identification of documents referenced in the NUCLARR system for HEP data. NUCLARR library dependencies are shown in Figure 23.

The module AdHocHum performs an ad hoc search for HEP data. The user selects parameter values which are used to establish search criteria for this search processing. These search restriction selections are specified on a single form with the aid of an elaborate set of help forms. Once the request is made to locate data records, the specified criteria are cross-checked for consistency and validity. The user is prompted for necessary changes to selections before data records are located. Records which meet the criteria determined by the user's selections are located and then referenced in the search storage

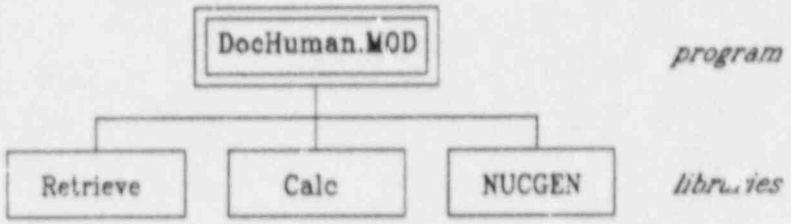


Figure 23. Library dependencies for the program DocHuman.

buffer. A variable is used for message handling so that a proper message appears when the search criteria form is presented. NUCLARR library dependencies are shown in Figure 24.

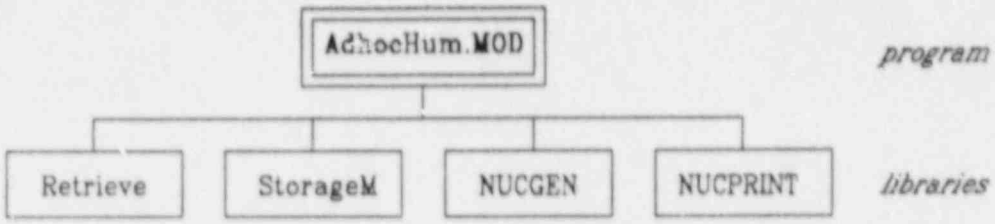


Figure 24. Library dependencies for the program AdHocHum.

The module GetRep provides for reporting on HEP records referenced in the search storage buffer. No aggregated HEP values from the data base will be included in a report from this module. GetRep can be accessed either from Retrieve or from DescrHum. NUCLARR library dependencies are shown in Figure 25.

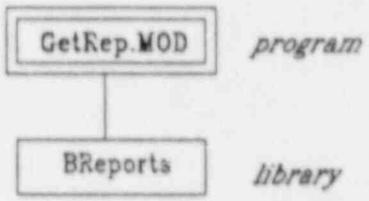


Figure 25. Library dependencies for the program GetRep.

The module PltHuman is used to plot HEP data which are referenced in the search storage buffer. This set of data must be aggregated before plots can be generated since the plots include aggregation values. NUCLARR library dependencies are shown in Figure 26.

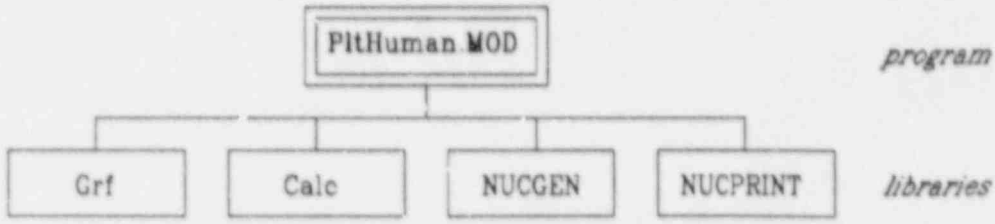


Figure 26. Library dependencies for the program PltHuman.

The module HEPAgg provides for aggregation of HEP data which are referenced in the search storage buffer. An aggregation is done for each task, and then an aggregation is done using all the task HEP values just calculated, creating an HEP for the entire set of data referenced. NUCLARR library dependencies are shown in Figure 27.

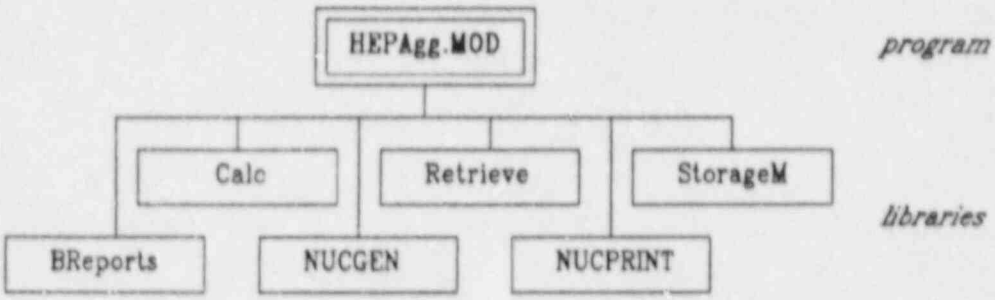


Figure 27. Library dependencies for the program HEPAgg.

The module GenASCII generates an ASCII file of data taken from HEP data records. This ASCII file can be created by using all HEP data records in DATABANK or by using just that data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 28.

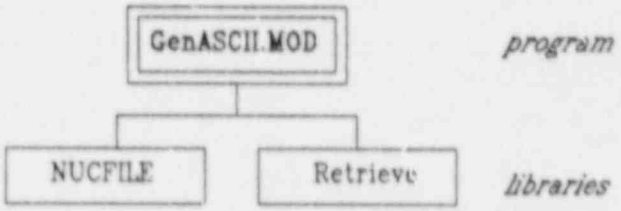


Figure 28. Library dependencies for the program GenASCII.

The module GetFileP moves a copy of the contents of a DOS file into the search storage buffer after first clearing the buffer. If the file is other than one which was created by SavFileP, the results of then using the search storage buffer will be unpredictable. NUCLARR library dependencies are shown in Figure 29.

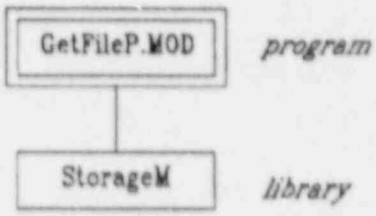


Figure 29. Library dependencies for the program GetFileP.

The module SavFileP moves a copy of the contents of the search storage buffer into a DOS file. If the requested DOS file already exists, it is overwritten with the buffer contents. NUCLARR library dependencies are shown in Figure 30.

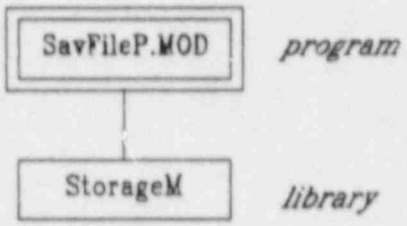


Figure 30. Library dependencies for the program SavFileP.

HCF Data Processing Libraries

This set of libraries provides procedures used by the programs which process HCF data. Each library consists of procedures which are related to one another in terms of functionality or other features. Some procedures from these libraries are used many times by various programs, modules, or other libraries importing them. In some cases, a procedure is imported and used only once. Regardless of how much they are used, procedures are grouped into these libraries for convenience of maintenance, to provide for needed overlay structuring, to minimize the amount of source code, and to minimize memory required during execution.

The library StatLib provides statistical calculation procedures for data aggregation processing. There are no NUCLARR library dependencies, as shown in Figure 31.

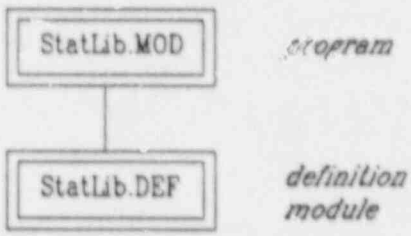


Figure 31. No library dependencies for the program StatLib.

The library HardFile provides procedures for copying hardware data from an ASCII file into the data base named HARDWARE. NUCLARR library dependencies are shown in Figure 32.

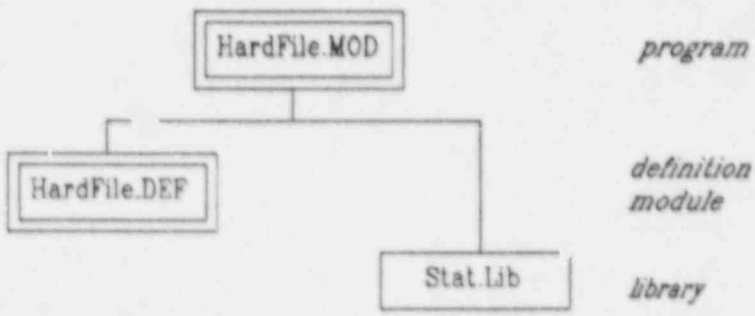


Figure 32. Library dependencies for the program HardFile.

The library ClrHouse provides the information needed to make contact with the data clearinghouse. There are no NUCLARR library dependencies, as shown in Figure 33.

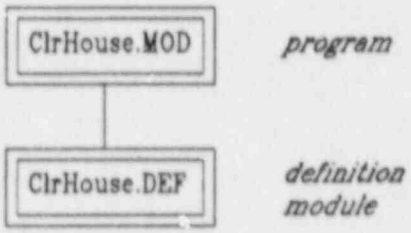


Figure 33. No library dependencies for the program ClrHouse.

The library General provides general purpose procedures for accessing and combining data elements. There are no NUCLARR library dependencies, as shown in Figure 34.

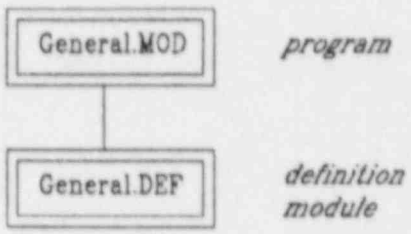


Figure 34. No library dependencies for the program General.

The library StoreMan provides for storage buffer manipulation. StoreMan procedures are used to add values to, retrieve values from, and clear a predefined storage buffer. There are no NUCLARR library dependencies, as shown in Figure 35.

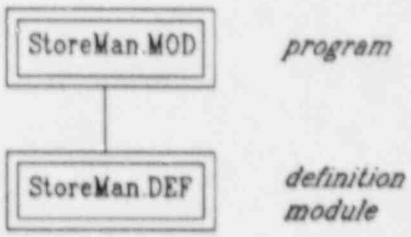


Figure 35. No library dependencies for the program StoreMan.

The library OlayHw provides procedures which monitor overlay processing. There are no NUCLARR library dependencies, as shown in Figure 36.

The library HWDispla provides procedures for on-screen viewing of hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 37.

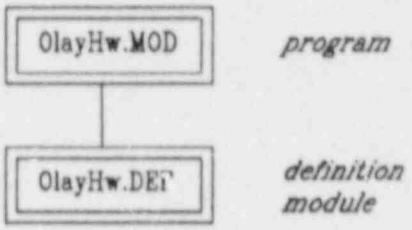


Figure 36. No library dependencies for the program OlayHw.

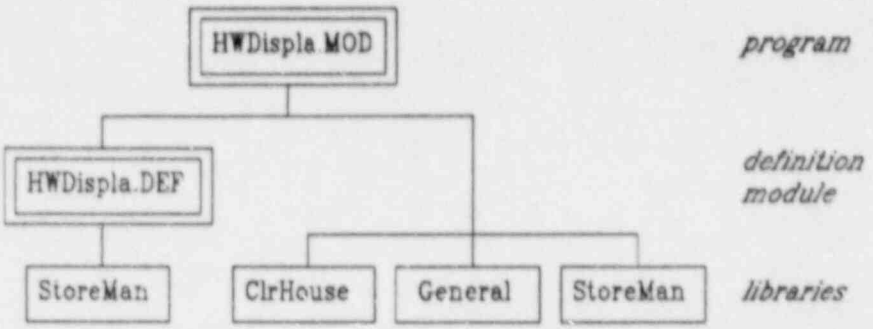


Figure 37. Library dependencies for the program HWDispla.

The library HWReport provides procedures for reporting on hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 38.

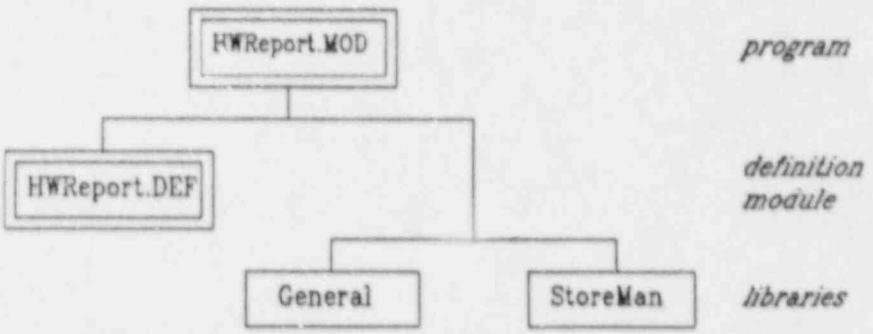


Figure 38. Library dependencies for the program HWReport.

The library HWFile provides procedures for creating an ASCII file from hardware event data. NUCLARR library dependencies are shown in Figure 39.

The library Graphics provides procedures for plotting. There are no NUCLARR library dependencies, as shown in Figure 40.

The library HWPlot provides procedures which create plots using data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 41.

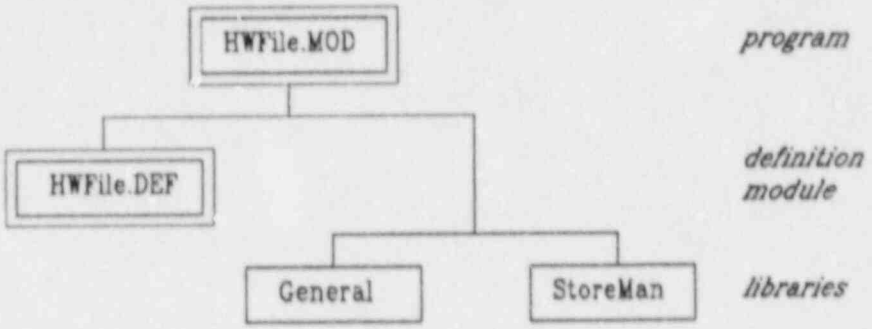


Figure 39. Library dependencies for the program HWFile.

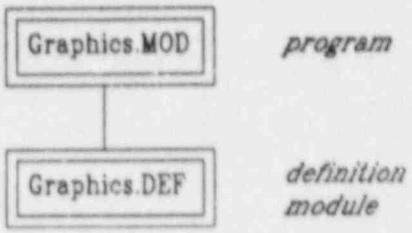


Figure 40. N. library dependencies for the program Graphics.

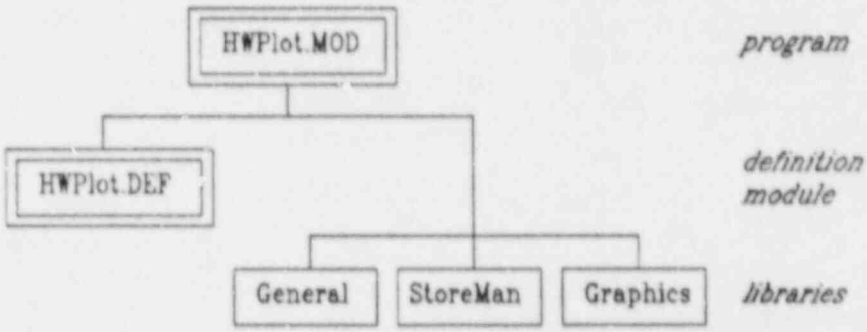


Figure 41. Library dependencies for the program HWPlot.

The library HardAg provides procedures for data aggregation processing. NUCLARR library dependencies are shown in Figure 42.

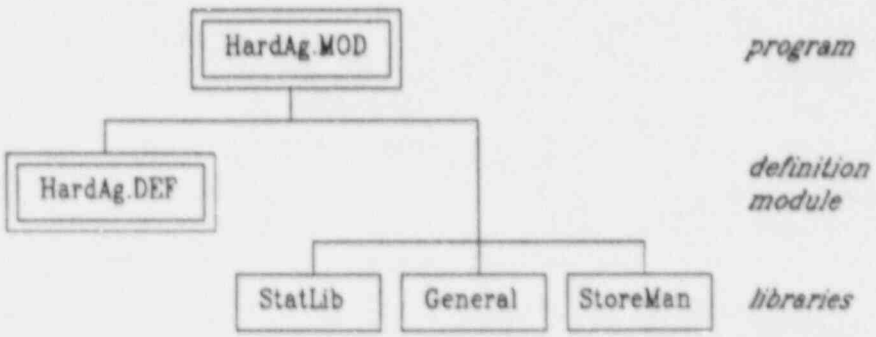


Figure 42. Library dependencies for the program HardAg.

The library AdHocHw provides procedures for ad hoc searching of hardware event records. Found records are then referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 43.

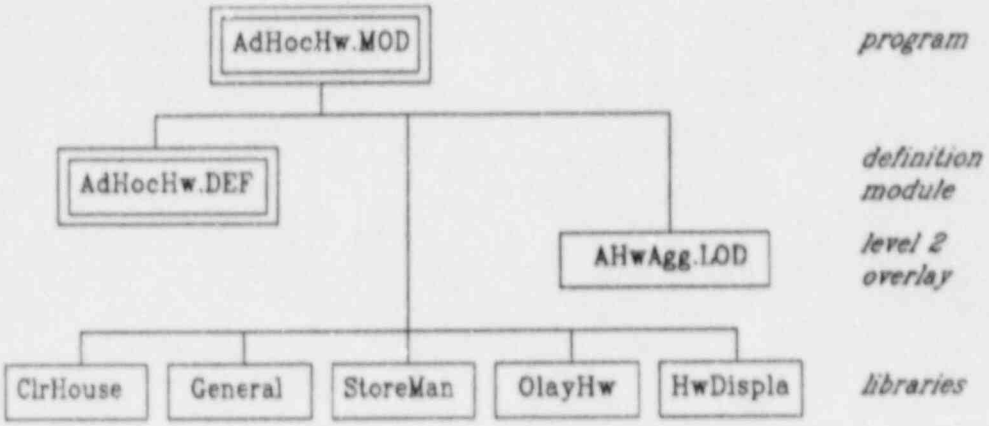


Figure 43. Library dependencies for the program AdHocHw.

The library DescrHw provides procedures for hardware event descriptive searching. Found records are then referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 44.

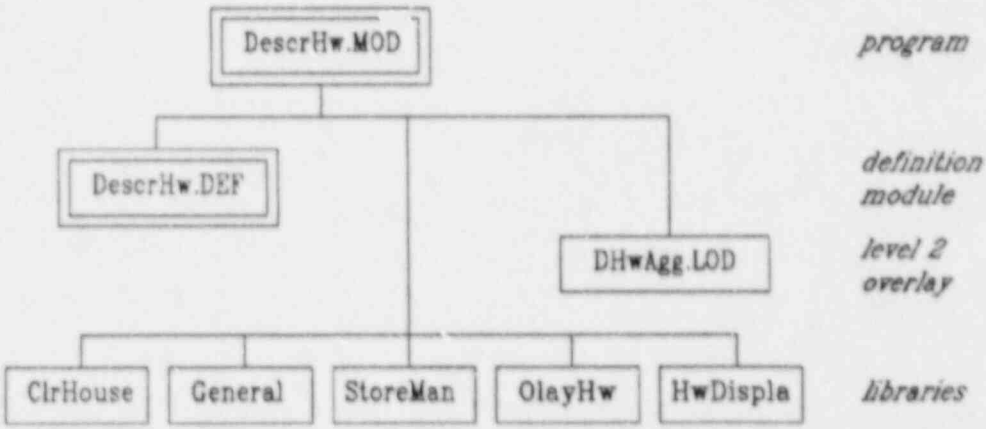


Figure 44. Library dependencies for the program DescrHw.

The library SFileIO provides procedures for copying data record references between a DOS file and the search storage buffer. NUCLARR library dependencies are shown in Figure 45.

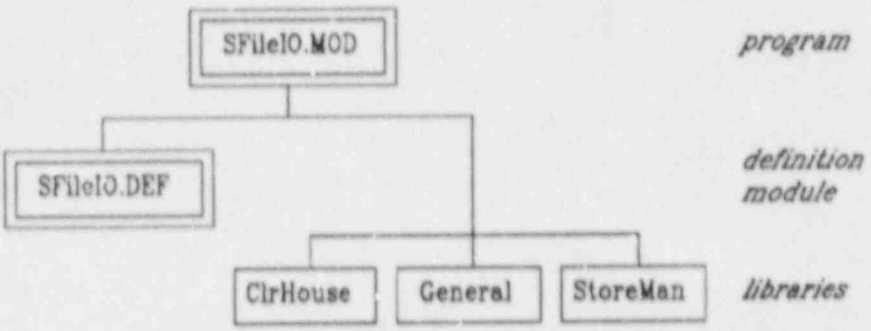


Figure 45. Library dependencies for the program SFileIO.

HCF Data Processing Programs

There are two programs used for HCF data processing. The program DataNtry is used to maintain the HARDWARE data base, while the program RetrHard is for retrieving data from HARDWARE.

The program DataNtry is used to enter and modify data in the HARDWARE data base and to maintain the qualifying records of HARDWARE. Qualifying records are those records used to evaluate the validity of data being entered. Examples of qualifying records are records for plant codes (facility identification), component codes, component design codes, failure mode codes, normal state codes, and application codes. DataNtry is also used to perform other maintenance chores such as data aggregation calculations and generation of various data reports. This program uses an overlay structure (see Figure 3). The following nine module descriptions (ManualDE through DBUtil) are for those modules included in DataNtry's overlay structure. These modules are all accessed by selection from the menu presented in DataNtry. NUCLARR library dependencies are shown in Figure 46.

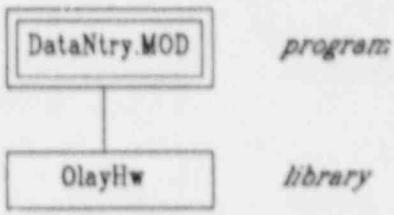


Figure 46. Library dependencies for the program DataNtry.

The module ManualDE is used to enter and modify data in the HARDWARE relation named Source. Calculations associated with the data are done during add and modify operations. NUCLARR library dependencies are shown in Figure 47.

The module LoadDB copies HCF data from an ASCII file into the HARDWARE relation named Source. NUCLARR library dependencies are shown in Figure 48.

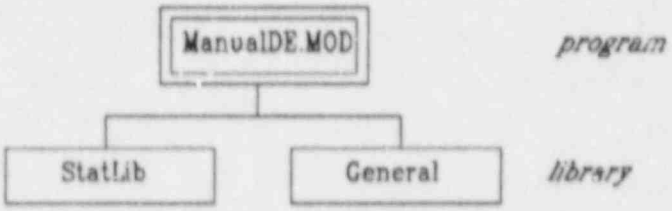


Figure 47. Library dependencies for the program ManualDE.

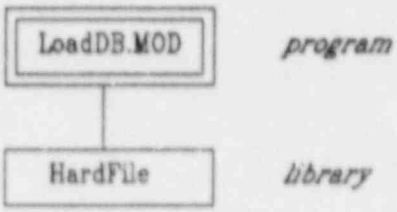


Figure 48. Library dependencies for the program LoadDB.

The module Aggreg aggregates HCF data and calculates the associated probabilities. This aggregation may be done for all basic events or for only those events not yet aggregated. NUCLARR library dependencies are shown in Figure 49.

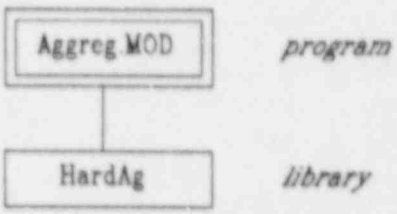


Figure 49. Library dependencies for the program Aggreg.

The module SorcDump controls the dumping of records from the relation named Source to a disk file. NUCLARR library dependencies are shown in Figure 50.

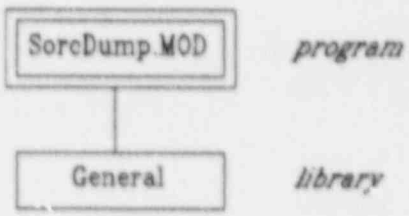


Figure 50. Library dependencies for the program SorcDump.

The module DManual controls the generation of raw data reports based on records in the relation named Source. NUCLARR library dependencies are shown in Figure 51.

The module EdDocumt provides for editing and reporting on documents referenced in

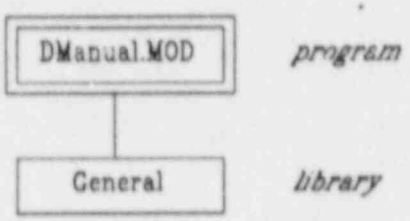


Figure 51. Library dependencies for the program DManual.

the NUCLARR system for HCF data. NUCLARR library dependencies are shown in Figure 52.

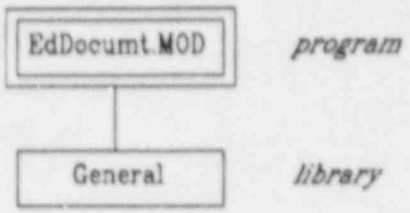


Figure 52. Library dependencies for the program EdDocumt.

The module EdPlants provides for editing and reporting on plant code (facility identification) records. NUCLARR library dependencies are shown in Figure 53.

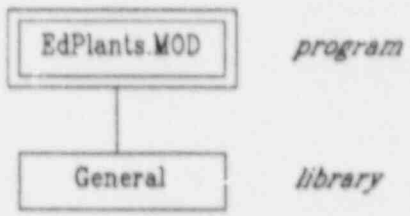


Figure 53. Library dependencies for the program EdPlants.

The module HWTables provides for editing and reporting on HARDWARE qualifying records (other than plant codes). NUCLARR library dependencies are shown in Figure 54.

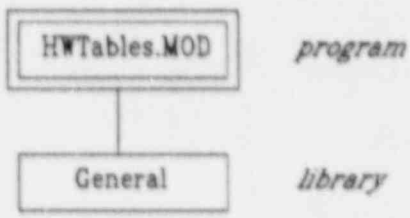


Figure 54. Library dependencies for the program HWTables.

The module DBUtil is used for miscellaneous data base utility functions such as counting the number of records in data base relations and deleting all records in the AgStat

relation. There are no NUCLARR library dependencies.

The program RetrHard makes HARDWARE data available for review but not for alteration. This program uses an overlay structure (see Figures 4, 5, and 6). The following module descriptions are for those modules included in RetrHard's overlay structure. The three level 1 overlay modules, DHardw, AHardw, and DocHardw, are accessed by selection from the menu presented in RetrHard. The other modules are level 2 overlay modules which are accessed from DHardw or from AHardw. NUCLARR library dependencies are shown in Figure 55.

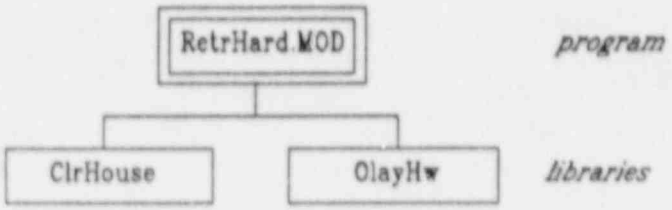


Figure 55. Library dependencies for the program RetrHard.

The module DHardw provides for descriptive search processing and access to the search storage buffer. The search storage buffer is used to accumulate HCF data record references located by the search process. This buffer is cleared at entry to DHardw and by explicit user request while search criteria are being established. The following nine module descriptions (DHwGetF through DHwSaveF) are for those modules included in DHardw's overlay structure (see Figure 5). These modules are all accessed by selection from the menu presented in DHardw. NUCLARR library dependencies are shown in Figure 56.

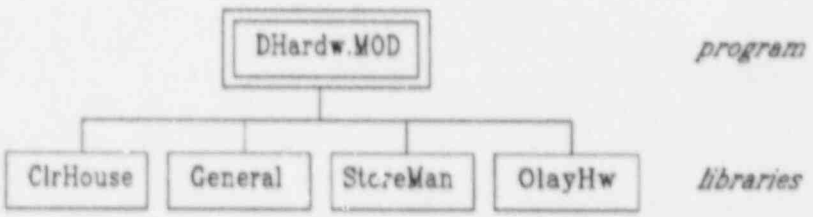


Figure 56. Library dependencies for the program DHardw.

The module DHwGetF moves a copy of the contents of a DOS file into the search storage buffer after first clearing the buffer. If the file is other than one which was created by DHwSaveF or AHwSaveF, the results of then using the search storage buffer will be unpredictable. NUCLARR library dependencies are shown in Figure 57.

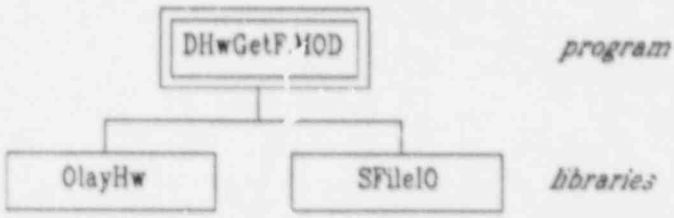


Figure 57. Library dependencies for the program DHwGetF.

The module DHwSrch performs a descriptive search for HCF data. To perform this search processing, the user selects a hardware category, hardware component, hardware component design, hardware event failure mode, hardware event normal state, and application. Records which meet the criteria determined by the user's selections are located and then referenced in the search storage buffer. Through the overlay structure, DHwSrch provides access to the module DHwAgg for aggregation of that data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 58.

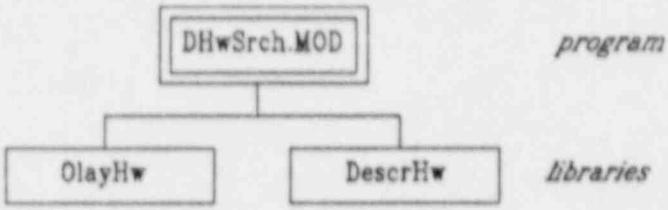


Figure 58. Library dependencies for the program DHwSrch.

The module TailrdAg provides a tailored search and aggregation of specified failure and exposure data combinations. This is an example of a customized search tailored to a specific class of data or type of analysis. Records which meet the criteria determined by the user's selections are located and then referenced in the search storage buffer. Through the overlay structure, TailrdAg provides access to the module DHwAgg for aggregation of that data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 59.

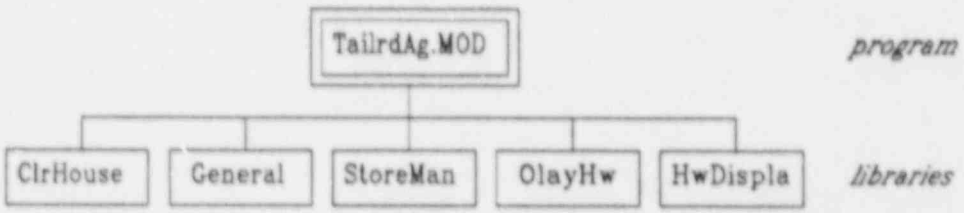


Figure 59. Library dependencies for the program TailrdAg.

The module DHwView provides on-screen viewing of those hardware event records currently referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 60.

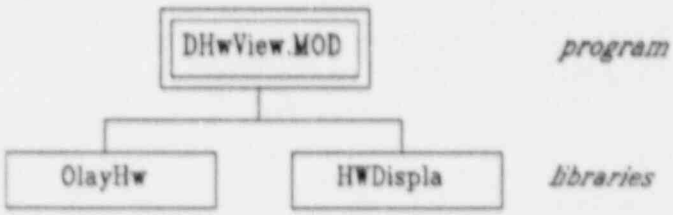


Figure 60. Library dependencies for the program DHwView.

The module DHwAgg aggregates data from hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 61.

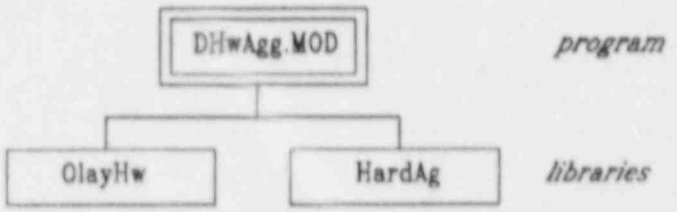


Figure 61. Library dependencies for the program DHwAgg.

The module DHwRep provides reports on hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 62.

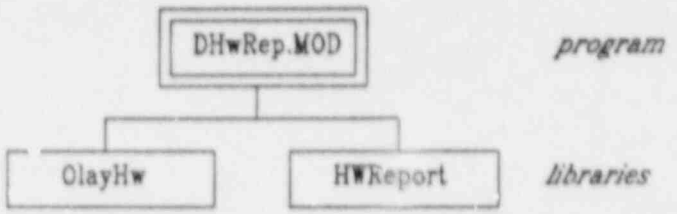


Figure 62. Library dependencies for the program DHwRep.

The module DHwPlot is used to plot probability data from hardware events which are referenced in the search storage buffer. This set of data must be aggregated before plots can be generated because the plots include aggregation values. NUCLARR library dependencies are shown in Figure 63.

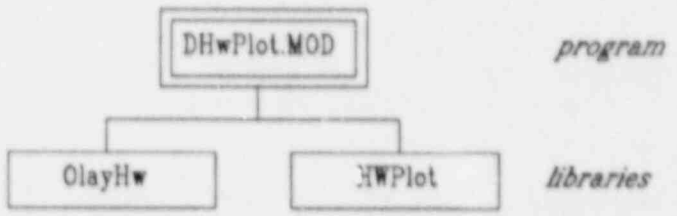


Figure 63. Library dependencies for the program DHwPlot.

The module DHwFile creates an ASCII file of hardware event data based on that hardware event data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 64.

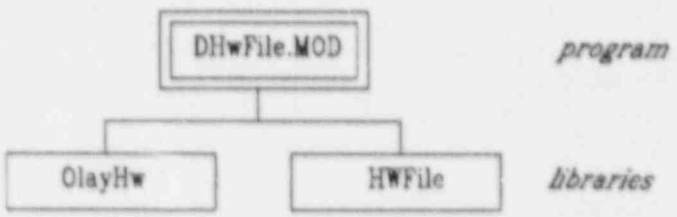


Figure 64. Library dependencies for the program DHwFile.

The module DHwSaveF moves a copy of the contents of the search storage buffer into a DOS file. If the requested DOS file already exists, it is overwritten with the buffer contents. NUCLARR library dependencies are shown in Figure 65.

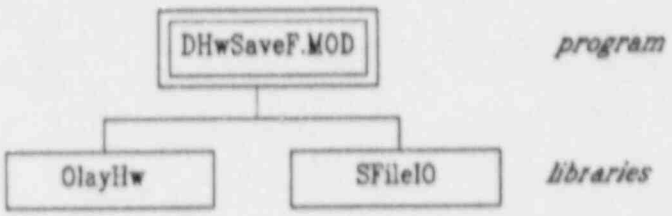


Figure 65. Library dependencies for the program DHwSaveF.

The module AHardw provides for ad hoc search processing and access to the search storage buffer. The search storage buffer is used to accumulate HCF data record references located by the search process. This buffer is cleared at entry to AHardw and by explicit user request while search criteria are being established. The following eight module descriptions (AHwGetF through AHwSaveF) are for those modules included in AHardw's overlay structure (see Figure 6). These modules are all accessed by selection from the menu presented in AHardw. NUCLARR library dependencies are shown in Figure 66.

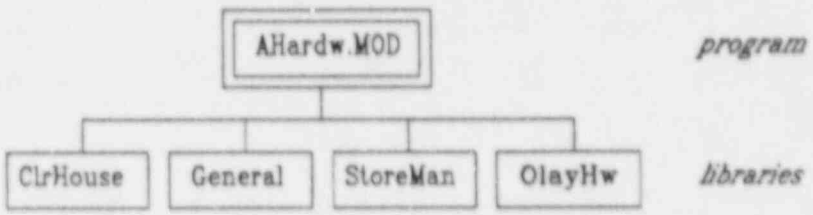


Figure 66. Library dependencies for the program AHardw.

The module AHwGetF moves a copy of the contents of a DOS file into the search storage buffer after first clearing the buffer. If the file is other than one which was created by AHwSaveF or DHwSaveF, the results of then using the search storage buffer will be unpredictable. NUCLARR library dependencies are shown in Figure 67.

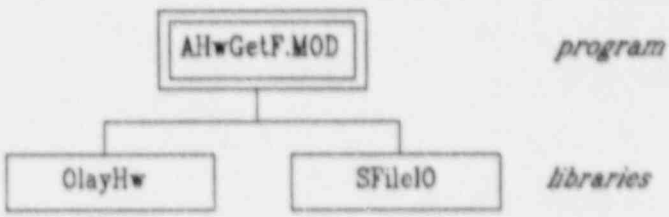


Figure 67. Library dependencies for the program AHwGetF.

The module AHwSrch performs an ad hoc search for HCF data. The user selects parameter values which are used to establish search criteria for this search processing. These search restriction selections are specified on a single form with the aid of an elaborate set of help forms. Once the request is made to locate data records, the specified criteria are cross-checked for consistency and validity. The user is prompted for necessary

changes to his input before data records are located. Records which meet the criteria determined by the user's selections are located and then referenced in the search storage buffer. A variable is used for message handling so that the proper message appears when the search criteria form is presented. Through the overl[illegible]ucture, AHwSrch provides access to the module AHwAgg for aggregation of that data re[illegible]ced in the search storage buffer. NUCLARR library dependencies are shown in Figure 68.

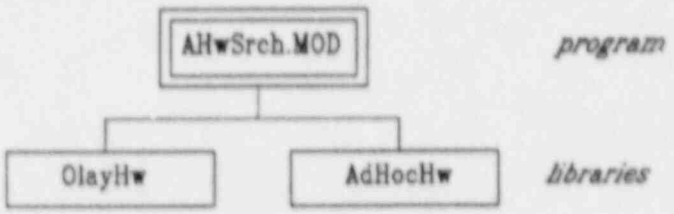


Figure 68. Library dependencies for the program AHwSrch.

The module AHwView provides on-screen viewing of those hardware event records currently referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 69.

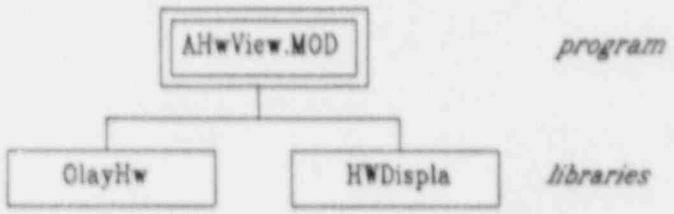


Figure 69. Library dependencies for the program AHwView.

The module AHwAgg aggregates data from hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 70.

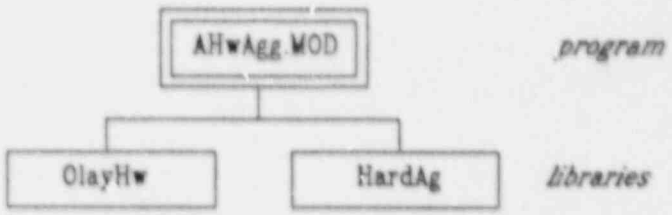


Figure 70. Library dependencies for the program AHwAgg.

The module AHwRep reports on hardware event records referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 71.

The module AHwPlot is used to plot probability data from hardware events which are referenced in the search storage buffer. This set of data must be aggregated before plots can be generated because the plots include aggregation values. NUCLARR library dependencies are shown in Figure 72.

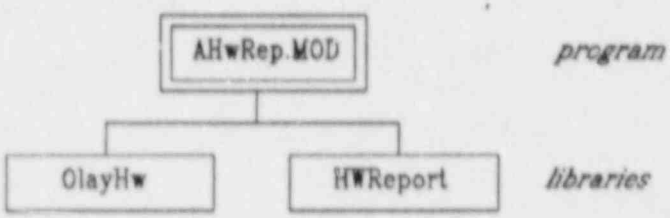


Figure 71. Library dependencies for the program AHwRep.

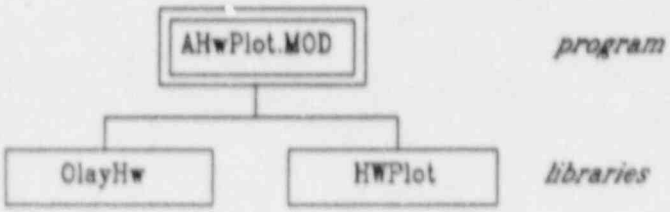


Figure 72. Library dependencies for the program AHwPlot.

The module AHwFile creates an ASCII file of hardware event data based on that hardware event data referenced in the search storage buffer. NUCLARR library dependencies are shown in Figure 73.

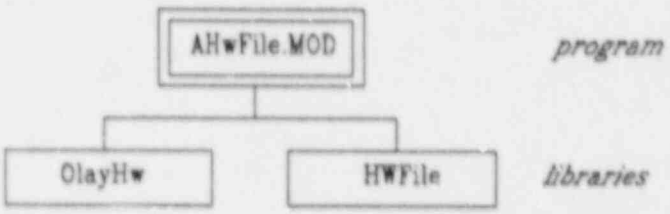


Figure 73. Library dependencies for the program AHwFile.

The module AHwSaveF moves a copy of the contents of the search storage buffer into a DOS file. If the requested DOS file already exists, it is overwritten with the buffer contents. NUCLARR library dependencies are shown in Figure 74.

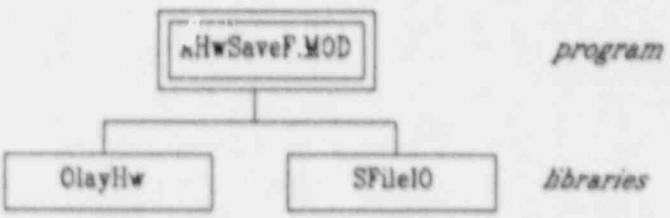


Figure 74. Library dependencies for the program AHwSaveF.

The module DocHardw allows the user to review identification of documents referenced in the NUCLARR system for HCF data. NUCLARR library dependencies are shown in Figure 75.

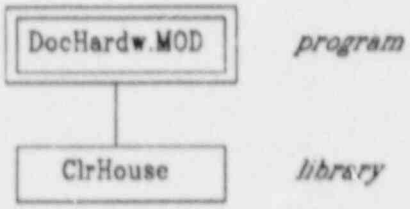


Figure 75. Library dependencies for the program DocHardw.

HOW TO USE THIS GUIDE

The purpose of this programmer's guide is to help an experienced software engineer understand how various products are used together to develop and maintain NUCLARR. Although this guide may help a less experienced software engineer to better understand the products used, it is not intended to supplant the technical documentation associated with these products.

Overview of the Guide

The preceding sections of this guide give an overview of the information a software engineer will need to understand in order to maintain the software of the NUCLARR system. The associated tools and products needed are also specified. In addition to the overview, a significant amount of in-depth information is included and, throughout this guide, references are found to material in the appendices which follow. Furthermore, those software engineers maintaining the NUCLARR system software will need to have access to documentation for the various tools and products identified in this guide.

This volume, along with the other volumes of this series, should be used in training a software engineer newly assigned to maintenance of NUCLARR. For training purposes, the sections of this volume should be read in the order written. The content of this particular volume will be of limited use to anyone other than a software engineer.

The most common usage of this guide will be that of locating specific modules and reviewing the design and the implementation structures of NUCLARR while resolving execution problems or implementing enhancements to the software. An allied usage would be that of determining appropriate compile and link sequences after making software changes. For all of these purposes, only sections or appendices applicable to the issue(s) of interest need be consulted.

The remainder of this section describes the content of the sections and appendices of this programmer's guide.

The PROGRAMMING ENVIRONMENT section specifies the programming language, system development tools and products, and hardware needed for system maintenance. The DATA STRUCTURE section describes the data base structure used and its associated mechanisms. The SOFTWARE ORGANIZATION section defines software terminology used and reviews organizational methodology and structures implemented in the NUCLARR system.

The section named PROGRAM COMPILATION AND LINKING covers general issues regarding the compile and link processes for software elements in the NUCLARR system. Next, the explicit compile and link commands to be used are listed.

The DATA BASE COMPILATION section explains issues concerning the compiling of the NUCLARR data bases and associated terminology.

In the section named DESCRIPTION OF SYSTEM PROGRAMS AND LIBRARIES, descriptions are provided for those software units programmed for the NUCLARR system. These descriptions are not detailed because their main purpose is to lead the programmer to the proper source code listing for understanding the details of a code's function. When it is expected that further help is needed, some of these descriptions do include references to methodology and techniques used or other additional details.

Appendix A provides a key used to locate documentation for those libraries accessed by but not part of the code programmed for the NUCLARR system.

Appendices B1, B2, C1, and C2 all contain high level flow charts for procedures programmed for the NUCLARR system. These charts provide an overview of what each procedure charted is doing and how it is done; consequently, these charts can serve as road maps to the source code listings when the software engineer is searching for particular elements of a procedure. These four appendices also provide procedure reference lists and a list of messages at the library or module level, depending on the specific appendix. The B appendices apply to HEP data processing, with B1 for libraries and B2 for modules. The C appendices apply to HCF data processing, with C1 for libraries and C2 for modules.

Appendix D presents the software elements involved with the set up of retrieval access. These elements are the DOS batch files used; module charts and reference lists (as explained above for appendices B1, B2, C1, and C2) for each module; THOR reports (as explained below for appendices E and F) for the data base named MASTER; and source code listings (as explained below for appendices G and H) for the three programs used in providing retrieval access.

Appendices E and F contain sample THOR reports showing the format and type of information available in such data base reports. These samples present a file summary, record definitions (relation schemas), and forms. THOR reports include their own table of contents and index. Appendix E is a sample from the data base named DATABANK, and Appendix F is a sample from the data base named HARDWARE.

Appendices G and H contain sample source code listings generated by the DOCUPROC utility. These samples show the form in which such listings are presented, with line numbering, table of contents, and index supplied. Appendix G is a sample for HEP data processing source code, and Appendix H is for HCF data processing source code.

An Example of Appendix Usage

In Appendix D can be found a microcosm example of the usage and form of other appendices in this volume. This presentation for retrieval access, including the module charts and reference lists, the schema and form for MASTER, and program listings, shows the full set of information used in reviewing software elements and their interaction. The

elements used here are simple enough and small enough to be readily understood while looking at them collectively. In contrast, consider the case for HEP data processing: Appendix B1 contains some flow charts for libraries; Appendix B2 contains some flow charts for modules; the schema for DATABANK takes about 500 pages to print; and the full set of source code listings would be a few hundred pages in printed form. By looking at these software elements together, their interaction can be observed. For example, in the charts and listings for the program MainMenu one can see where the data base MASTER is accessed and the form MenuM1 is displayed for the user. Again, notice that the listings and THOR reports include a table of contents and index for ease of locating elements of interest.

Requests for Additional Information

Programmers and other users of this volume desiring additional information are requested to contact either the NRC technical monitor or the NUCLARR Clearinghouse at the address listed below:

Thomas G. Ryan
U.S. Nuclear Regulatory Commission - RES
Reliability and Human Factors Branch
5640 Nicholson Lane, NL/N-316
Rockville, MD 20852 USA
(Phone) 301-492-3550

David I. Gertman
NUCLARR Data Clearinghouse
Idaho National Engineering Laboratory
P. O. Box 1625
Idaho Falls, ID 83415 USA
(Phone) 208-526-0652

REFERENCES

1. D. I. Gertman, W. E. Gilmore, W. J. Galyean, M. R. Groh, C. D. Gentillon, and B. G. Gilbert, Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR), Volume I: Summary Description, NUREG/CR-4639, EGG-2458, February 1988.

2. W. E. Gilmore, C. D. Gentillon, D. I. Gertman, G. H. Beers, W. J. Galyean, and B. G. Gilbert, Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR), Volume IV: User's Guide, Parts 1, 2, and 3, NUREG/CR-4639, EGG-2458, May 1988.

3. H. D. Stewart and K. D. Russell, SAGE: Modula-2 Tools and Utilities, Version 2.0, EGG-CATT-8229 (to be published).

4. D. M. Snider and K. L. Wagner, CRYSTAL Graphics: Two-dimensional Graphics in Modula-2, Version 2.0, EGG-CATT-8230 (to be published).

APPENDIX A

LIBRARY KEY

APPENDIX A

LIBRARY KEY

For a library [illegible] as part of the NUCLARR system software but accessed by NUCLARR, it is [illegible]s in which library set that library resides. The purpose of this library key is to [illegible]hich library set within the programming environment contains a particular library of interest. The list, on the left side of the library key page, names the library sets accessed by NUCLARR system software.

The column labels at an angle across the top of the key refer to the names listed to the left of the key. The left-most column of the key lists library names. An X to the right of a library name indicates which library set contains that library.

Libraries coded explicitly for NUCLARR are described in the text of this volume, and high level flow charts for some of them are found in Appendices B1 and C1. These NUCLARR libraries are not included on the library key.

LIBRARY KEY

SAGE System Libraries

Libraries provided within the SAGE system developed at INEL.

Documentation for these libraries can be found in the SAGE System manual.

Ad Hoc Standard Library

Set of libraries which augment the Logitech supplied libraries. These libraries are a part of the SAGE System.

Documentation for these libraries can be found in the SAGE System manual.

Modula-2 Libraries

Libraries provided by the Modula-2 language environment, supplied by Logitech, Inc. of Redwood City, California.

Documentation for these libraries can be found in the Modula-2 manual.

CRYSTAL Graphics Libraries

Set of libraries developed at INEL.

Documentation for these libraries can be found in the CRYSTAL Graphics manual.

Library	SAGE	Ad Hoc	Modula-2	CRYSTAL
ASCII			X	
Binary		X		
Convert		X		
DiskLib	X			
Files		X		
LCMath	X			
MoveLib	X			
MathLib		X		
PltAsth				X
PltCurve				X
PltDAxis				X
PltMess				X
PltInit				X
PltOpen				X
PltPrnt				X
PltSet				X
PltTDef				X
Program		X		
Reports	X			
Sage	X			
SageLib	X			
Scroll	X			
SimpleIO		X		
StandardIO		X		
String		X		
Strings			X	
SYSTEM			X	
Terminal		X		
ThorPort	X			
TimeLib	X			

APPENDIX B1

HEP DATA PROCESSING LIBRARY CHARTS

APPENDIX B1

HEP DATA PROCESSING LIBRARY CHARTS

This appendix contains high level flow charts for some libraries used in HEP data processing. Only charts for those libraries programmed explicitly for NUCLARR are included. These charts provide an overview of what each procedure is doing and how it is done; consequently these charts can serve as road maps to the source code.

These charts are organized by library, with the charts of a library's procedures presented in alphabetical order. Three lists precede the charts for each library. An alphabetic, italicized list of procedures local to the library is given first. This list is subdivided into exported procedures and those which are not exported. These local procedures are the procedures charted. The next list identifies procedures imported and names the library from which each procedure is imported. This list is alphabetized by procedure name with the procedure name italicized. If the libraries named here are not listed in Appendix A, they may be charted in this appendix. The final list is of messages referenced in the charts. Each message is assigned a number by which the messages are referenced and by which the list is ordered.

The elements in these charts are boxes and circles. Boxes contain descriptive comments on the processes being charted. Circled numbers are used to show chart continuation. Circled letters are used to show looping structures. Path lines, used to connect chart elements, have an arrow head at one end, thus showing the direction of activity flow from element to element.

Within boxes of the charts, procedure names are frequently used. These names are bolded and italicized. If a procedure name is considered descriptive enough, it may be used as a description of the process being shown; otherwise, the name will occur under a description and be indented showing that this procedure is used in the process described. If a local procedure is referenced, its name is followed by the dash character and then the word 'local'.

When multiple lines exit from a box, the condition determining the use of a path line will be indicated at the line. Any unlabeled path line exiting a box indicates the normal path to be taken.

When data base forms or relations are mentioned, they are shown within quotation marks.

LIBRARY NUCGEN

Local Procedures

Exported

CalcMedHEPEF
Card2Str0Fill
Clearinghouse
ClearStack
ConvertErrFailToUErrUFail
ConvertToDataManualPage
DataCheck
DocCodes
DocType
GetCellType
GetCellValidity
GetMatrixDesc
GetRowDescr
GetVerb
MyCloseRelation
MyOpenRelation
OutputFile
SearchDoc

Other

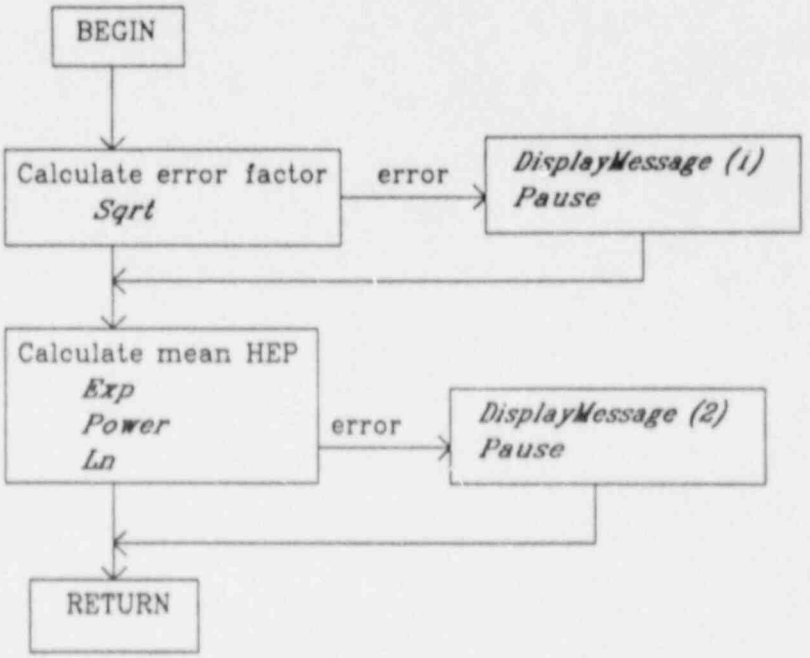
-none-

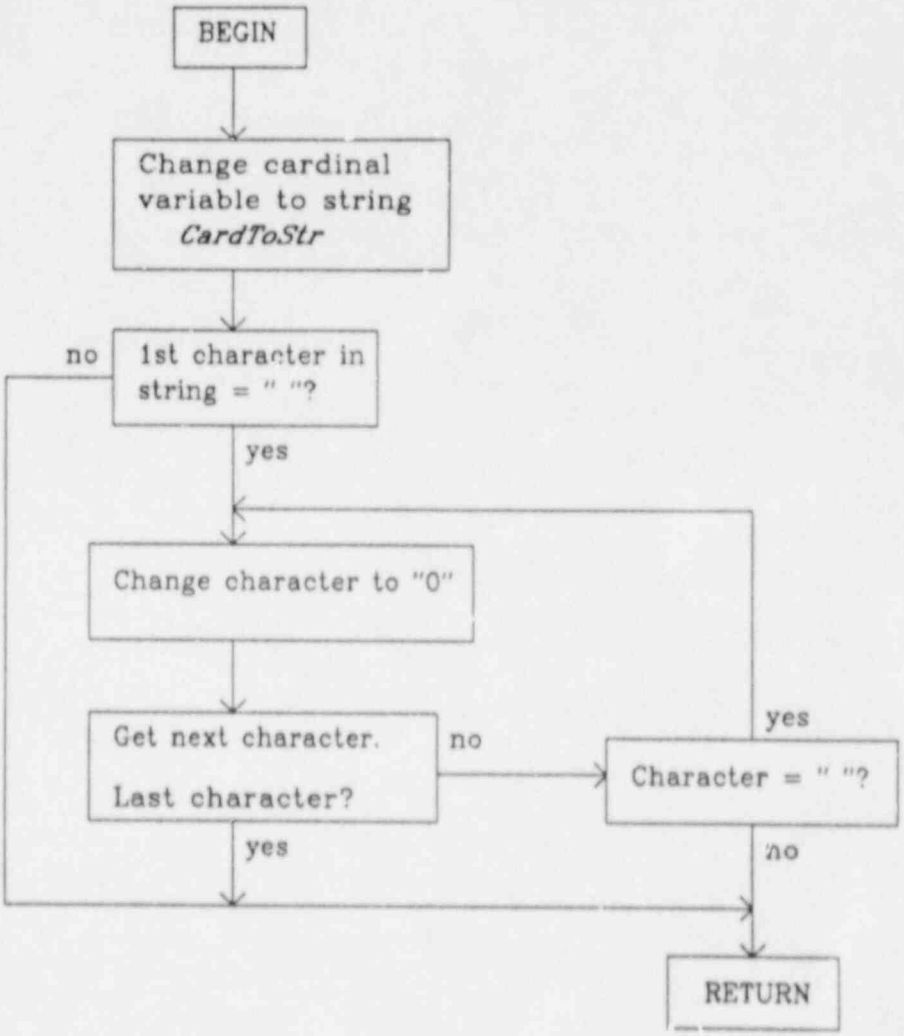
Procedures Imported

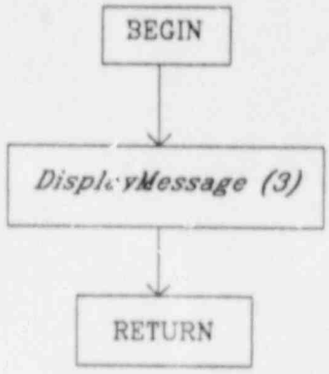
CardToStr Convert
ClearField Sage
ClearRelation Sage
CloseRelation Sage
CloseRelationFiles Sage
CloseReport Reports
CompareFieldC Sage
Concat String
CondRead Terminal
CopyField Sage
DefineHeader Reports
DisplayForm Sage
DisplayMessage Sage
Exp MathLib
FillChar SortLib
FindRecord Sage
GetFieldA Sage
GetFieldB Sage
GetFieldC Sage
GetFileName Files
GetRepeatName Sage
Ln MathLib
OpenRelation Sage
OpenReport Reports
Pause ThorPort
Power MathLib
PutFieldA Sage
PutFieldC Sage
ReadRecord Sage
Sqrt MathLib
StrToCard Convert
TopOfPage Reports
WrForm Reports
WriteRecord Sage
WrLn Reports
WrString Reports

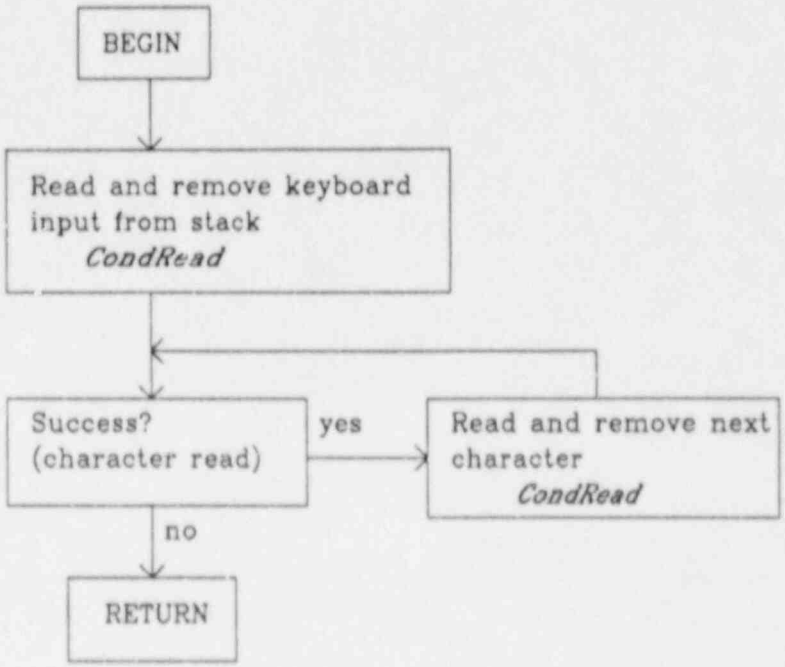
Messages

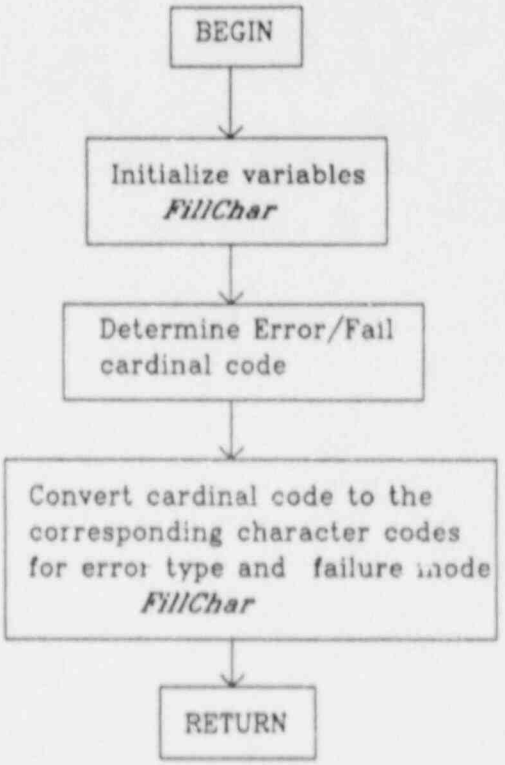
(1) - "* ERROR - error factor could not be calculated"
(2) - "* ERROR - CalcMedHEPEF, attempted: Ln(0)"
(3) - "Telephone Clearinghouse for assistance at (208)526-0735 or FTS 583-0735"
(4) - "* ERROR - could not open CellVal"
(5) - "* ERROR - could not open Columns"
(6) - "* ERROR - could not open MATRCOL"
(7) - "Invalid selection"

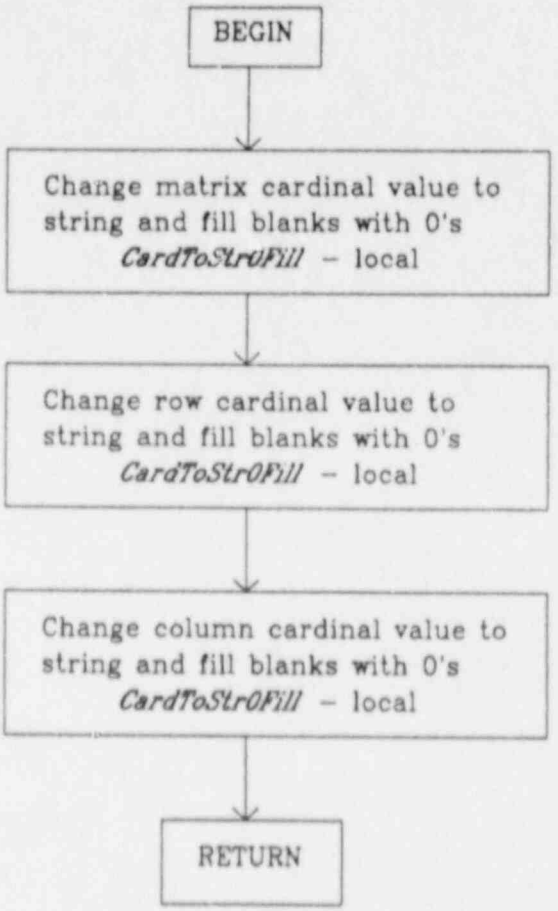

BEGIN
Calculate error factor
Sqrt
error
DisplayMessage (1)
Pause
Calculate mean HEP
Exp
Power
Ln
error
DisplayMessage (2)
Pause
RETURN

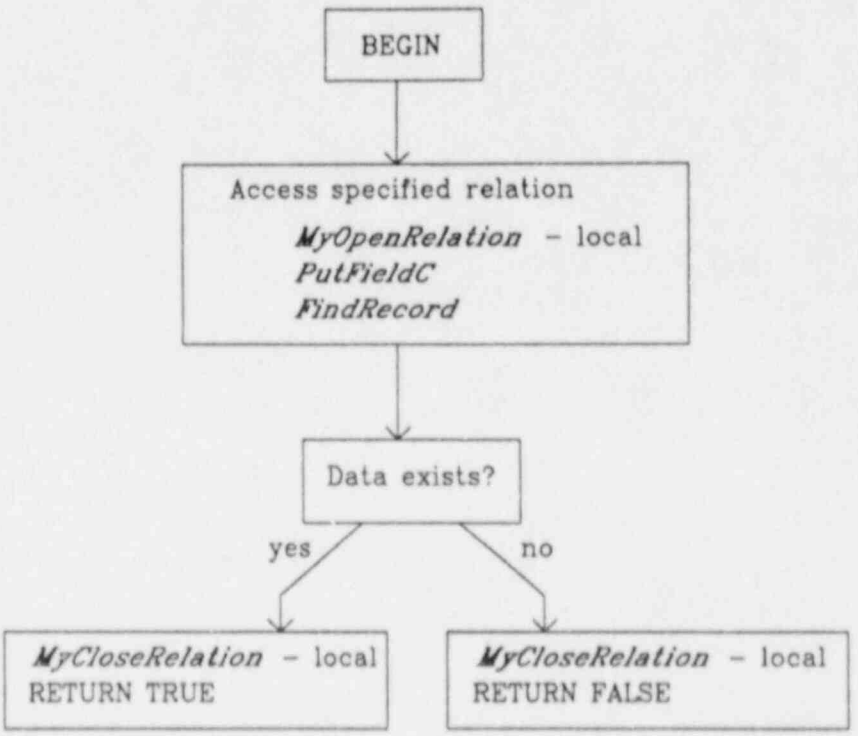

BEGIN
Change cardinal variable to string CardToStr
no
1st character in string = " "?
yes
Change character to "0"
Get next character.
Last character?
no
Character = " "?
yes
yes
no
RETURN

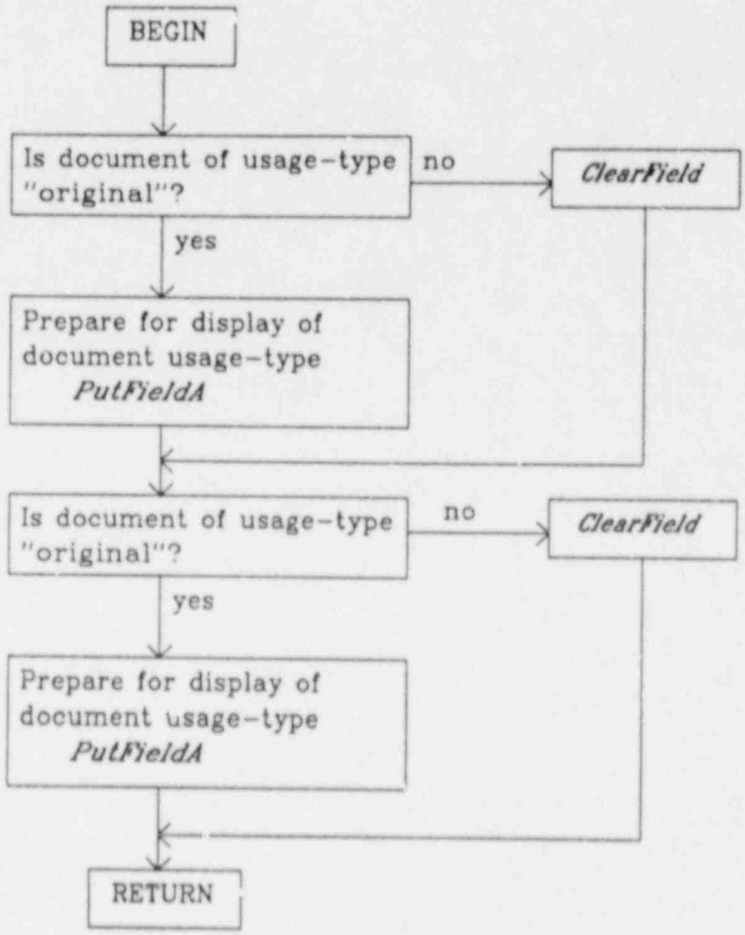

BEGIN
DisplayMessage (3)
RETURN

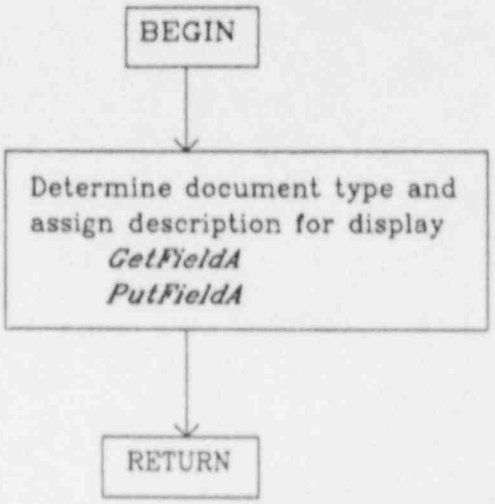

BEGIN
Read and remove keyboard input from stack
CondRead
Success? (character read)
yes
Read and remove next character
CondRead
no
RETURN

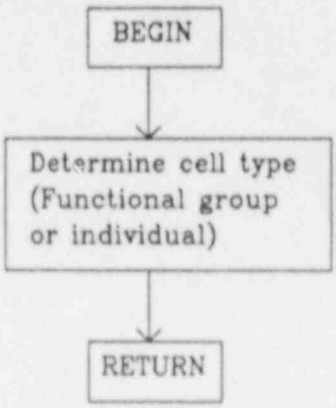

BEGIN
Initialize variables
FillChar
Determine Error/Fail
cardinal code
Convert cardinal code to the
corresponding character codes
for error type and failure mode
FillChar
RETURN

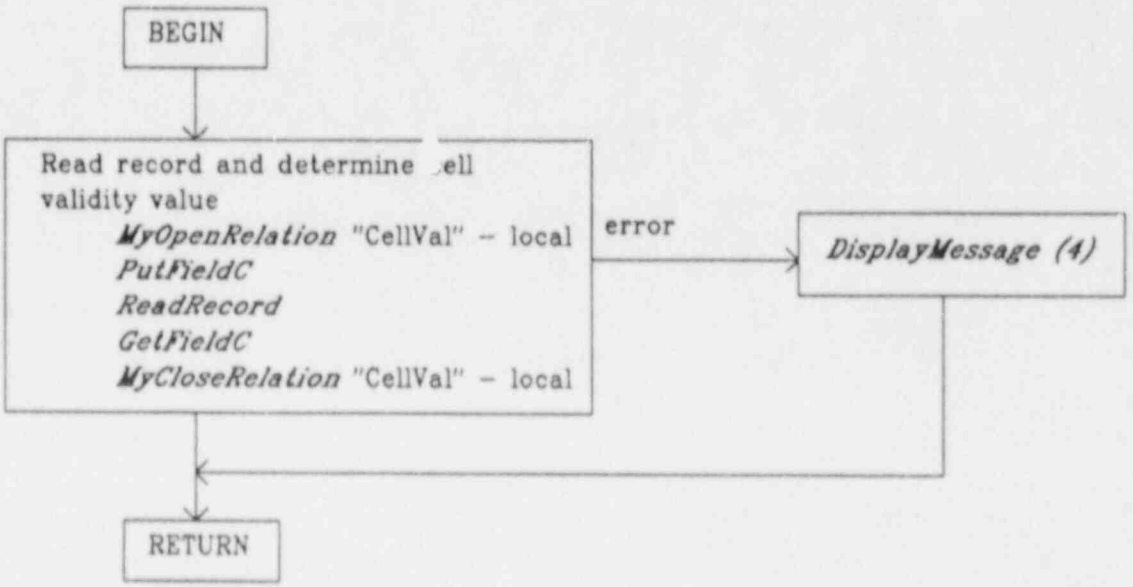

BEGIN
Change matrix cardinal value to string and fill blanks with 0's
CardToStr0Fill - local
Change row cardinal value to string and fill blanks with 0's
CardToStr0Fill - local
Change column cardinal value to string and fill blanks with 0's
CardToStr0Fill - local
RETURN

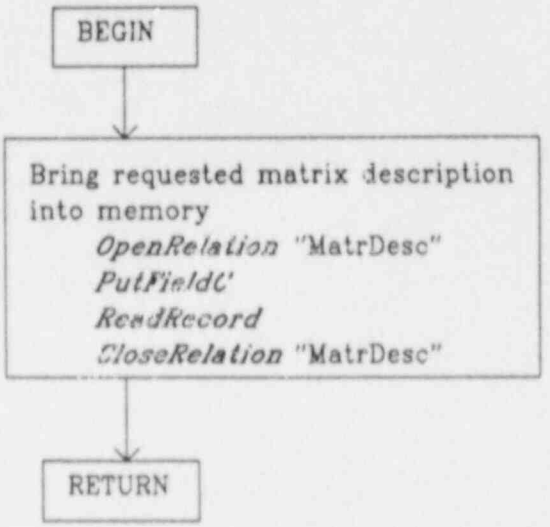

BEGIN
Access specified relation
MyOpenRelation - local
PutFieldC
FindRecord
Data exists?
yes
no
MyCloseRelation - local
RETURN TRUE
MyCloseRelation - local
RETURN FALSE

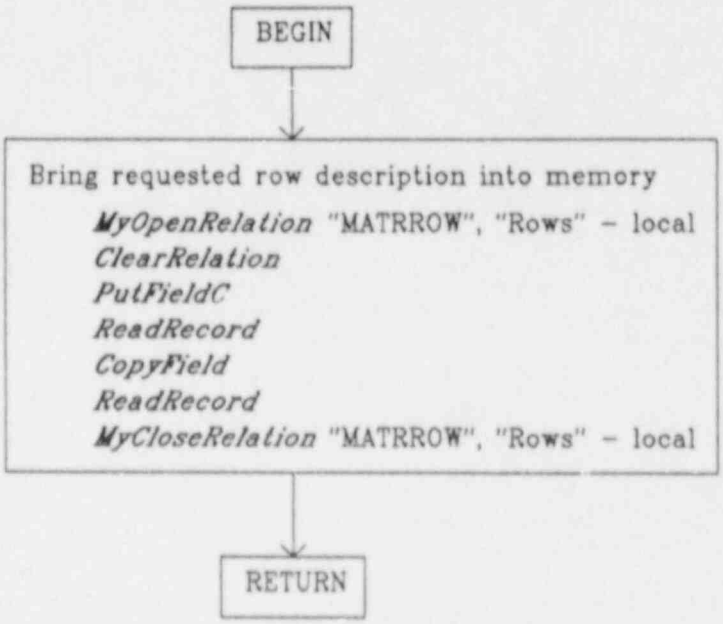

BEGIN
Is document of usage-type "original"?
no
ClearField
yes
Prepare for display of document usage-type
PutFieldA
Is document of usage-type "original"?
no
ClearField
yes
Prepare for display of document usage-type
PutFieldA
RETURN

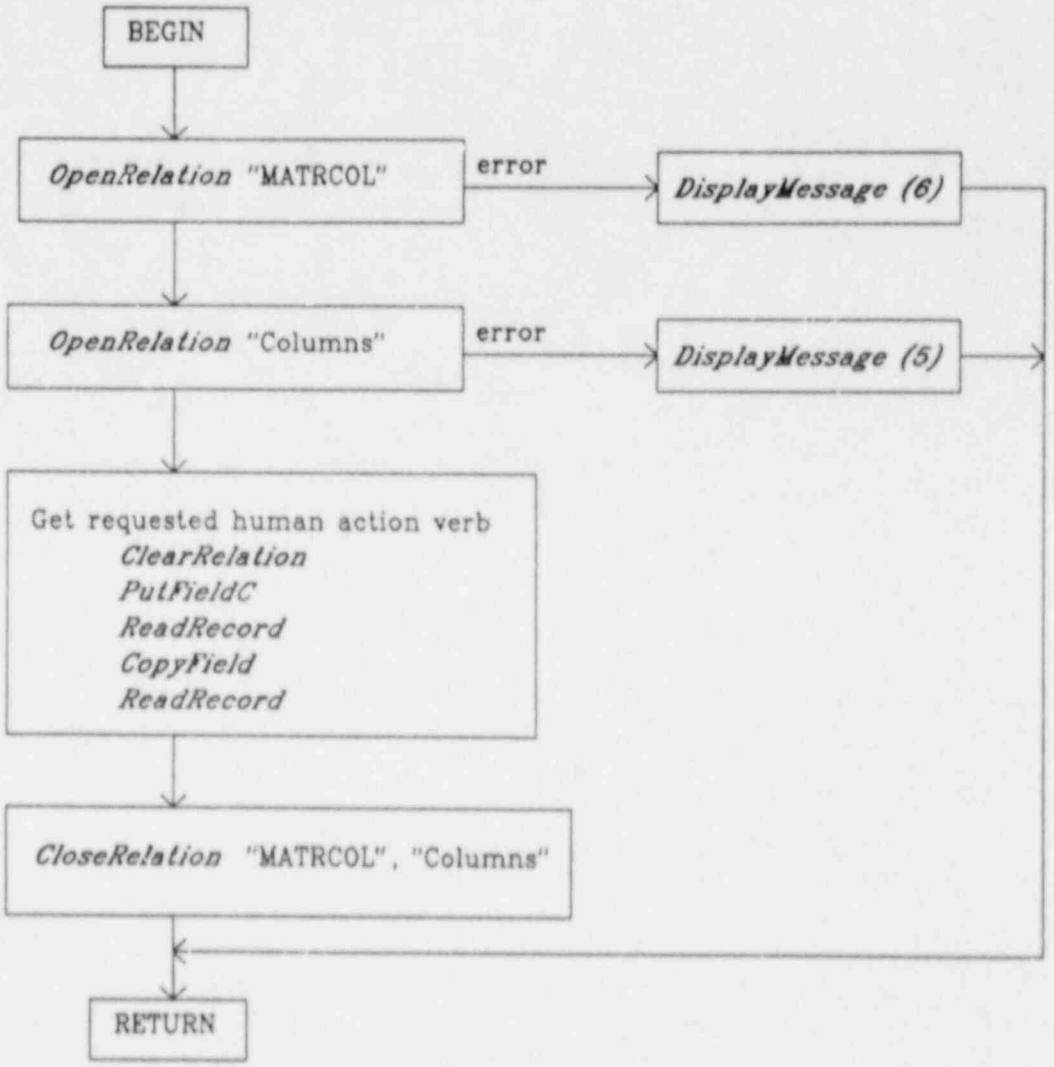

BEGIN
Determine document type and assign description for display
GetFieldA
PutFieldA
RETURN

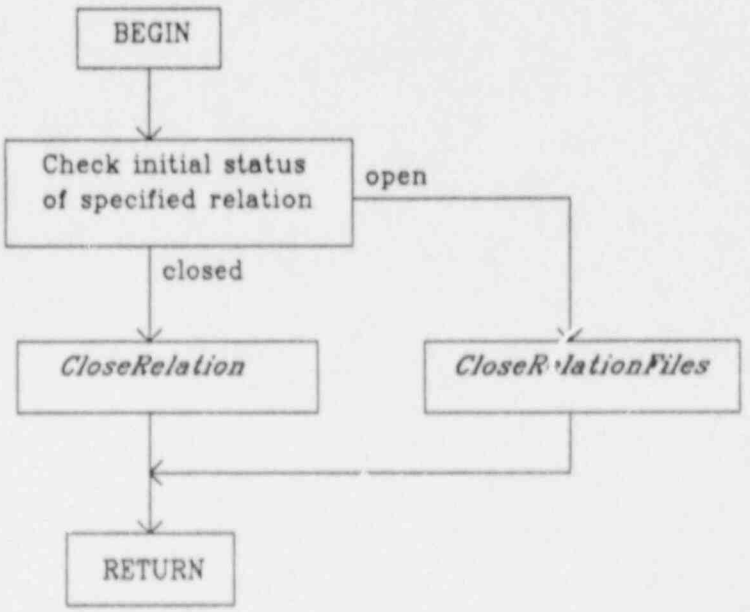

BEGIN
Determine cell type
(Functional group
or individual)
RETURN

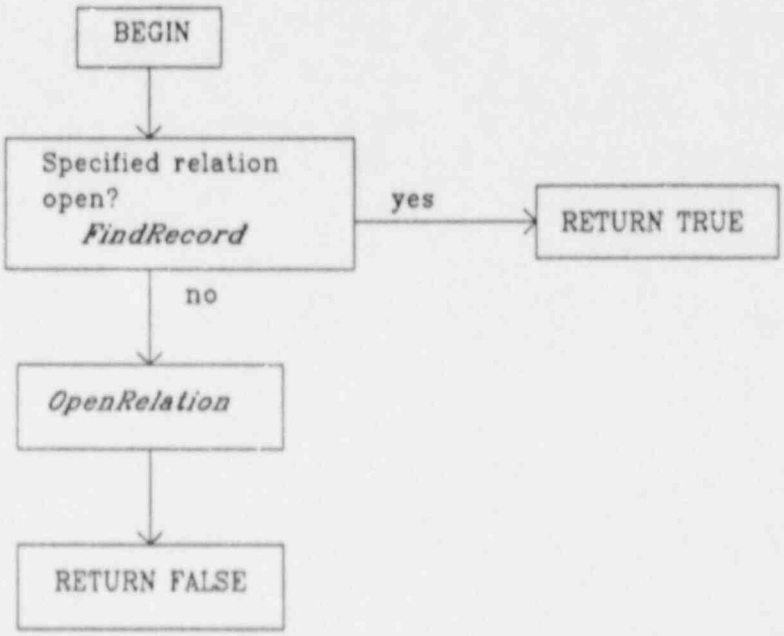

BEGIN
Read record and determine cell validity value
MyOpenRelation "CellVal" - local
PutFieldC
ReadRecord
GetFieldC
MyCloseRelation "CellVal" - local
error
DisplayMessage (4)
RETURN

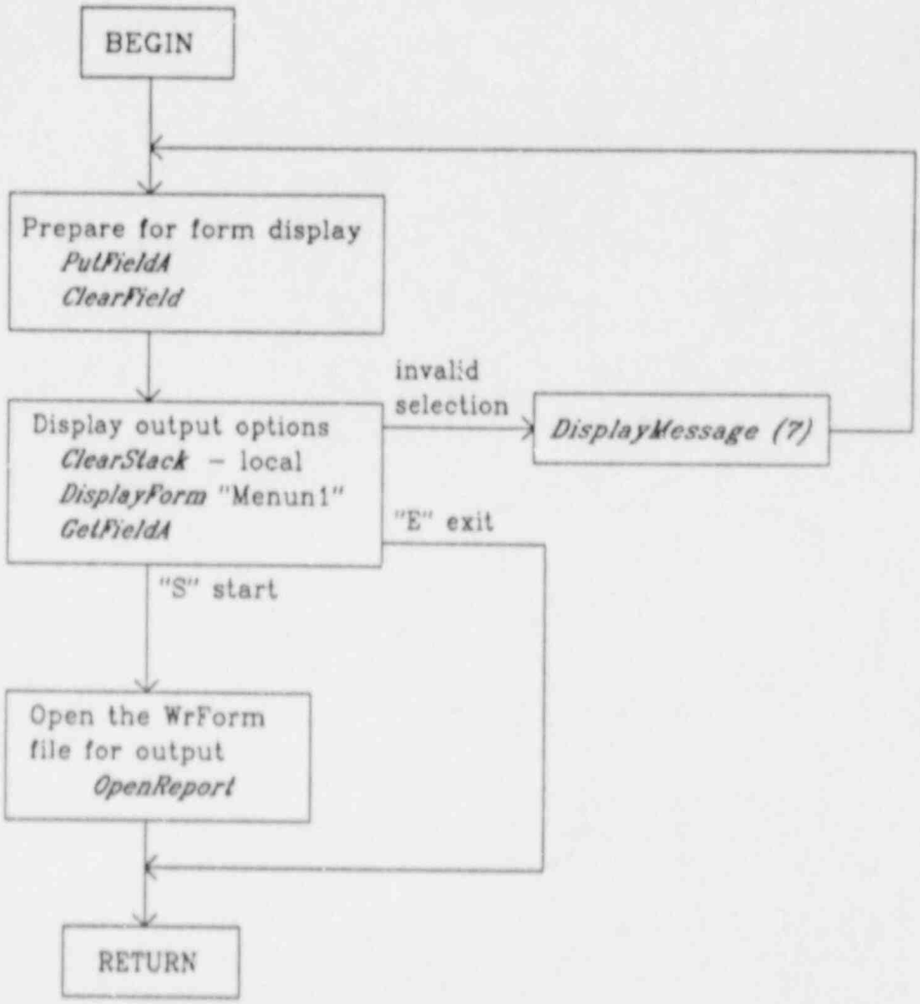

BEGIN
Bring requested matrix description into memory
OpenRelation "MatrDesc"
PutFieldC'
ReadRecord
CloseRelation "MatrDesc"
RETURN

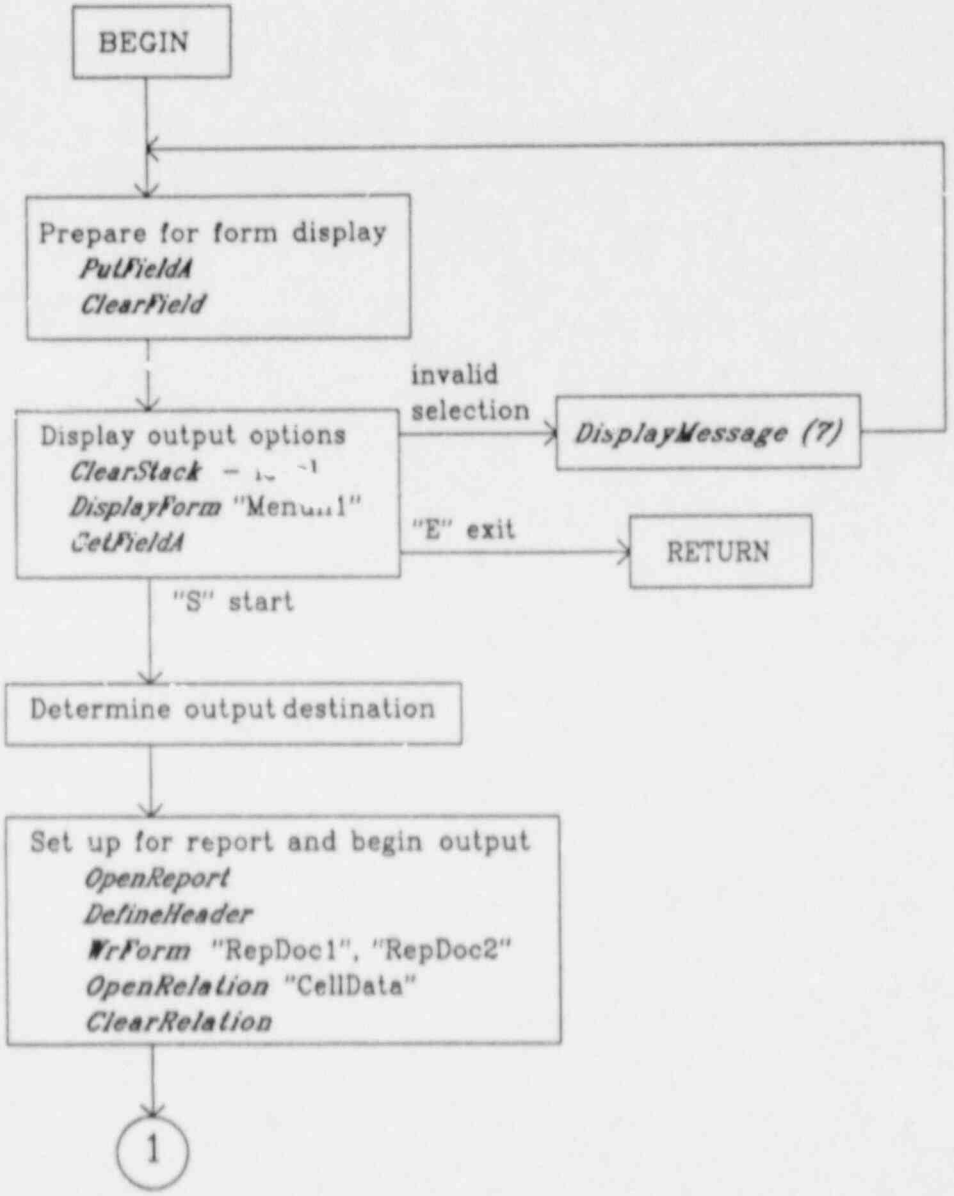

BEGIN
Bring requested row description into memory
MyOpenRelation "MATRROW", "Rows" - local
ClearRelation
PutFieldC
ReadRecord
CopyField
ReadRecord
MyCloseRelation "MATRROW", "Rows" - local
RETURN

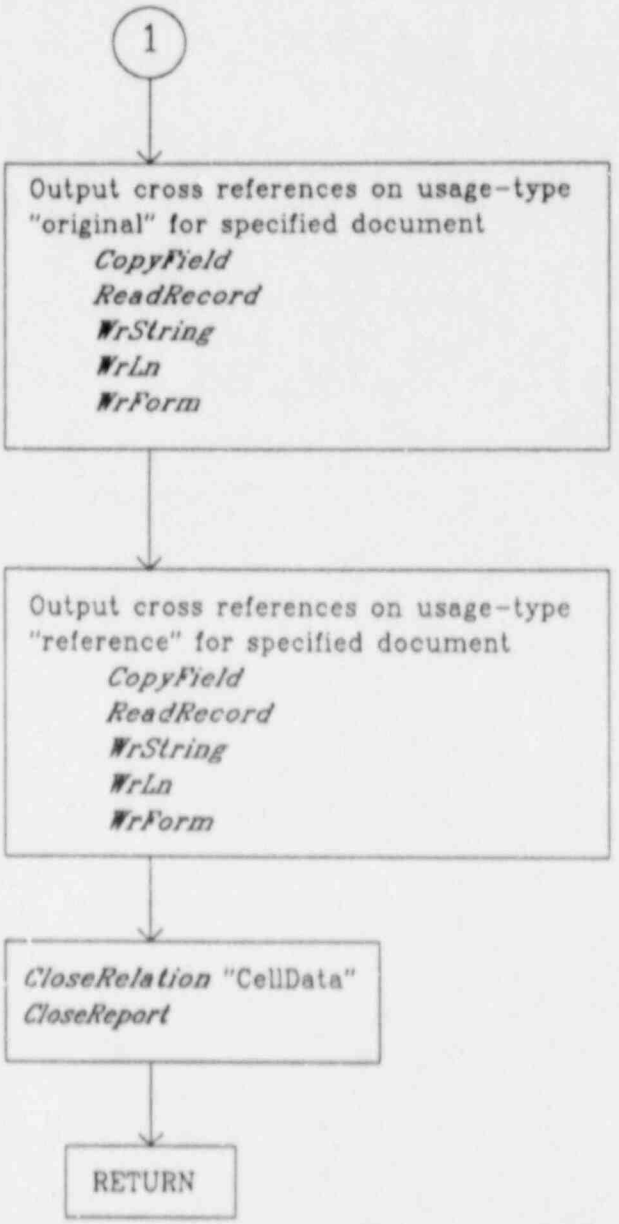

BEGIN
OpenRelation "MATRCOL"
error
DisplayMessage (6)
OpenRelation "Columns"
error
DisplayMessage (5)
Get requested human action verb
ClearRelation
PutFieldC
ReadRecord
CopyField
ReadRecord
CloseRelation "MATRCOL", "Columns"
RETURN


BEGIN
Check initial status of specified relation
open
closed
CloseRelation
CloseRelationFiles
RETURN


BEGIN
Specified relation open?
FindRecord
yes
RETURN TRUE
no
OpenRelation
RETURN FALSE


BEGIN
Prepare for form display
PutFieldA
ClearField
Display output options
ClearStack - local
DisplayForm "Menun1"
GetFieldA
invalid
selection
DisplayMessage (7)
"E" exit
"S" start
Open the WrForm
file for output
OpenReport
RETURN


BEGIN
Prepare for form display
PutFieldA
ClearField
Display output options
ClearStack
DisplayForm "Menu1"
GetFieldA
invalid selection
DisplayMessage (7)
"E" exit
RETURN
"S" start
Determine output destination
Set up for report and begin output
OpenReport
DefineHeader
WrForm "RepDoc1", "RepDoc2"
OpenRelation "CellData"
ClearRelation
1


1
Output cross references on usage-type "original" for specified document
CopyField
ReadRecord
WrString
WrLn
WrForm
Output cross references on usage-type "reference" for specified document
CopyField
ReadRecord
WrString
WrLn
WrForm
CloseRelation "CellData"
CloseReport
RETURN

LIBRARY StorageM

Local Procedures

Exported

ClearSBuffer
DecodeCellDataKey
GetFile
IncludeNewKey
RemoveKey
RetrieveKey
SaveFile
ShowBufferDescription
ViewBufferData

Other

GetDocumentAndPlant
ReadyForFileIO
ShowCombinedHEPs

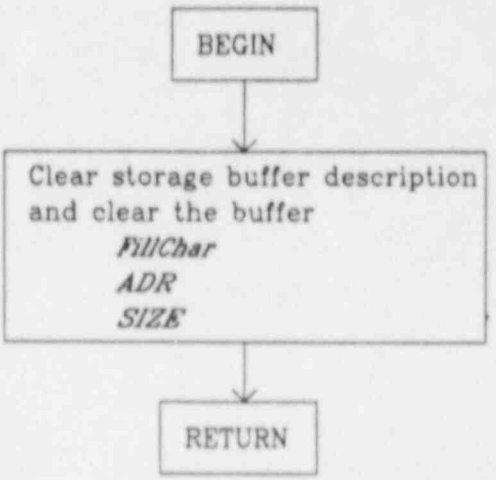
Procedures Imported

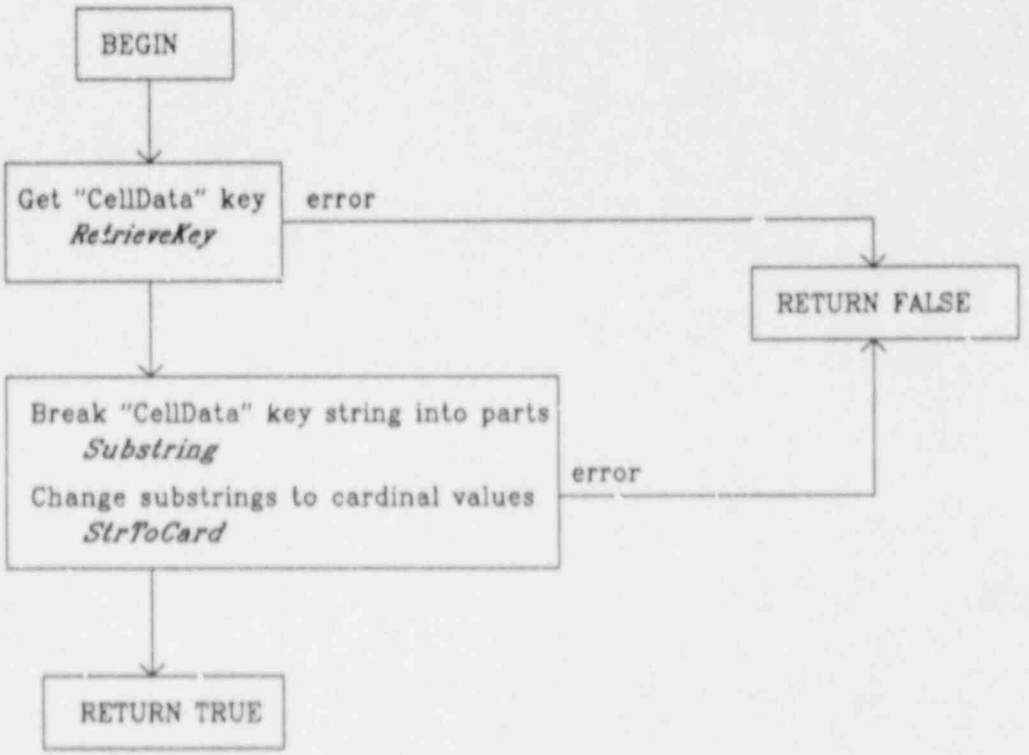
ADR SYSTEM
BinaryDelete SortLib
BinaryInsert SortLib
BinarySearch SortLib
CalcMedHEPEF NUCGEN
ClearField Sage
ClearingHouse NUCGEN
ClearRelation Sage
ClearStack NUCGEN
Close Files
CloseRelation Sage
CloseRelationFiles Sage
CompareKey SortLib
Concat String
ConvertToDataManualPage NUCGEN
CopyField Sage
DisplayBackground Sage
DisplayForm Sage
DisplayMessage Sage
DisplaySource NUCPRINT
FillChar MoveLib
GetCellType NUCGEN
GetFieldA Sage
GetFieldB Sage
GetFieldC Sage
GetFieldF Sage
Lookup DiskLib
MoveString ThorUtil
OpenRelation Sage
PutFieldA Sage
PutFieldB Sage
PutFieldC Sage
PutFieldF Sage
ReadBytes Binary
Remove Files
Reset Files
Rewrite Files
SIZE SYSTEM
StrToCard Convert
Substring String
TSIZE SYSTEM
WriteBytes Binary

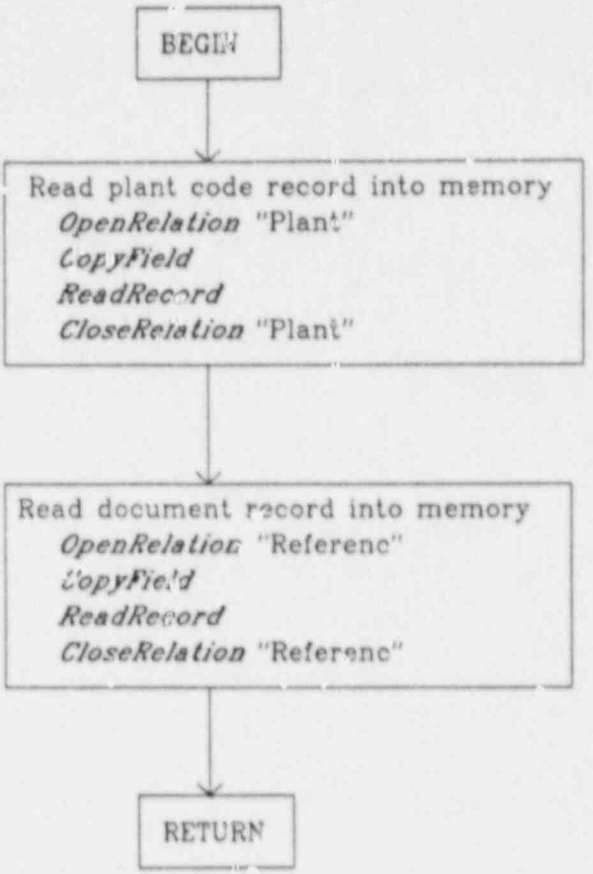
Messages

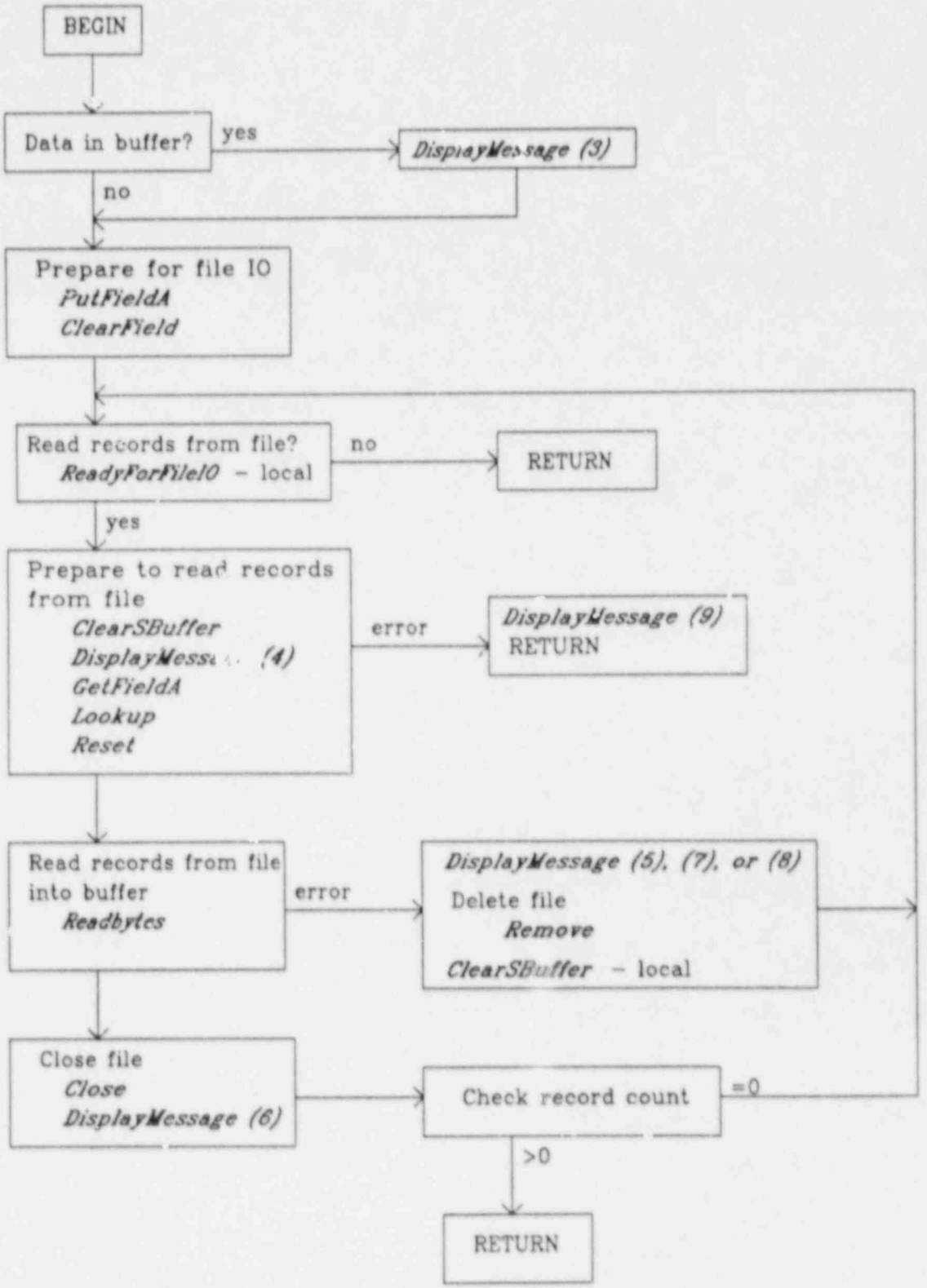
(1) - "A valid DOS file name is needed for this option"
(2) - "Selected option is not provided"
(3) - "Current located records will be discarded if you proceed"
(4) - "Copying records from requested file . . . please stand by"
(5) - "Can not read records on requested file"
(6) - "All records have been copied from requested file"
(7) - "Can not read description on requested file"
(8) - "Can not read record count on requested file"
(9) - "Requested file is not available"
(10) - "Last Sample Plot being displayed"

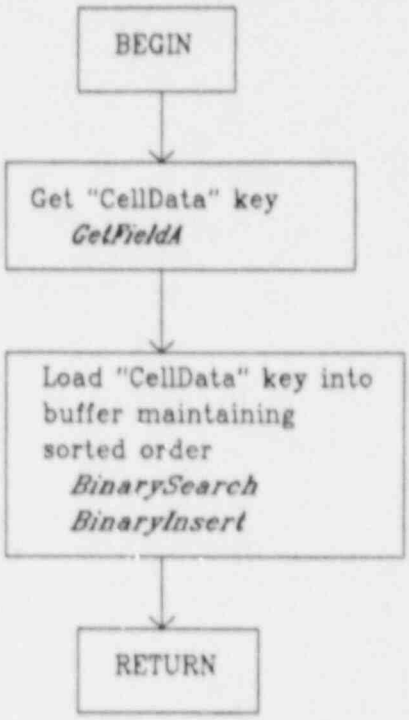
(11) - ''Failed in attempt to copy the storage buffer to requested file''
(12) - ''Storage buffer is empty, there is nothing to copy to a file''
(13) - ''Records have been copied to requested file''
(14) - ''Description of storage buffer was updated''
(15) - ''No HEP available for this task''
(16) - ''Division by zero attempted (for mean or erf)''
(17) - ''No HEP available for this cell''
(18) - ''Can not determine functional group''
(19) - ''No HEP available for this functional group''
(20) - ''First record in storage buffer, use 'P' again to see first record''
(21) - ''Last record in storage buffer, use 'N' again to see first record''
(22) - ''No data in buffer to be viewed''
(23) - ''Could not read search buffer''
(24) - ''Illegal Error/Fail code in data''
(25) - ''Requested record removed from search buffer''
(26) - "Unselect was used to clear S buffer"

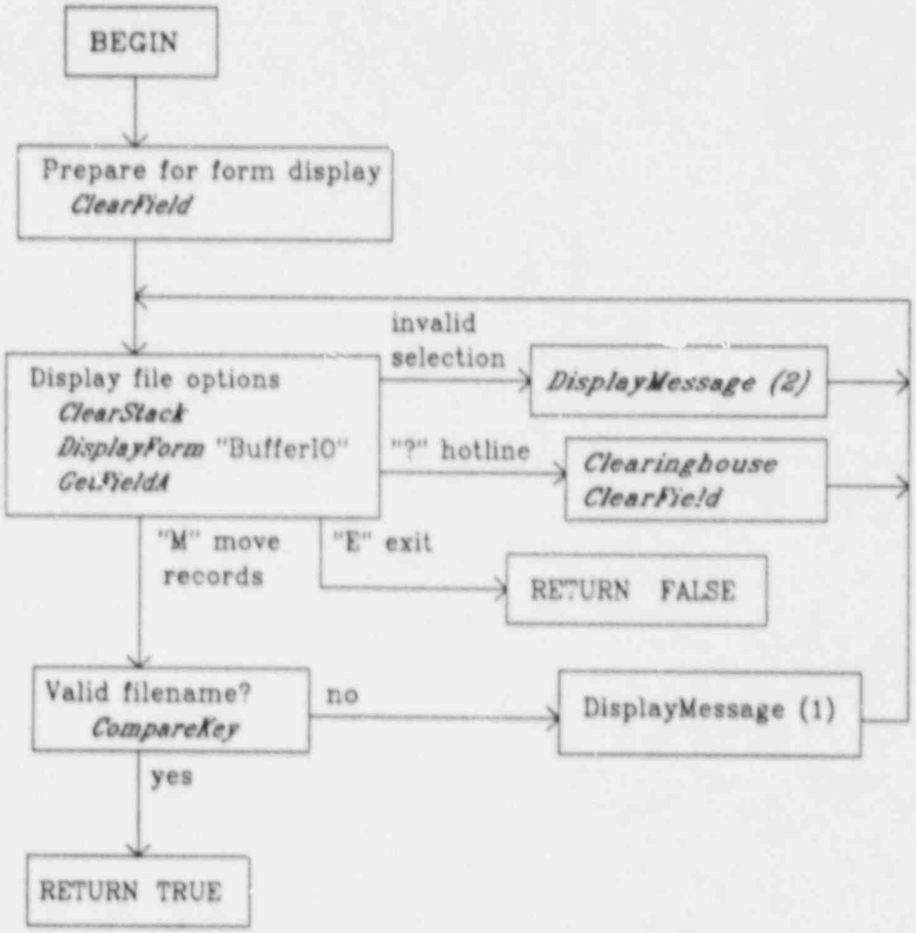

BEGIN
Clear storage buffer description
and clear the buffer
FillChar
ADR
SIZE
RETURN

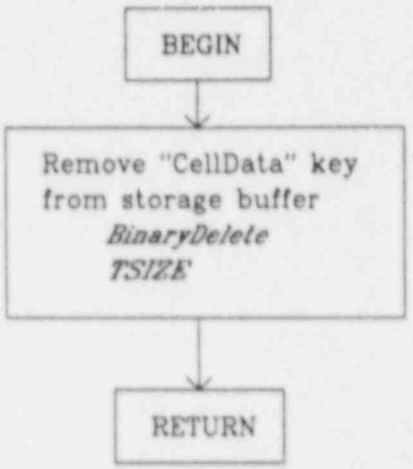

BEGIN
Get "CellData" key
RetrieveKey
error
RETURN FALSE
Break "CellData" key string into parts
Substring
Change substrings to cardinal values
StrToCard
error
RETURN TRUE

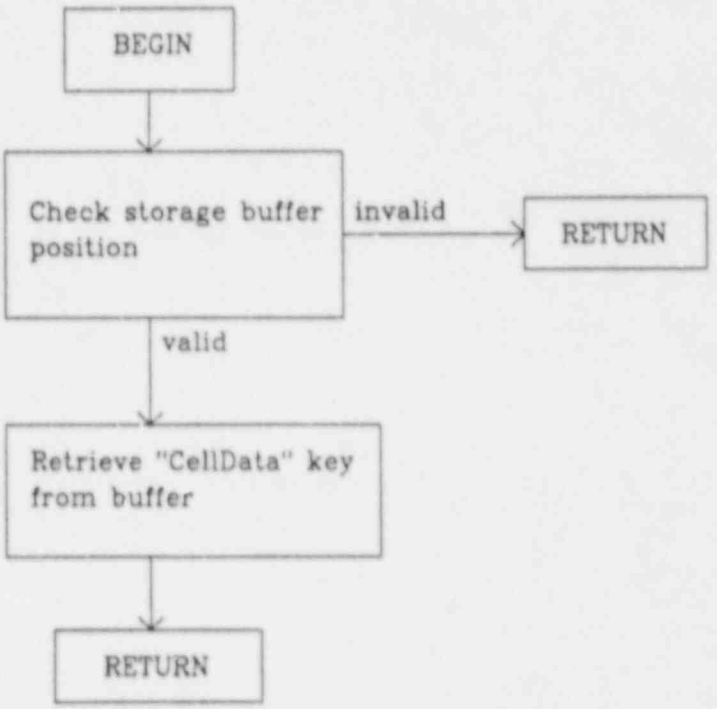

BEGIN
Read plant code record into memory
OpenRelation "Plant"
CopyField
ReadRecord
CloseRelation "Plant"
Read document record into memory
OpenRelation "Referenc"
CopyField
ReadRecord
CloseRelation "Referenc"
RETURN

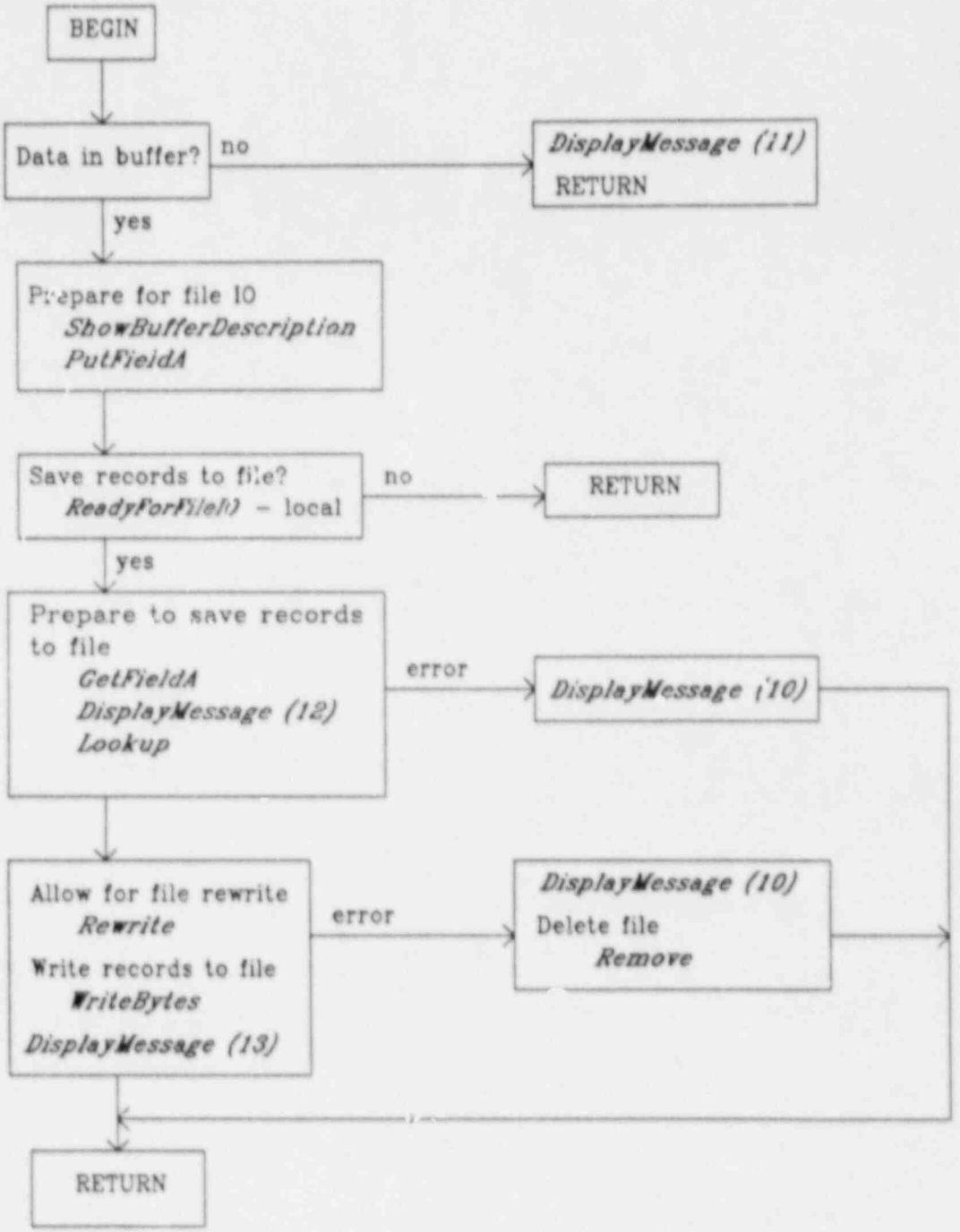

BEGIN
Data in buffer?
yes
DisplayMessage (3)
no
Prepare for file IO
PutFieldA
ClearField
Read records from file?
ReadyForFileIO - local
no
RETURN
yes
Prepare to read records from file
ClearSBuffer
DisplayMessage (4)
GetFieldA
Lookup
Reset
error
DisplayMessage (9)
RETURN
Read records from file into buffer
Readbytes
error
DisplayMessage (5), (7), or (8)
Delete file
Remove
ClearSBuffer - local
Close file
Close
DisplayMessage (6)
Check record count
=0
>0
RETURN

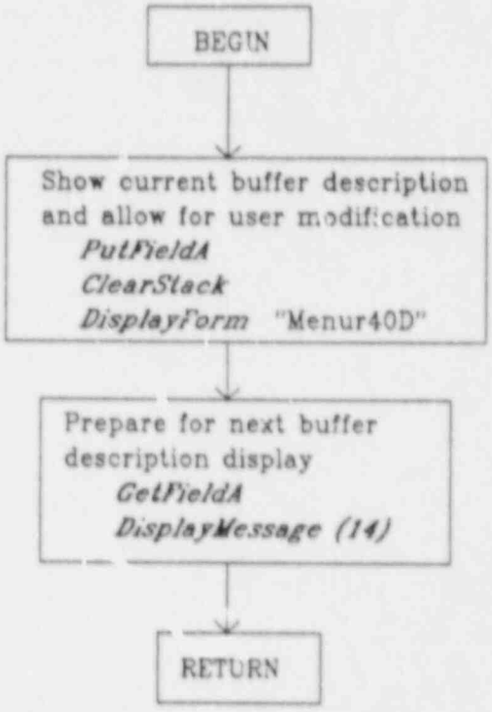

BEGIN
Get "CellData" key
GetFieldA
Load "CellData" key into
buffer maintaining
sorted order
BinarySearch
BinaryInsert
RETURN

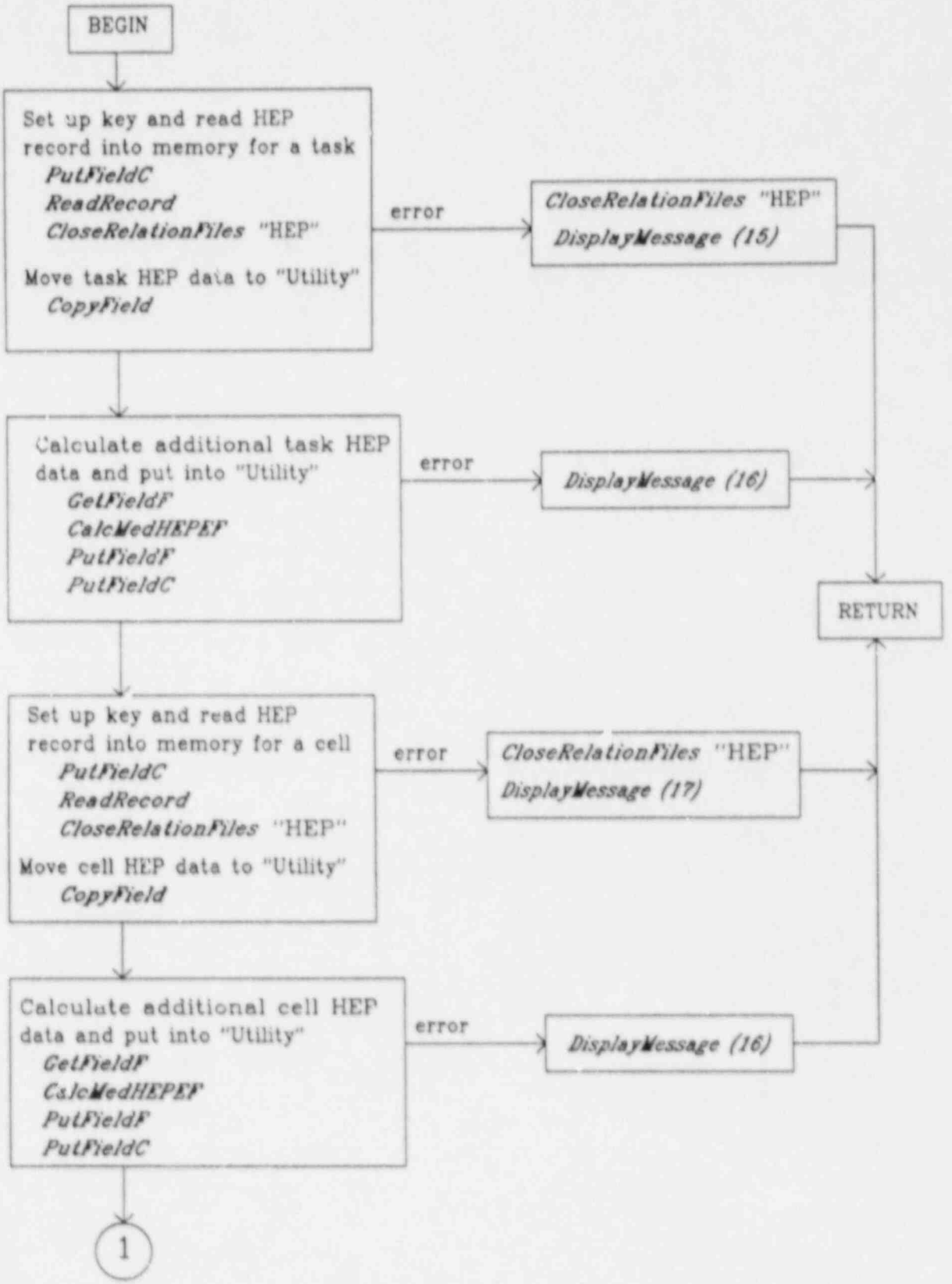

BEGIN
Prepare for form display
ClearField
Display file options
ClearStack
DisplayForm "BufferIO"
GetFieldA
invalid selection
DisplayMessage (2)
"?" hotline
Clearinghouse ClearField
"M" move records
"E" exit
RETURN FALSE
Valid filename?
CompareKey
no
DisplayMessage (1)
yes
RETURN TRUE

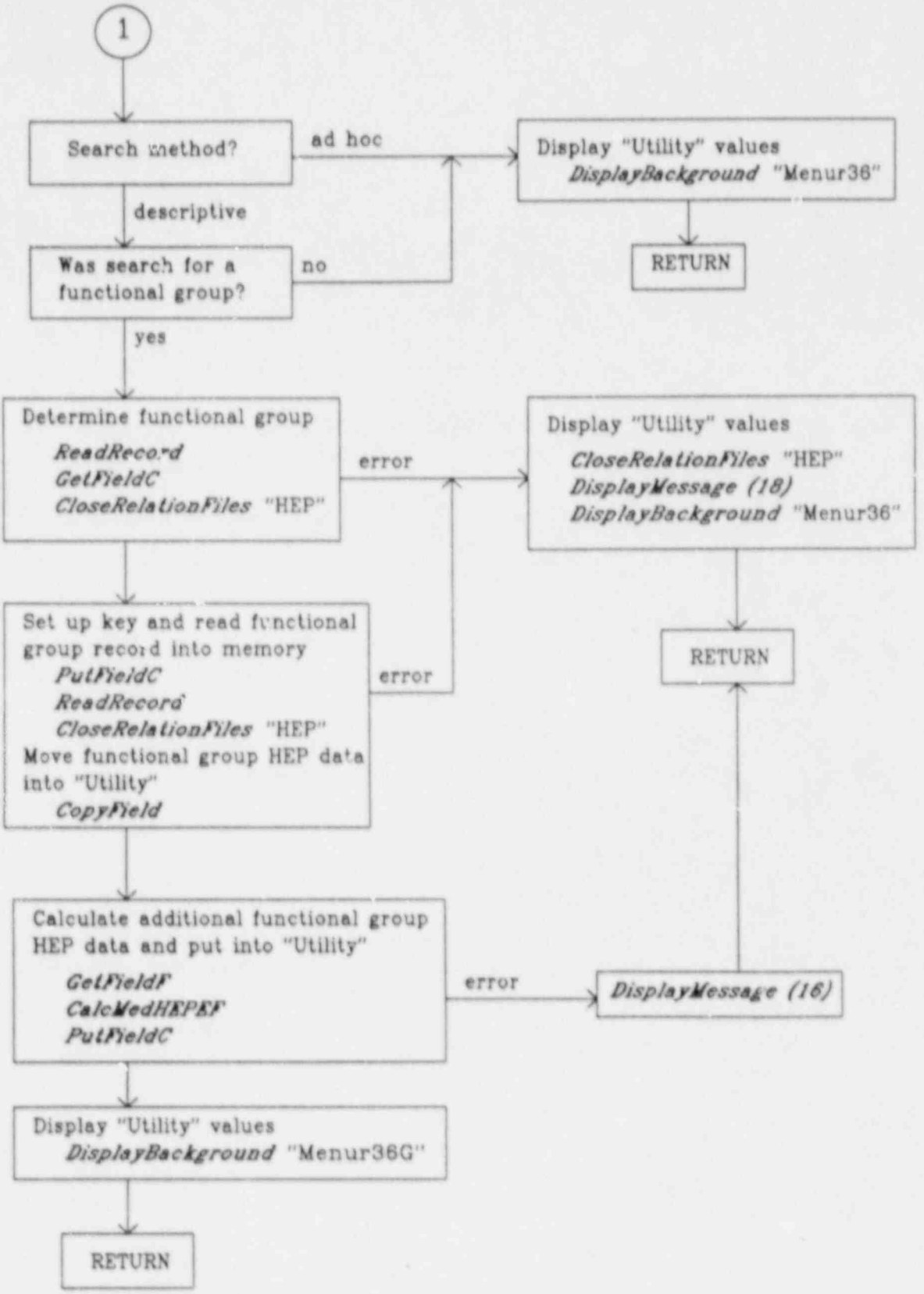

BEGIN
Remove "CellData" key
from storage buffer
BinaryDelete
TSIZE
RETURN

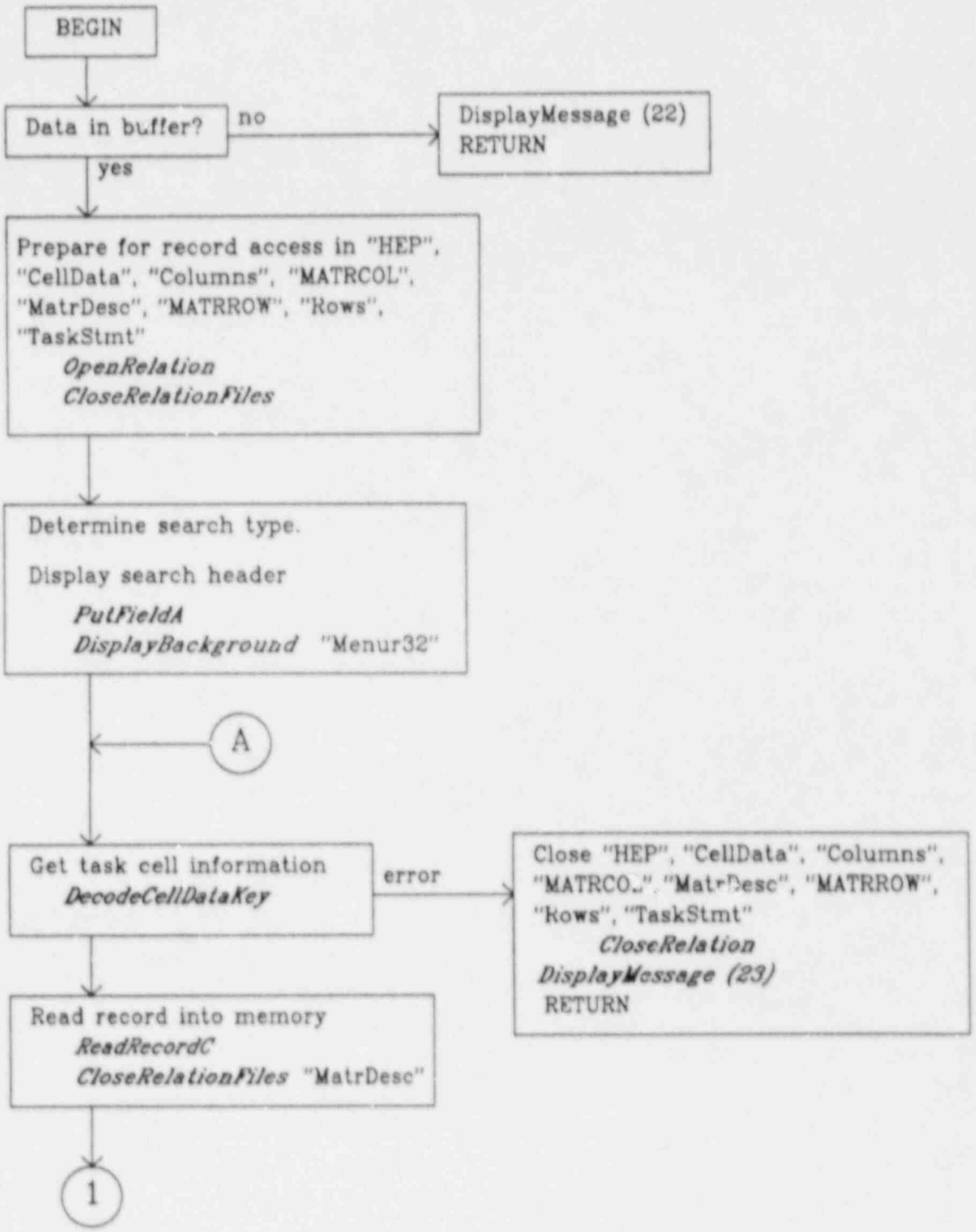

BEGIN
Check storage buffer position
invalid
RETURN
valid
Retrieve "CellData" key from buffer
RETURN

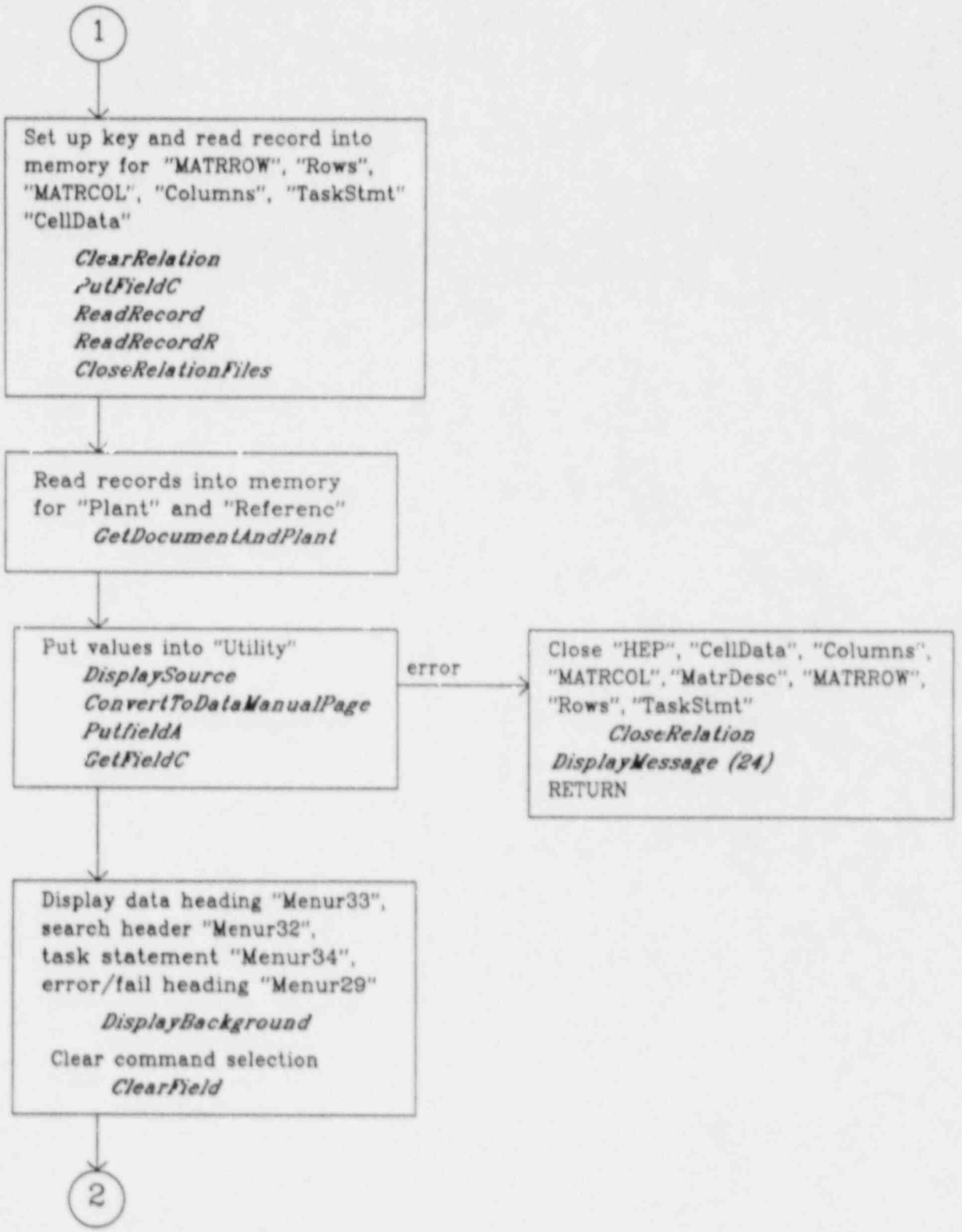

BEGIN
Data in buffer?
no
DisplayMessage (11)
RETURN
yes
Prepare for file IO
ShowBufferDescription
PutFieldA
Save records to file?
ReadyForFile() - local
no
RETURN
yes
Prepare to save records to file
GetFieldA
DisplayMessage (12)
Lookup
error
DisplayMessage (10)
Allow for file rewrite
Rewrite
Write records to file
WriteBytes
DisplayMessage (13)
error
DisplayMessage (10)
Delete file
Remove
RETURN

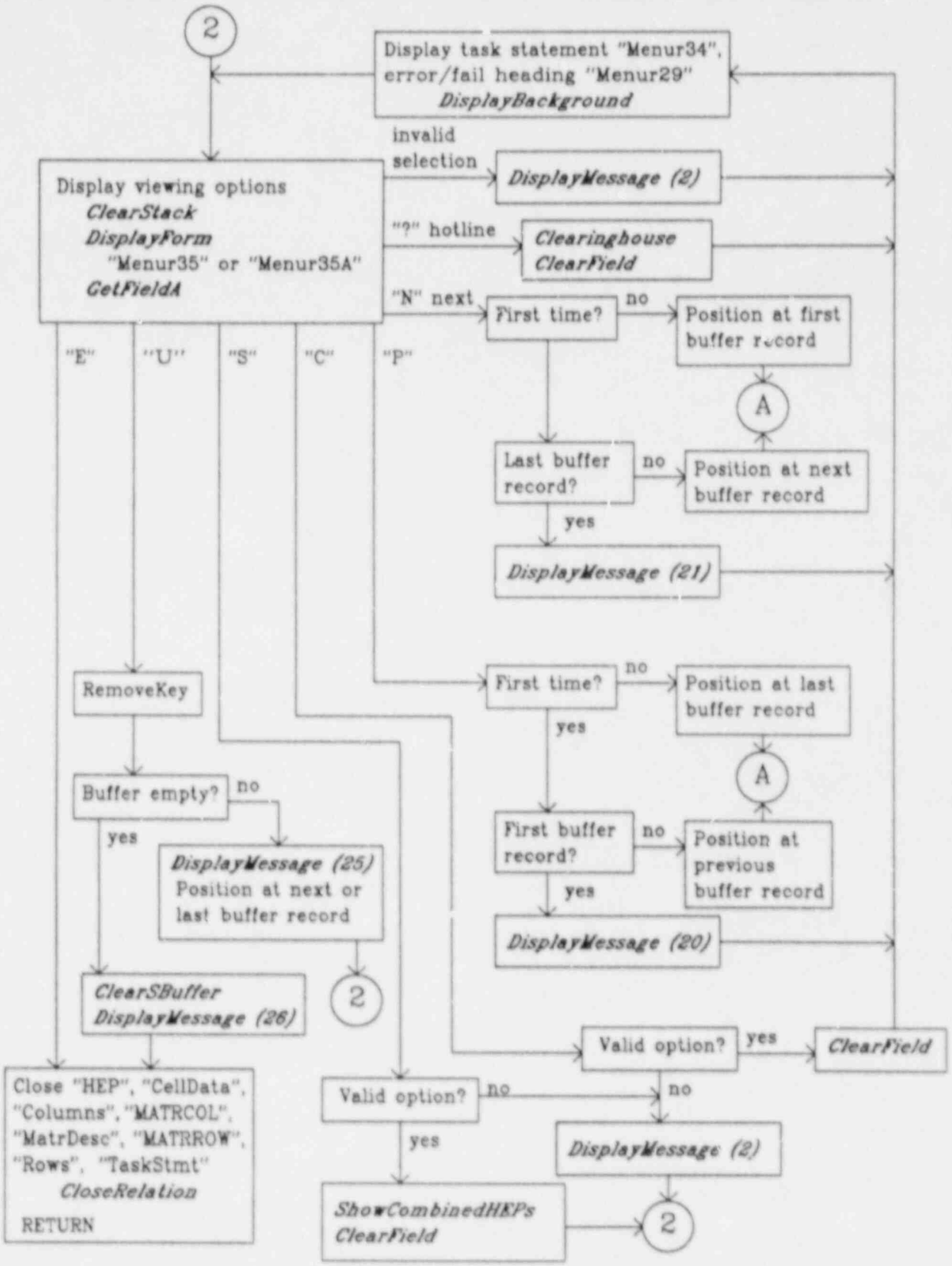

BEGIN
Show current buffer description
and allow for user modification
PutFieldA
ClearStack
DisplayForm "Menur40D"
Prepare for next buffer
description display
GetFieldA
DisplayMessage (14)
RETURN


BEGIN
Set up key and read HEP record into memory for a task
PutFieldC
ReadRecord
CloseRelationFiles "HEP"
Move task HEP data to "Utility"
CopyField
error
CloseRelationFiles "HEP"
DisplayMessage (15)
Calculate additional task HEP data and put into "Utility"
GetFieldF
CalcMedHEPEF
PutFieldF
PutFieldC
error
DisplayMessage (16)
RETURN
Set up key and read HEP record into memory for a cell
PutFieldC
ReadRecord
CloseRelationFiles "HEP"
Move cell HEP data to "Utility"
CopyField
error
CloseRelationFiles "HEP"
DisplayMessage (17)
Calculate additional cell HEP data and put into "Utility"
GetFieldF
CalcMedHEPEF
PutFieldF
PutFieldC
error
DisplayMessage (16)
1


1
Search method?
ad hoc
descriptive
Was search for a functional group?
no
yes
Display "Utility" values
DisplayBackground "Menur36"
RETURN
Determine functional group
ReadRecord
GetFieldC
CloseRelationFiles "HEP"
error
Display "Utility" values
CloseRelationFiles "HEP"
DisplayMessage (18)
DisplayBackground "Menur36"
RETURN
Set up key and read functional group record into memory
PutFieldC
ReadRecord
CloseRelationFiles "HEP"
error
Move functional group HEP data into "Utility"
CopyField
Calculate additional functional group HEP data and put into "Utility"
GetFieldF
CalcMedHEPEF
PutFieldC
error
DisplayMessage (16)
Display "Utility" values
DisplayBackground "Menur36G"
RETURN


BEGIN
Data in buffer?
no
DisplayMessage (22)
RETURN
yes
Prepare for record access in "HEP", "CellData", "Columns", "MATRCOL", "MatrDesc", "MATRROW", "Rows", "TaskStmt"
OpenRelation
CloseRelationFiles
Determine search type.
Display search header
PutFieldA
DisplayBackground "Menur32"
A
Get task cell information
DecodeCellDataKey
error
Close "HEP", "CellData", "Columns", "MATRCOL", "MatrDesc", "MATRROW", "Rows", "TaskStmt"
CloseRelation
DisplayMessage (23)
RETURN
Read record into memory
ReadRecordC
CloseRelationFiles "MatrDesc"
1


1
Set up key and read record into memory for "MATRROW", "Rows", "MATRCOL", "Columns", "TaskStmt" "CellData"
ClearRelation
PutFieldC
ReadRecord
ReadRecordR
CloseRelationFiles
Read records into memory for "Plant" and "Referenc"
GetDocumentAndPlant
Put values into "Utility"
DisplaySource
ConvertToDataManualPage
PutfieldA
GetFieldC
error
Close "HEP", "CellData", "Columns", "MATRCOL", "MatrDesc", "MATRROW", "Rows", "TaskStmt"
CloseRelation
DisplayMessage (24)
RETURN
Display data heading "Menur33", search header "Menur32", task statement "Menur34", error/fail heading "Menur29"
DisplayBackground
Clear command selection
ClearField
2


2
Display task statement "Menur34",
error/fail heading "Menur29"
DisplayBackground
invalid selection
DisplayMessage (2)
Display viewing options
ClearStack
DisplayForm
"Menur35" or "Menur35A"
GetFieldA
"?" hotline
Clearinghouse
ClearField
"N" next
First time?
no
Position at first buffer record
A
Last buffer record?
no
Position at next buffer record
yes
DisplayMessage (21)
"E"
"U"
"S"
"C"
"P"
RemoveKey
First time?
no
Position at last buffer record
yes
A
Buffer empty?
no
yes
First buffer record?
no
Position at previous buffer record
yes
DisplayMessage (25)
Position at next or last buffer record
DisplayMessage (20)
ClearSBuffer
DisplayMessage (26)
2
Valid option?
yes
ClearField
Close "HEP", "CellData", "Columns", "MATRCOL", "MatrDesc", "MATRROW", "Rows", "TaskStmt"
CloseRelation
RETURN
Valid option?
no
no
yes
DisplayMessage (2)
ShowCombinedHEPs
ClearField
2

LIBRARY Retrieve

Local Procedures

Exported

CheckForError *FilesLeftOpen*

Other

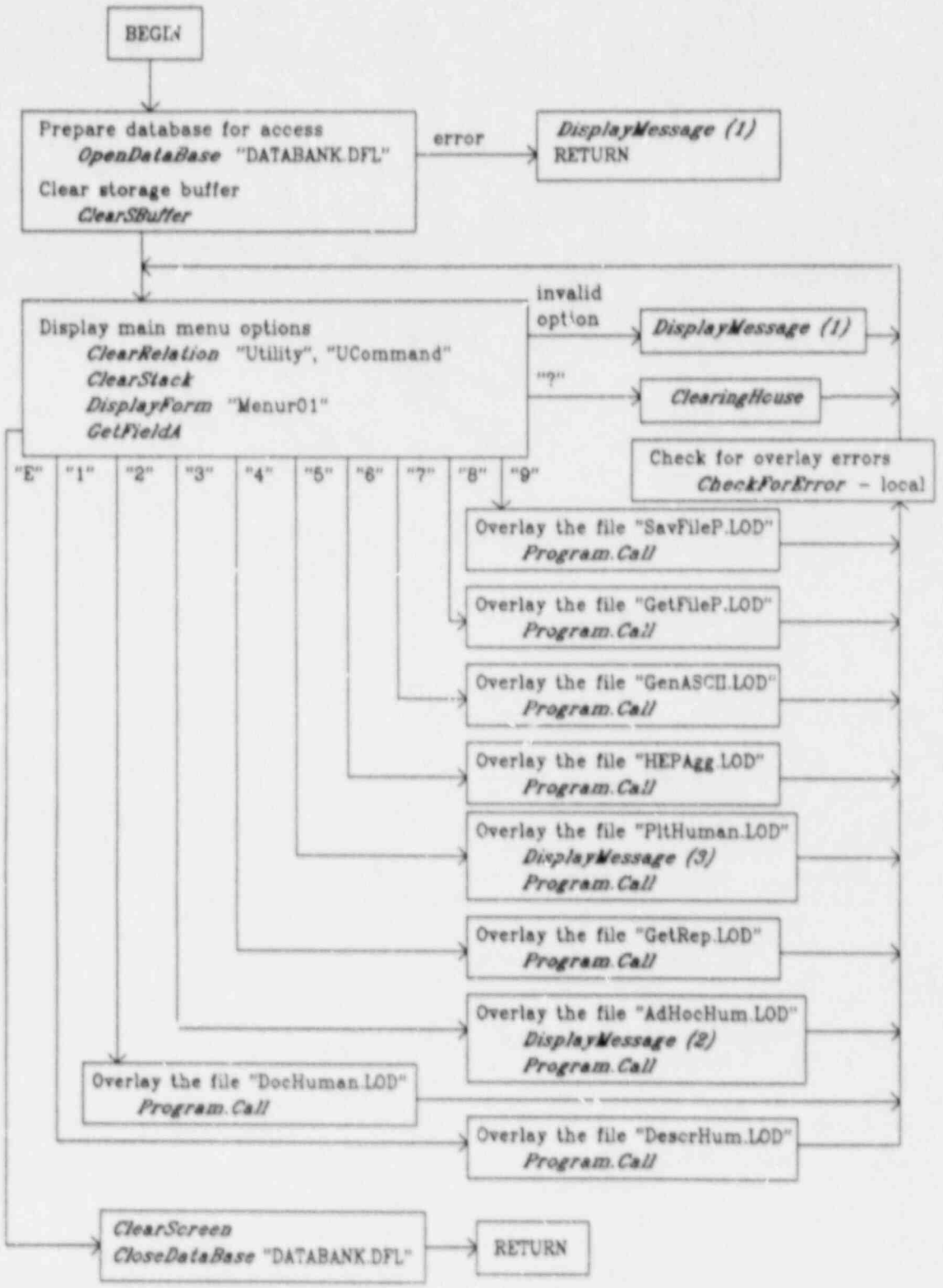
-None-

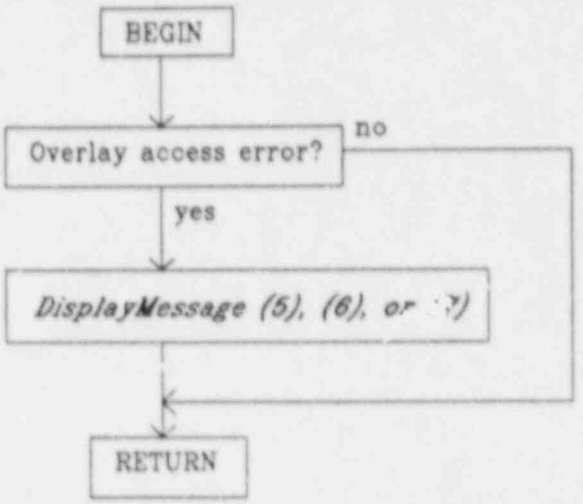
Procedures Imported

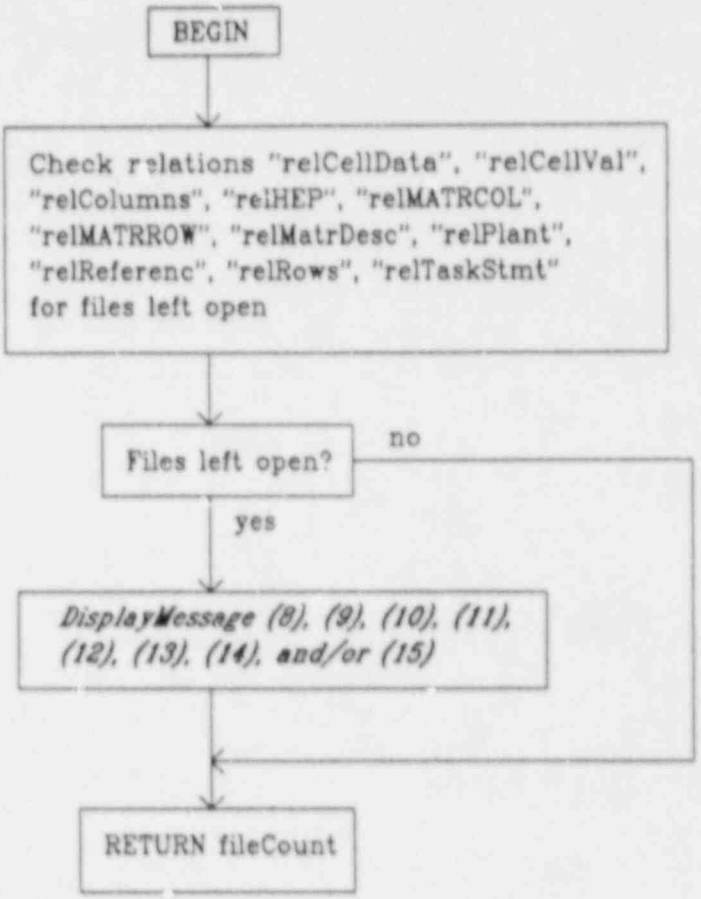
Call Program
Clearinghouse NUCGEN
ClearRelation Sage
ClearSBuffer StorageM
ClearScreen ThorPort
ClearStack NUCGEN
CloseDataBase Sage
DisplayForm Sage
DisplayMessage Sage
FindRecord Sage
GetFieldA Sage
OpenDataBase Sage
Pause ThorPort

Messages

(1) - "The HRA Data Bank can not be accessed"
(2) - "Preparing ad hoc search screen - standby . . ."
(3) - "Preparing for plot processing - standby . . ."
(4) - "Invalid option"
(5) - "Missing module"
(6) - "Version error"
(7) - "Selected overlay cannot be provided"
(8) - "CellData relation is open"
(9) - "CellVal relation is open"
(10) - "Columns relatino is open"
(11) - "HEP relation is open"
(12) - "MATRCOL relation is open"
(11) - "MATRROW relation is open"
(12) - "MatrDesc relation is open"
(13) - "Plant relation is open"
(14) - "Referenc relation is open"
(15) - "TaskStmt relation is open"


BEGIN
Prepare database for access
OpenDataBase "DATABANK.DFL"
Clear storage buffer
ClearSBuffer
error
DisplayMessage (1)
RETURN
Display main menu options
ClearRelation "Utility", "UCommand"
ClearStack
DisplayForm "Menur01"
GetFieldA
invalid option
DisplayMessage (1)
"?"
ClearingHouse
Check for overlay errors
CheckForError - local
"E"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"
Overlay the file "SavFileP.LOD"
Program.Call
Overlay the file "GetFileP.LOD"
Program.Call
Overlay the file "GenASCII.LOD"
Program.Call
Overlay the file "HEPAgg.LOD"
Program.Call
Overlay the file "PltHuman.LOD"
DisplayMessage (3)
Program.Call
Overlay the file "GetRep.LOD"
Program.Call
Overlay the file "AdHocHum.LOD"
DisplayMessage (2)
Program.Call
Overlay the file "DocHuman.LOD"
Program.Call
Overlay the file "DescrHum.LOD"
Program.Call
ClearScreen
CloseDataBase "DATABANK.DFL"
RETURN


BEGIN
Overlay access error?
no
yes
DisplayMessage (5), (6), or (7)
RETURN


BEGIN
Check relations "relCellData", "relCellVal", "relColumns", "relHEP", "relMATRCOL", "relMATRROW", "relMatrDesc", "relPlant", "relReferenc", "relRows", "relTaskStmt" for files left open
Files left open?
no
yes
DisplayMessage (8), (9), (10), (11), (12), (13), (14), and/or (15)
RETURN fileCount

APPENDIX B2

HEP DATA PROCESSING MODULE CHARTS

APPENDIX B2

HEP DATA PROCESSING MODULE CHARTS

This appendix contains high level flow charts for some modules used in HEP data processing. These charts provide an overview of what each module and each of its procedures is doing and how it is done; consequently, these charts can serve as road maps to the source code.

These charts are organized by module. For a particular module, the chart of the module is first, followed by charts of that module's procedures presented in alphabetical order. Three lists precede the charts for each module. An alphabetic, italicized list of procedures local to the module is given first. These local procedures are the procedures charted. The next list identifies procedures imported and names the library from which each procedure is imported. This list is alphabetized by procedure name with the procedure name italicized. If the libraries named here are not listed in Appendix A, they may be charted in Appendix B1. The final list is of messages referenced in the charts. Each message is assigned a number by which the messages are referenced and by which the list is ordered.

The elements in these charts are boxes and circles. Boxes contain descriptive comments on the processes being charted. Circled numbers are used to show chart continuation. Circled letters are used to show looping structures. Path lines, used to connect chart elements, have an arrow head at one end, thus showing the direction of activity flow from element to element.

Within boxes of the charts, procedure names are frequently used. These names are bolded and italicized. If a procedure name is considered descriptive enough, it may be used as a description of the process being shown; otherwise, the name will occur under a description and be indented showing that this procedure is used in the process described. If a local procedure is referenced, its name is followed by the dash character and then the word 'local'.

When multiple lines exit from a box, the condition determining the use of a path line will be indicated at the line. Any unlabeled path line exiting a box indicates the normal path to be taken.

When data base forms or relations are mentioned, they are shown within quotation marks.

MODULE Retrieve

Local Procedures

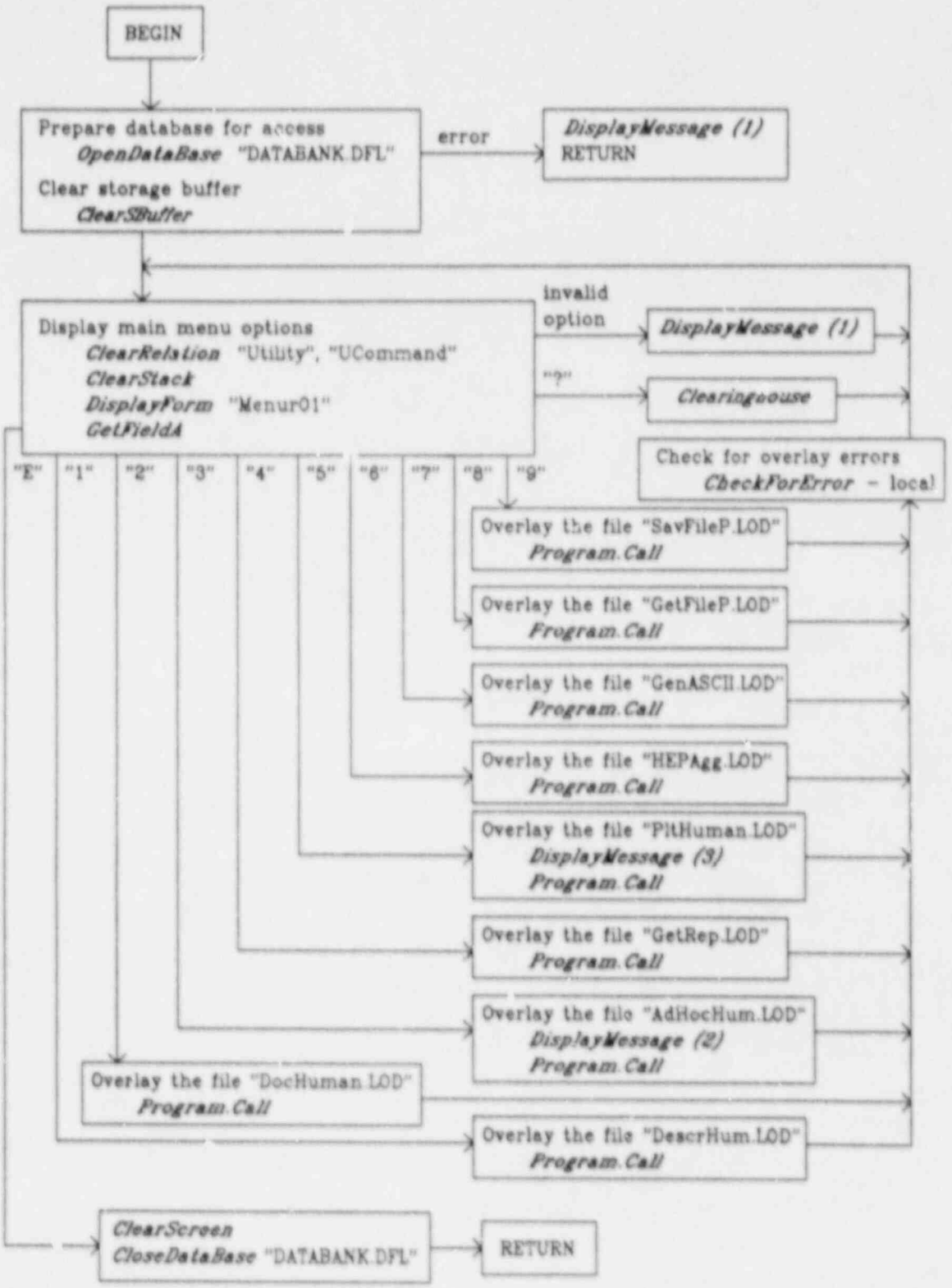
CheckForError
FilesLeftOpen

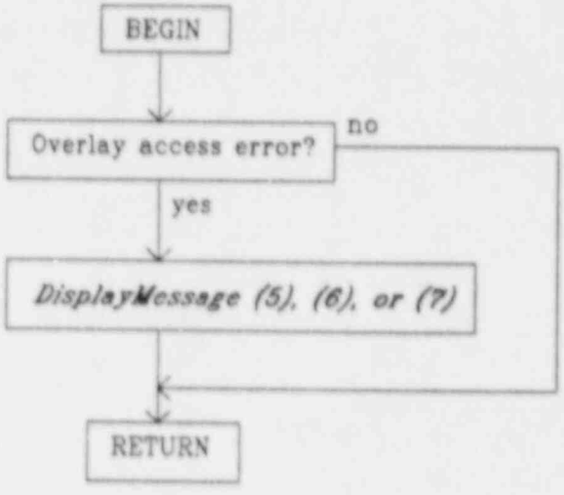
Procedures Imported

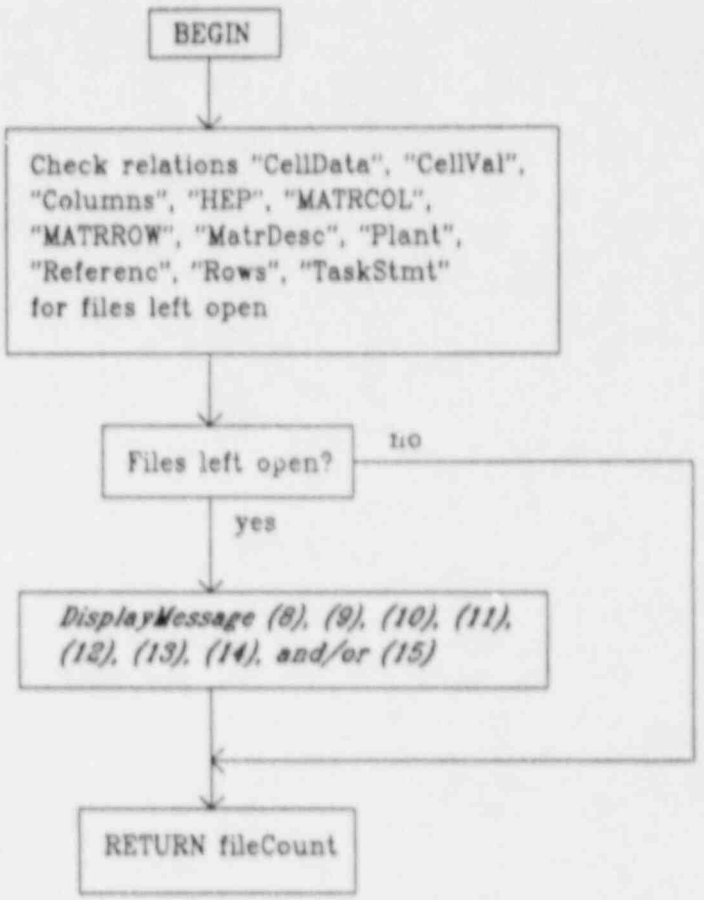
Call Program
Clearinghouse NUCGEN
ClearRelation Sage
ClearSBuffer StorageM
ClearScreen ThorPort
ClearStack NUCGEN
CloseDataBase Sage
DisplayForm Sage
DisplayMessage Sage
FindRecord Sage
GetFieldA Sage
OpenDataBase Sage
Pause ThorPort

Messages

(1) - "The HRA Data Bank can not be accessed"
(2) - "Preparing ad hoc search screen - standby . . ."
(3) - "Preparing for plot processing - standby . . ."
(4) - "Invalid option"
(5) - "Missing module"
(6) - "Version error"
(7) - "Selected overlay cannot be provided"
(8) - "CellData relation is open"
(9) - "CellVal relation is open"
(10) - "Columns relatino is open"
(11) - "MEP relation is open"
(12) - "MATRCOL relation is open"
(11) - "MATRROW relation is open"
(12) - "MatrDesc relation is open"
(13) - "Plant relation is open"
(14) - "Referenc relation is open"
(15) - "TaskStmt relation is open"


BEGIN
Prepare database for access
OpenDataBase "DATABANK.DFL"
Clear storage buffer
ClearSBuffer
error
DisplayMessage (1)
RETURN
Display main menu options
ClearRelation "Utility", "UCommand"
ClearStack
DisplayForm "Menur01"
GetFieldA
invalid option
DisplayMessage (1)
"?"
Clearingoouse
Check for overlay errors
CheckForError - local
"E"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"
Overlay the file "SavFileP.LOD"
Program.Call
Overlay the file "GetFileP.LOD"
Program.Call
Overlay the file "GenASCII.LOD"
Program.Call
Overlay the file "HEPAgg.LOD"
Program.Call
Overlay the file "PltHuman.LOD"
DisplayMessage (3)
Program.Call
Overlay the file "GetRep.LOD"
Program.Call
Overlay the file "AdHocHum.LOD"
DisplayMessage (2)
Program.Call
Overlay the file "DocHuman.LOD"
Program.Call
Overlay the file "DescrHum.LOD"
Program.Call
ClearScreen
CloseDataBase "DATABANK.DFL"
RETURN


BEGIN
Overlay access error?
no
yes
DisplayMessage (5), (6), or (7)
RETURN


BEGIN
Check relations "CellData", "CellVal", "Columns", "HEP", "MATRCOL", "MATRROW", "MatrDesc", "Plant", "Referenc", "Rows", "TaskStmt" for files left open
Files left open?
no
yes
DisplayMessage (8), (9), (10), (11), (12), (13), (14), and/or (15)
RETURN fileCount

MODUL

Local Procedures

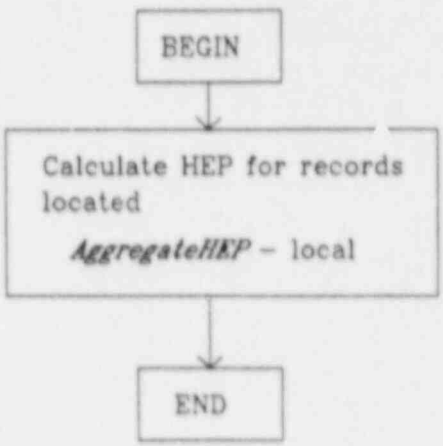
AggregateHEP
CalculateForTaskInBuffer
GetCellDataRecord
ReadCellDataRecord

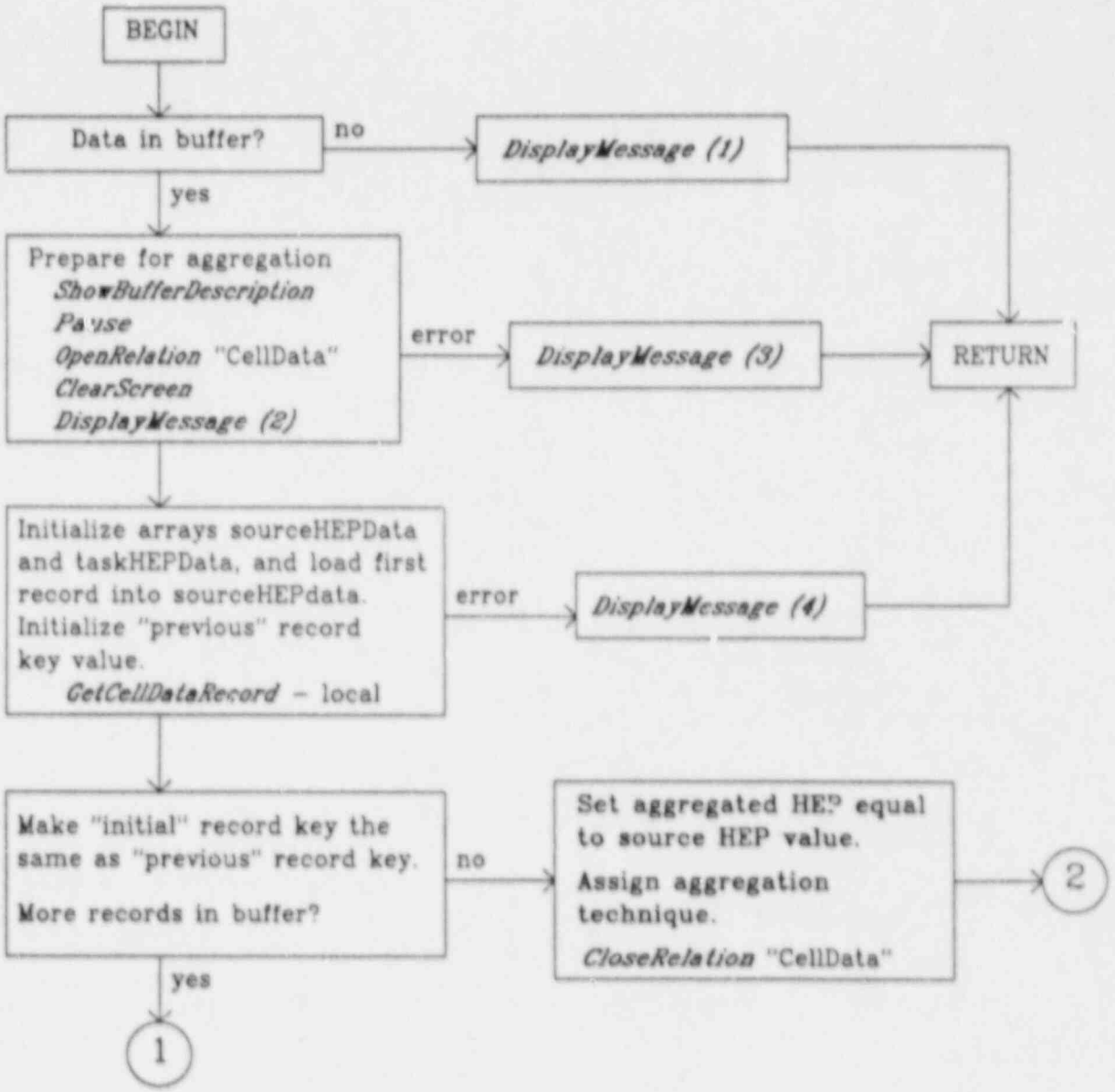
Procedures Imported

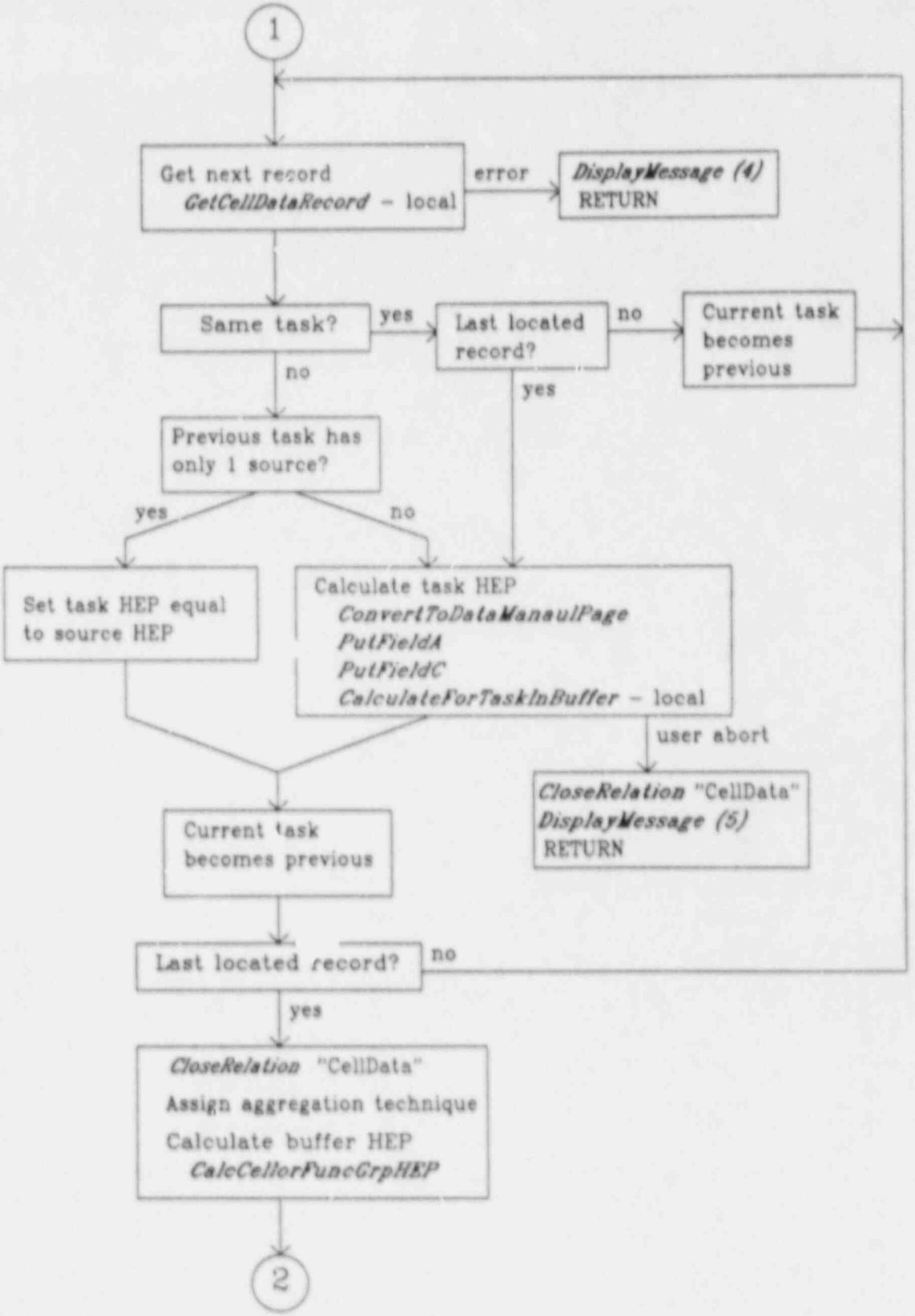
CalcCellorFuncGrpHEP Calc
CalcTaskHEP Calc
Clearinghouse NUCGEN
ClearScreen ThorPort
ClearStack NUCGEN
Close Files
CloseDataBase Sage
CloseRelation Sage
CloseRelationFiles Sage
CloseReport Reports
CompareKey SortLib
Concat String
CondRead Terminal
ConvertToDataManualPage NUCGEN
CreateBuffrReport BReports
DecodeCellDataKey StorageM
DefineReport Reports
DisplayForm Sage
DisplayMessage Sage
DisplaySource NUCPRINT
FillChar MoveLib
FindRecord Sage
GetFieldA Sage
GetFieldC Sage
GetFieldF Sage
Lookup DiskLib
OpenDataBase Sage
OpenRelation Sage
OpenReport Reports
Pause ThorPort
ReadBytes Binary
ReadRecord Sage
ReadRecordC Sage
ReadRecordR Sage
Remove Files
Reset Files
Rewrite Files
ShowBufferDescription StorageM
TopOfPage Reports
WrForm Reports
WriteBytes Binary

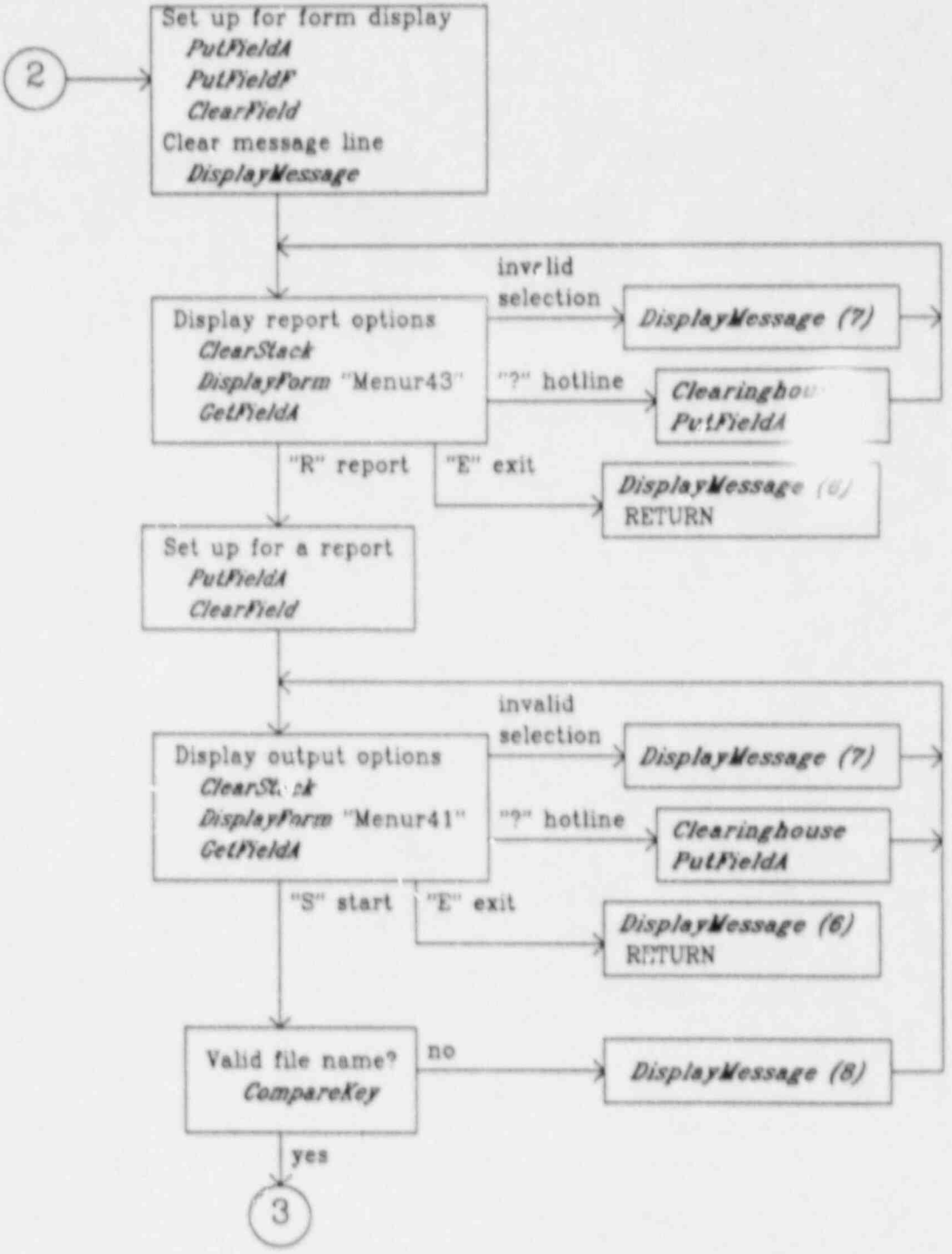
Messages

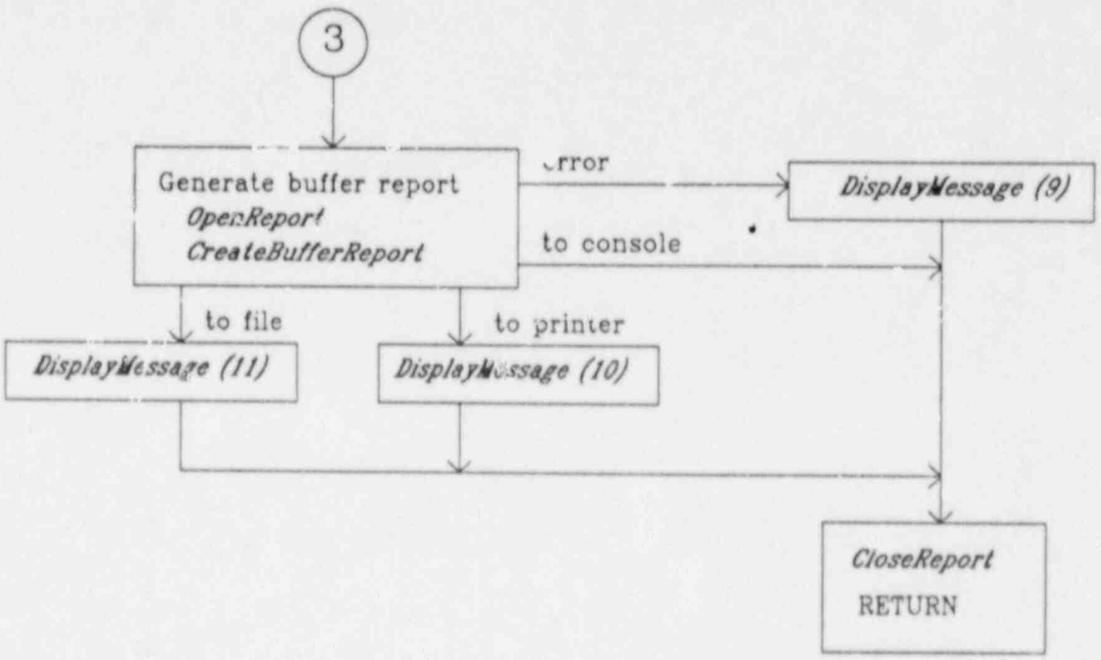
(1) - "No data in buffer, therefore no aggregation"
(2) - "Aggregation in process - please standby ..."
(3) - "Failed opening cell data for HEP aggregation"
(4) - "Can not access one of the located data records"
(5) - "Aggregation of located records stopped as requested"
(6) - "Buffer aggregation completed, no report was requested"
(7) - "Invalid selection"
(8) - "A valid DOS file name is needed for this option"
(9) - "Error trying to open report file"
(10) - "Report with aggregate HEP sent to printer"
(11) - "Report with aggregate HEP sent to file 'xxxxxx'"
(12) - "Failed in reading cell data for HEP aggregation"

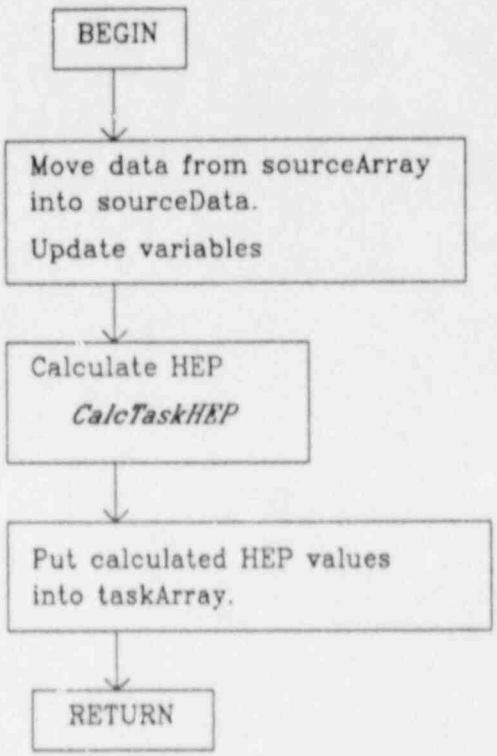

BEGIN
Calculate HEP for records located
AggregateHEP – local
END

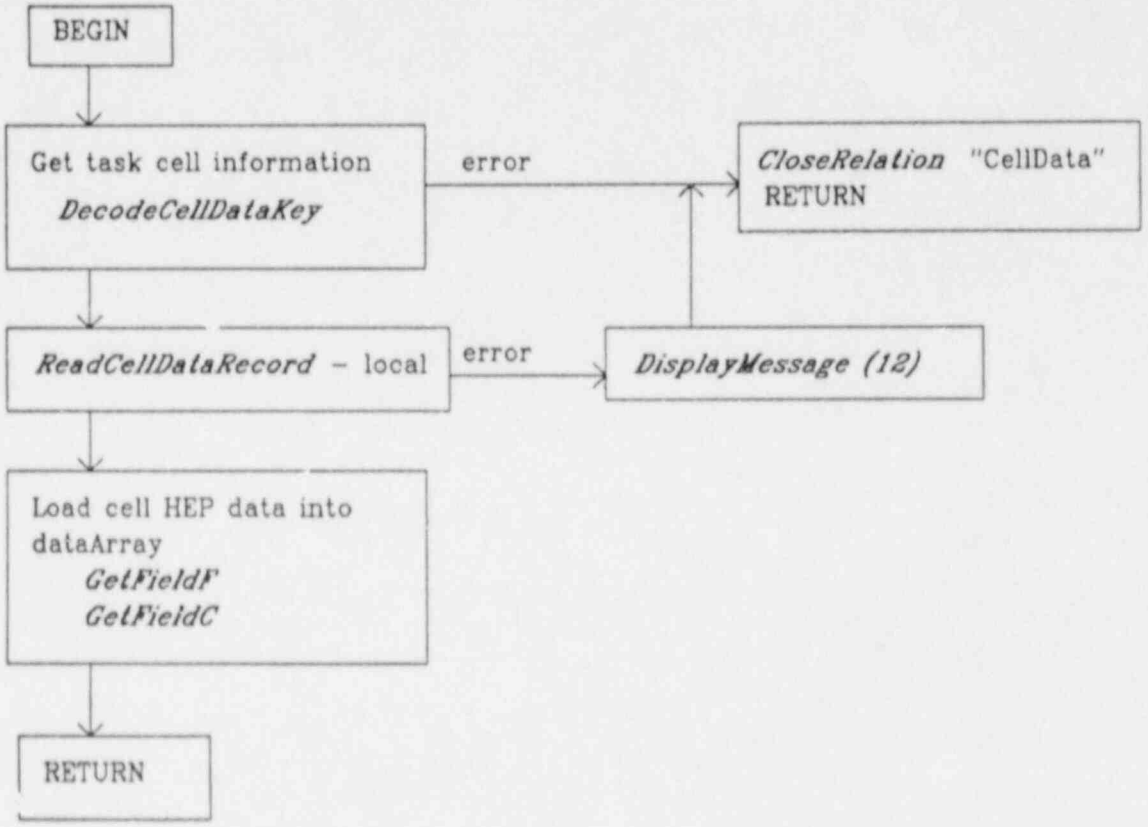

BEGIN
Data in buffer?
no
DisplayMessage (1)
yes
Prepare for aggregation
ShowBufferDescription
Pause
OpenRelation "CellData"
ClearScreen
DisplayMessage (2)
error
DisplayMessage (3)
RETURN
Initialize arrays sourceHEPData and taskHEPData, and load first record into sourceHEPdata.
Initialize "previous" record key value.
GetCellDataRecord - local
error
DisplayMessage (4)
Make "initial" record key the same as "previous" record key.
More records in buffer?
no
Set aggregated HEP equal to source HEP value.
Assign aggregation technique.
CloseRelation "CellData"
2
yes
1

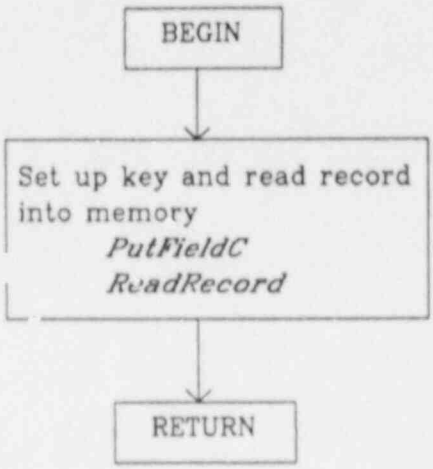

1
Get next record
GetCellDataRecord - local
error
DisplayMessage (4)
RETURN
Same task?
yes
Last located record?
no
Current task becomes previous
no
yes
Previous task has only 1 source?
yes
no
Set task HEP equal to source HEP
Calculate task HEP
ConvertToDataManualPage
PutFieldA
PutFieldC
CalculateForTaskInBuffer - local
user abort
CloseRelation "CellData"
DisplayMessage (5)
RETURN
Current task becomes previous
Last located record?
no
yes
CloseRelation "CellData"
Assign aggregation technique
Calculate buffer HEP
CalcCellorFuncGrpHEP
2


2
Set up for form display
PutFieldA
PutFieldF
ClearField
Clear message line
DisplayMessage
Display report options
ClearStack
DisplayForm "Menur43"
GetFieldA
invalid selection
DisplayMessage (7)
"?" hotline
Clearinghouse
PutFieldA
"R" report
"E" exit
DisplayMessage (6)
RETURN
Set up for a report
PutFieldA
ClearField
Display output options
ClearStack
DisplayForm "Menur41"
GetFieldA
invalid selection
DisplayMessage (7)
"?" hotline
Clearinghouse
PutFieldA
"S" start
"E" exit
DisplayMessage (6)
RETURN
Valid file name?
CompareKey
no
DisplayMessage (8)
yes
3


3
Generate buffer report
OpenReport
CreateBufferReport
Error
DisplayMessage (9)
to console
to file
to printer
DisplayMessage (11)
DisplayMessage (10)
CloseReport
RETURN


BEGIN
Move data from sourceArray into sourceData.
Update variables
Calculate HEP
CalcTaskHEP
Put calculated HEP values into taskArray.
RETURN


BEGIN
Get task cell information
DecodeCellDataKey
error
CloseRelation "CellData"
RETURN
ReadCellDataRecord - local
error
DisplayMessage (12)
Load cell HEP data into
dataArray
GetFieldF
GetFieldC
RETURN


BEGIN
Set up key and read record into memory
PutFieldC
ReadRecord
RETURN

APPENDIX C1

HCF DATA PROCESSING LIBRARY CHARTS

APPENDIX C1

HCF DATA PROCESSING LIBRARY CHARTS

This appendix contains high level flow charts for some libraries used in HCF data processing. Only charts for those libraries programmed explicitly for NUCLARR are included. These charts provide an overview of what each procedure is doing and how it is done; consequently, these charts can serve as road maps to the source code.

These charts are organized by library, with the charts of a library's procedures presented in alphabetical order. Three lists precede the charts for each library. An alphabetic, italicized list of procedures local to the library is given first. This list is subdivided into exported procedures and those which are not exported. These local procedures are the procedures charted. The next list identifies procedures imported and names the library from which each procedure is imported. This list is alphabetized by procedure name with the procedure name italicized. If the libraries named here are not listed in Appendix A, they may be charted in this appendix. The final list is of messages referenced in the charts. Each message is assigned a number by which the messages are referenced and by which the list is ordered.

The elements in these charts are boxes and circles. Boxes contain descriptive comments on the processes being charted. Circled numbers are used to show chart continuation. Circled letters are used to show looping structures. Path lines, used to connect chart elements, have an arrow head at one end, thus showing the direction of activity flow from element to element.

Within boxes of the charts, procedure names are frequently used. These names are bolded and italicized. If a procedure name is considered descriptive enough, it may be used as a description of the process being shown; otherwise, the name will occur under a description and be indented showing that this procedure is used in the process described. If a local procedure is referenced, its name is followed by the dash character and then the word 'local'.

When multiple lines exit from a box, the condition determining the use of a path line will be indicated at the line. Any unlabeled path line exiting a box indicates the normal path to be taken.

When data base forms or relations are mentioned, they are shown within quotation marks.

LIBRARY ClrHouse

Local Procedures

Exported

Clearinghouse

Other

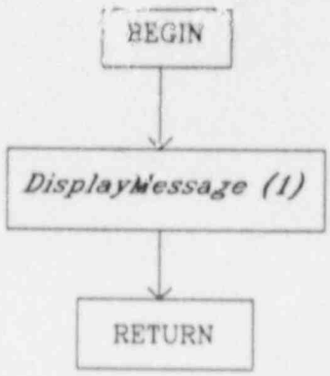
-none-

Procedures Imported

DisplayMessage Sage

Messages

(1) - ''Telephone Clearinghouse for assistance at (208)526-0735 or FTS 583-0735''


BEGIN
DisplayMessage (1)
RETURN

LIBRARY General

Local Procedures

Exported

Among
BlankData
CheckRelationStatus
ClearMessage
ClearStack
ClearUEvent
ConvertMilitaryDate
DisplaySageError
FailureGroupMember
GetCatName
GetFailureGroups
MilitaryDate
MyCloseRelation
MyOpenRelation
PlacesC
ShowSageError

Other

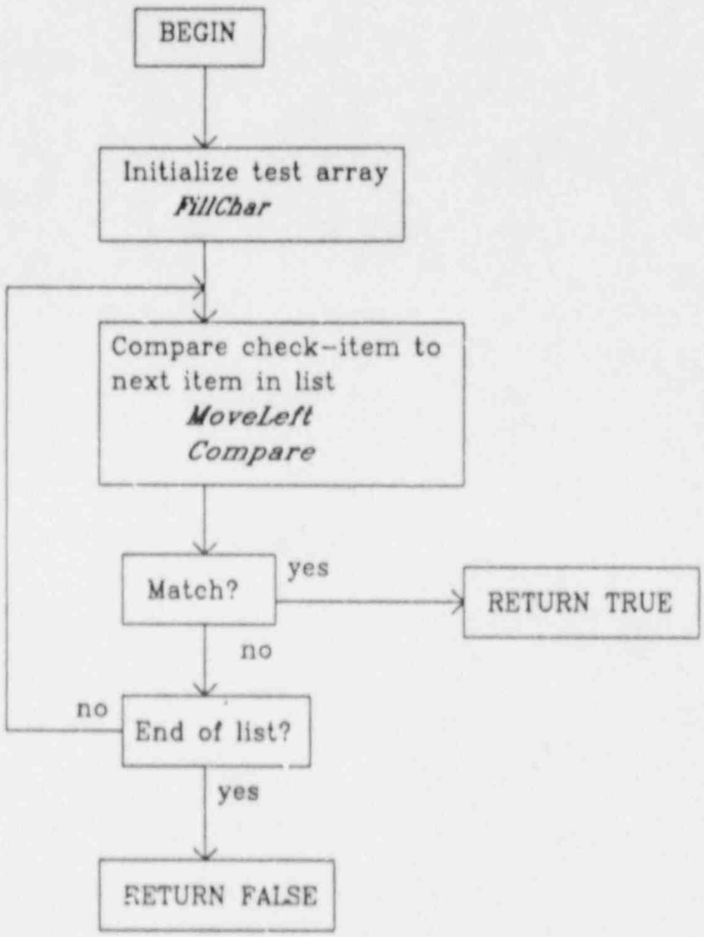
-none-

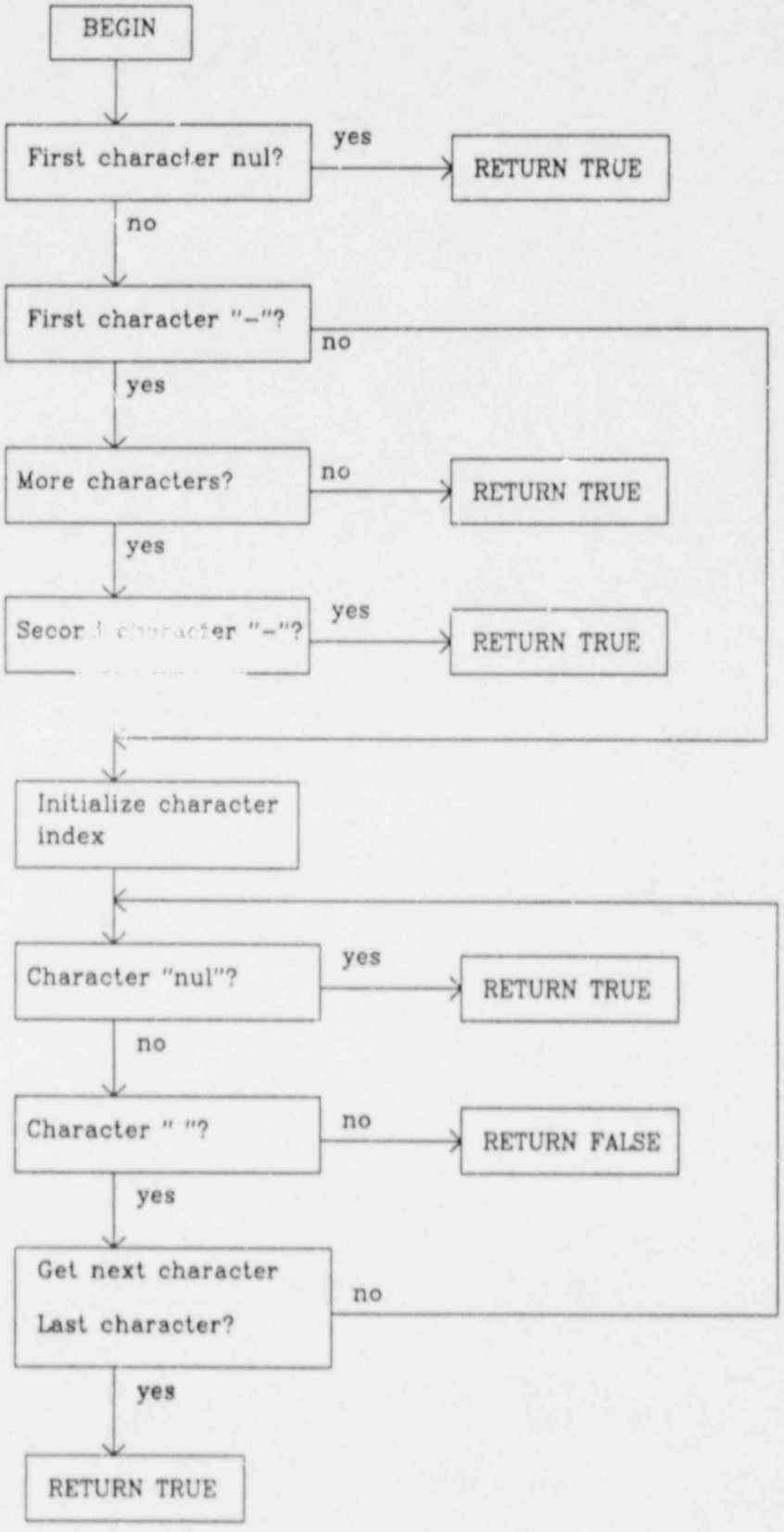
Procedures Imported

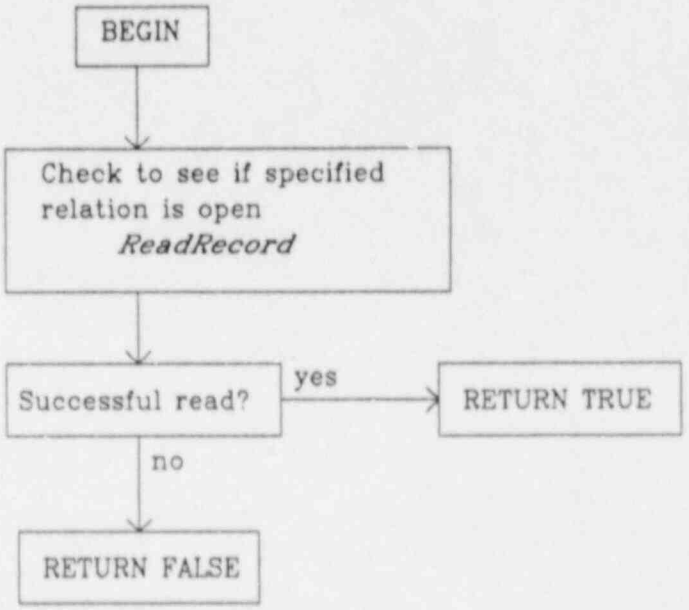
ADR SYSTEM
CardToStr Convert
ClearField Sage
ClearRelation Sage
ClearScreen ThorPort
CloseDataBase Sage
CloseRelation Sage
CloseRelationFiles Sage
Compare String
CompareFieldA Sage
CompareFieldC Sage
CompareKey SortLib
Concat String
CondRead Terminal
CopyField Sage
CurrentDate1 TimeLib
CursorMove ThorPort
DefineFieldCheck Sage
DeleteRecord Sage
DeleteRelation Sage
DisplayForm Sage
DisplayFormV Sage
DisplayFormVIP Sage
DisplayMessage Sage
FillChar SortLib
FindRecord Sage
FixFileName ThorPort
GetBlock Sage
GetFieldA Sage
GetFieldB Sage
GetFieldC Sage
GetFieldF Sage
GetFieldI Sage
GetFieldP Sage
GetRepeatName Sage
Length String
MoveLeft SortLib
MoveString ThorUtil
OpenDataBase Sage
OpenRelation Sage
Pause ThorPort
Position String
PutFieldA Sage
PutFieldC Sage
PutFieldF Sage
PutFieldI Sage
ReadRecord Sage
RearrecordA Sage
RelationIsOpen Sage
ReWriteRecord Sage
ScanChar SortLib
Sort SortLib
StrToCard Convert
Substring String
WriteLn Terminal
WriteRecord Sage
WriteString Terminal

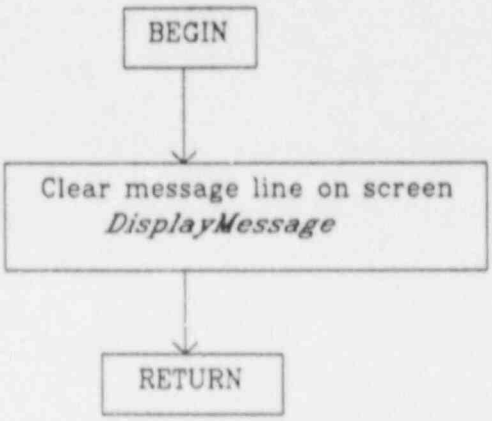
Messages

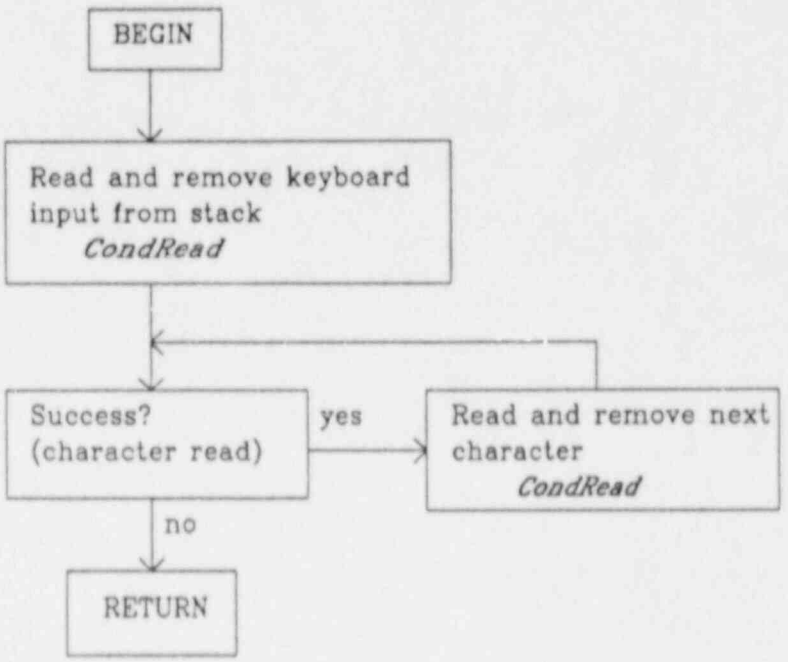
-none-

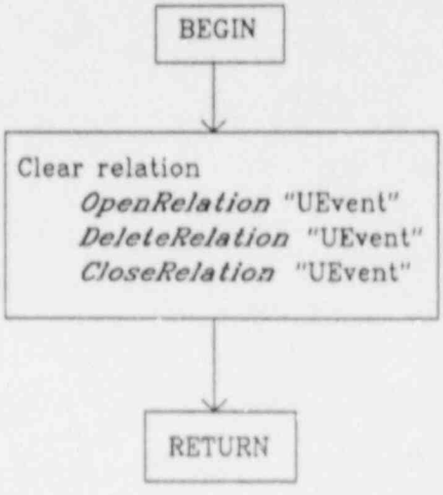

BEGIN
Initialize test array
FillChar
Compare check-item to
next item in list
MoveLeft
Compare
Match?
yes
RETURN TRUE
no
no
End of list?
yes
RETURN FALSE

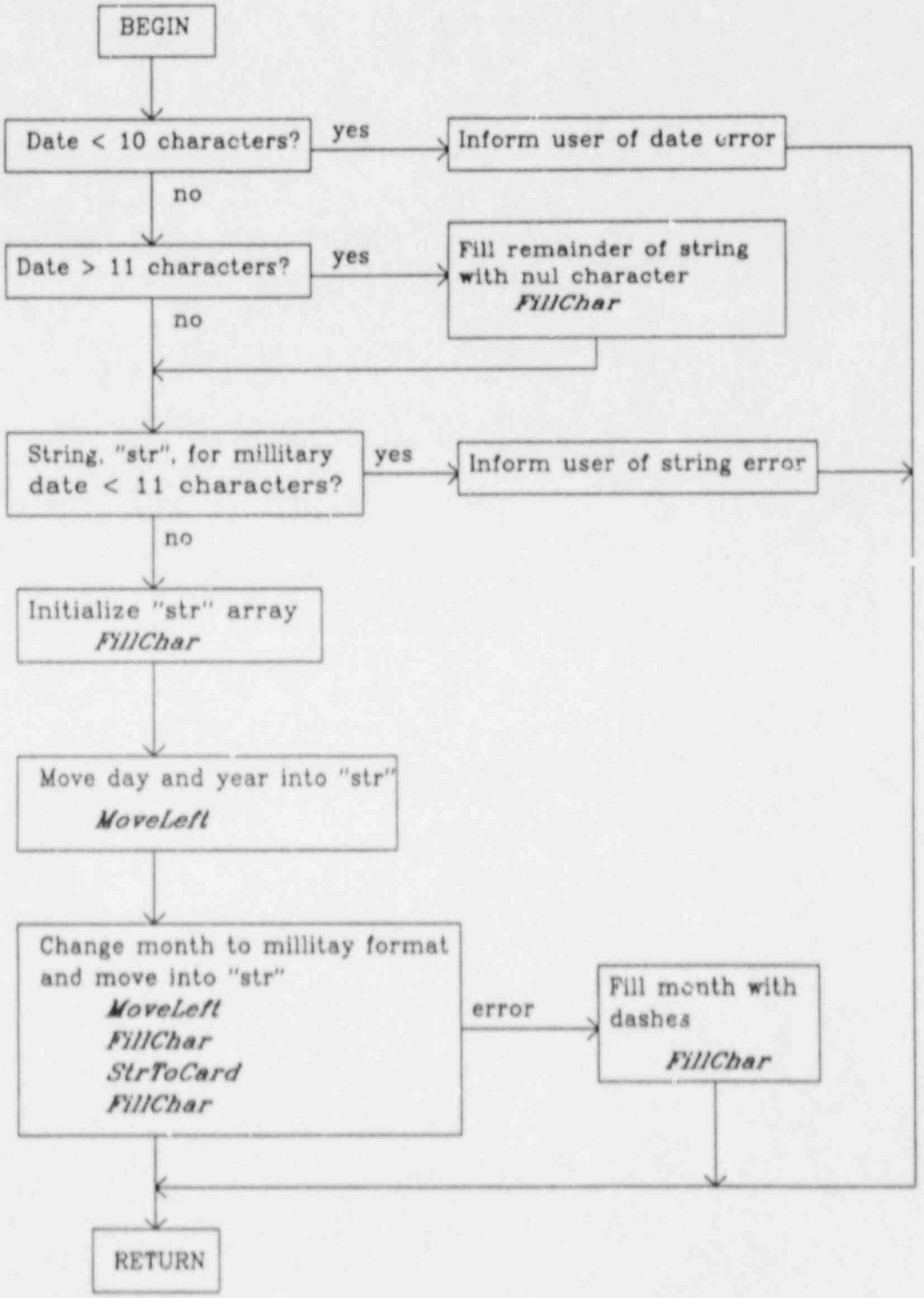

BEGIN
First character nul?
yes
RETURN TRUE
no
First character "-"?
no
yes
More characters?
no
RETURN TRUE
yes
Second character "-"?
yes
RETURN TRUE
Initialize character index
Character "nul"?
yes
RETURN TRUE
no
Character " "?
no
RETURN FALSE
yes
Get next character
Last character?
no
yes
RETURN TRUE

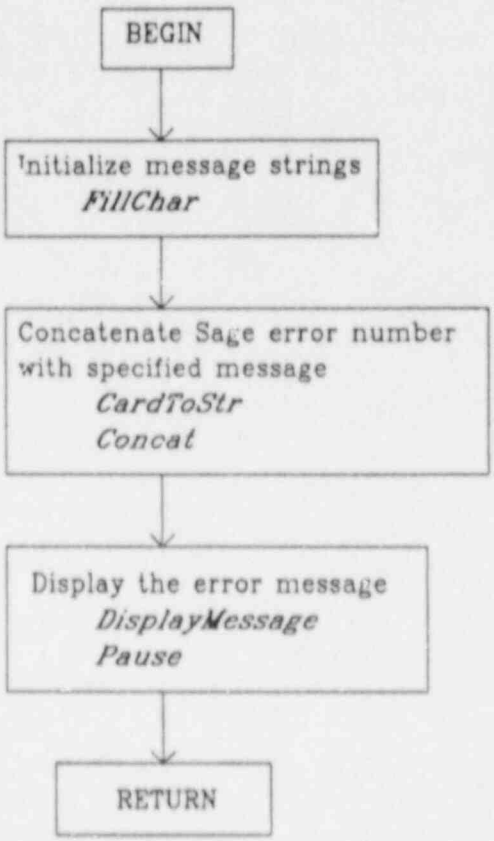

BEGIN
Check to see if specified relation is open
ReadRecord
Successful read?
yes
RETURN TRUE
no
RETURN FALSE

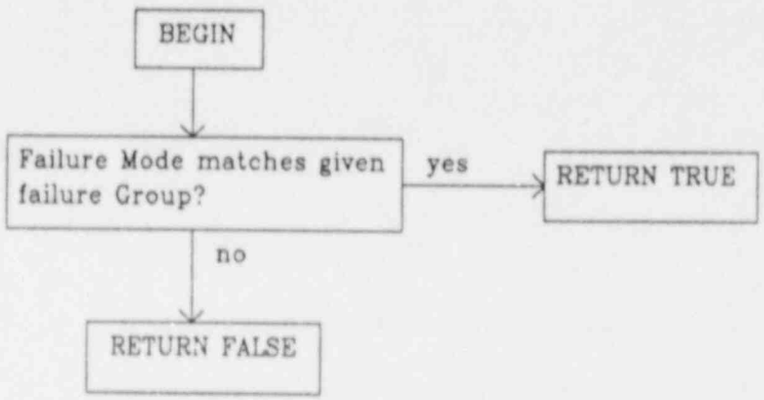

BEGIN
Clear message line on screen
DisplayMessage
RETURN

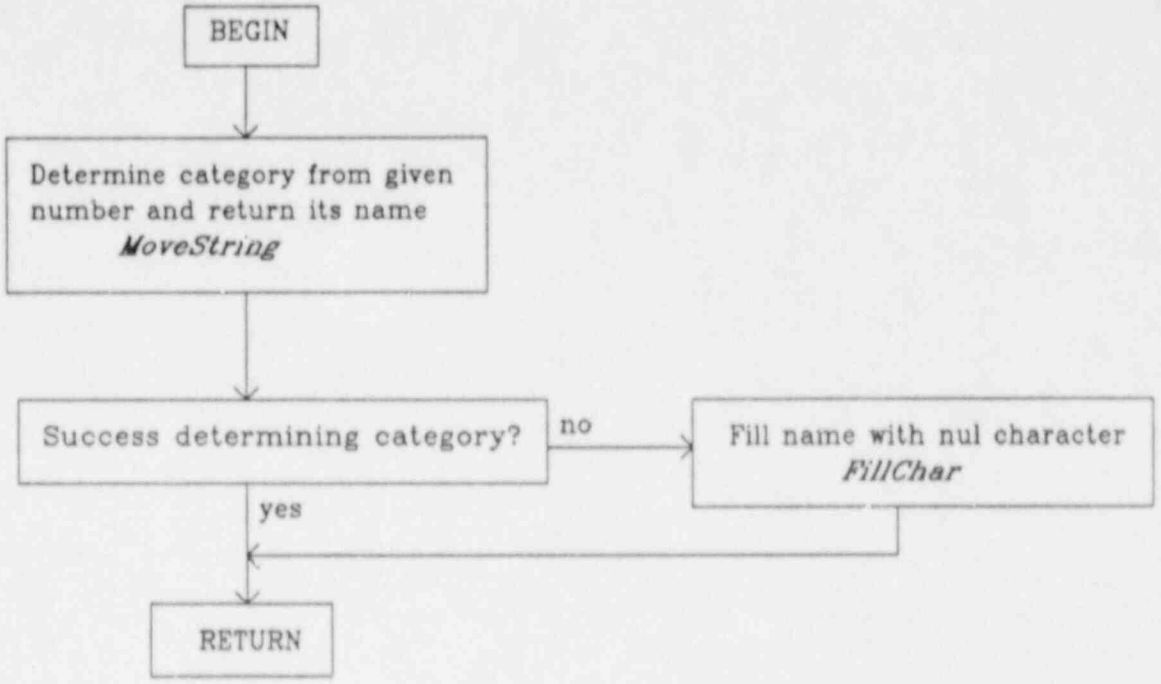

BEGIN
Read and remove keyboard input from stack
CondRead
Success? (character read)
yes
Read and remove next character
CondRead
no
RETURN

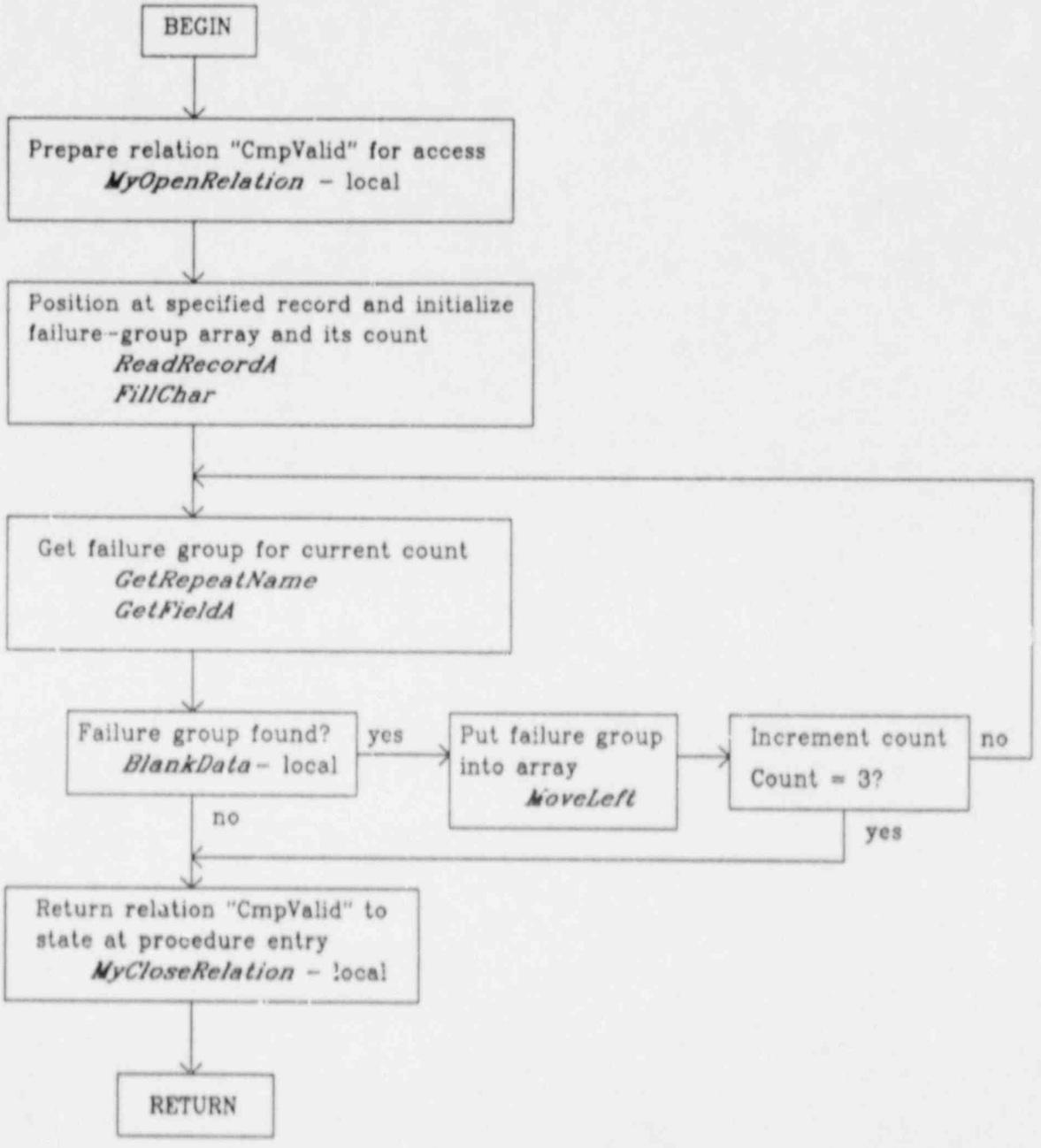

BEGIN
Clear relation
OpenRelation "UEvent"
DeleteRelation "UEvent"
CloseRelation "UEvent"
RETURN

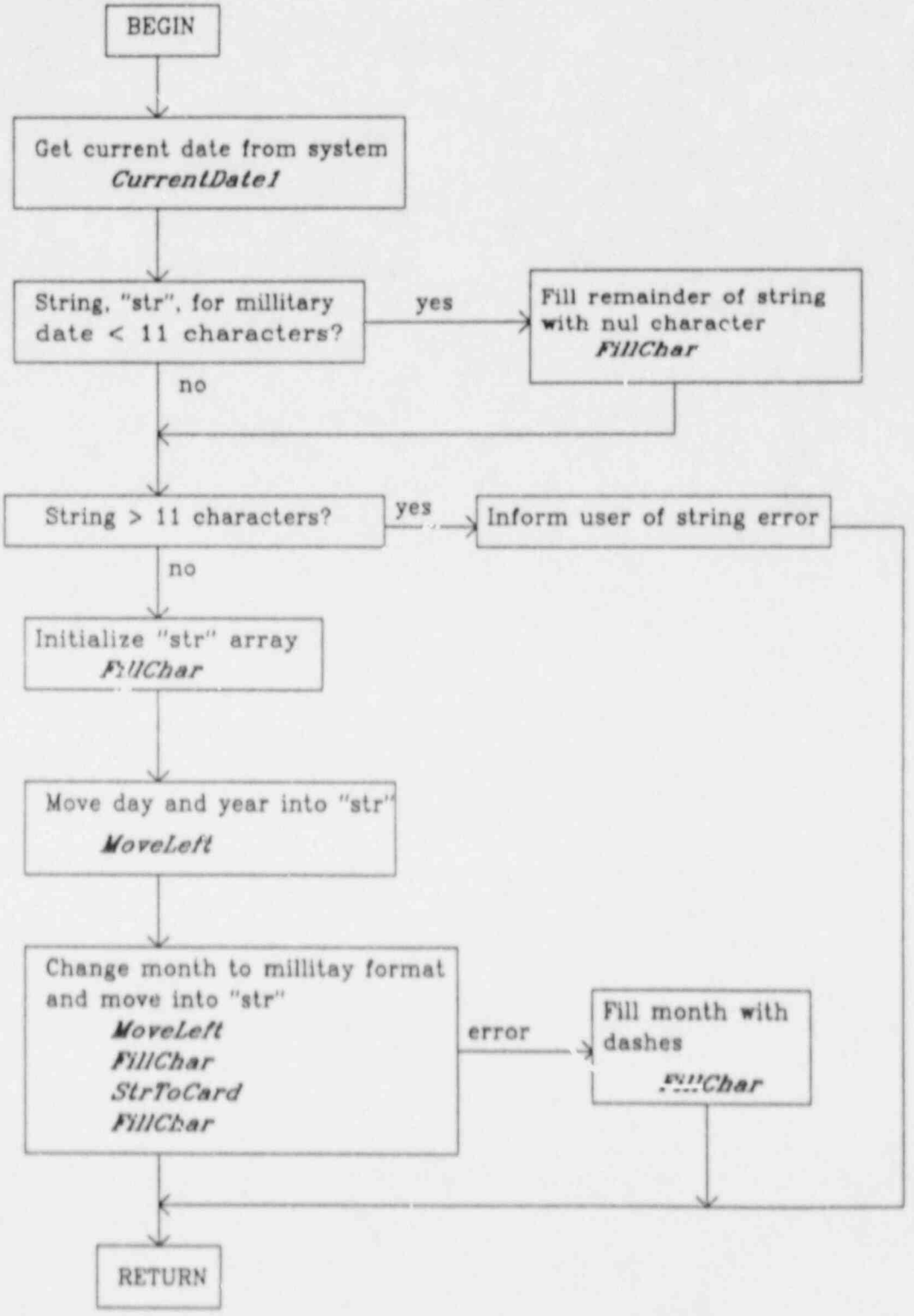

BEGIN
Date < 10 characters?
yes
Inform user of date error
no
Date > 11 characters?
yes
Fill remainder of string with nul character
FillChar
no
String, "str", for millitary date < 11 characters?
yes
Inform user of string error
no
Initialize "str" array
FillChar
Move day and year into "str"
MoveLeft
Change month to millitay format and move into "str"
MoveLeft
FillChar
StrToCard
FillChar
error
Fill month with dashes
FillChar
RETURN

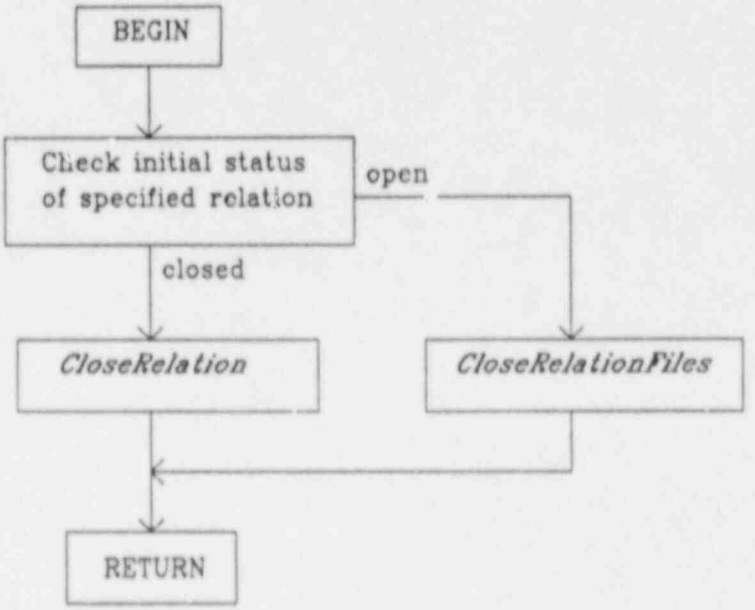

BEGIN
Initialize message strings
FillChar
Concatenate Sage error number
with specified message
CardToStr
Concat
Display the error message
DisplayMessage
Pause
RETURN

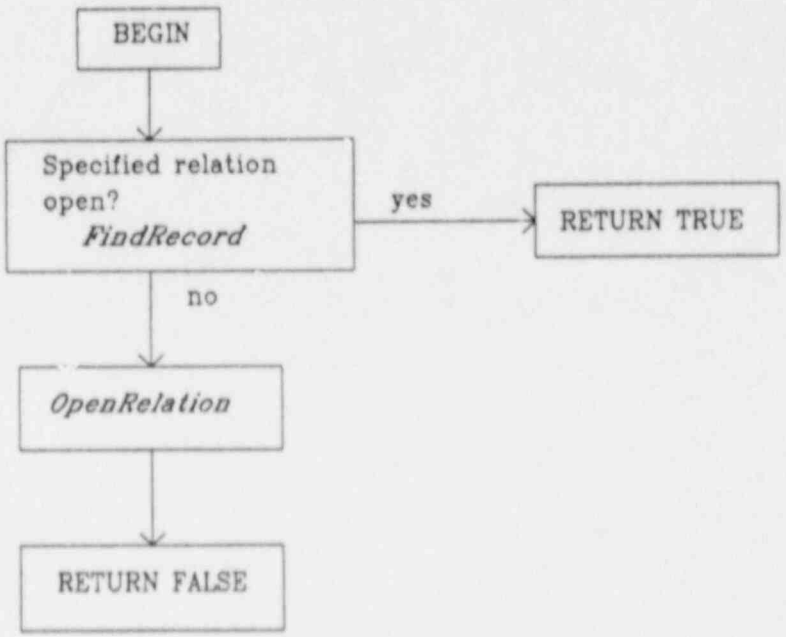

BEGIN
Failure Mode matches given failure Group?
yes
RETURN TRUE
no
RETURN FALSE

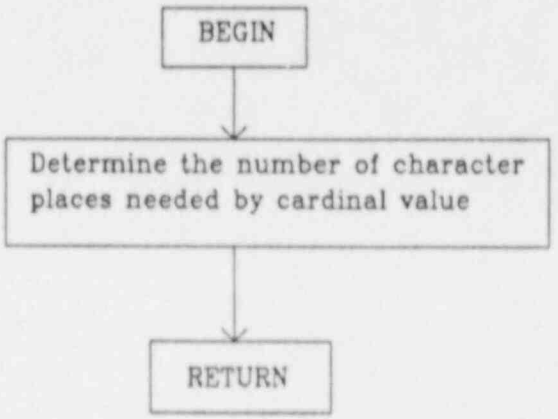

BEGIN
Determine category from given number and return its name
MoveString
Success determining category?
no
Fill name with nul character
FillChar
yes
RETURN

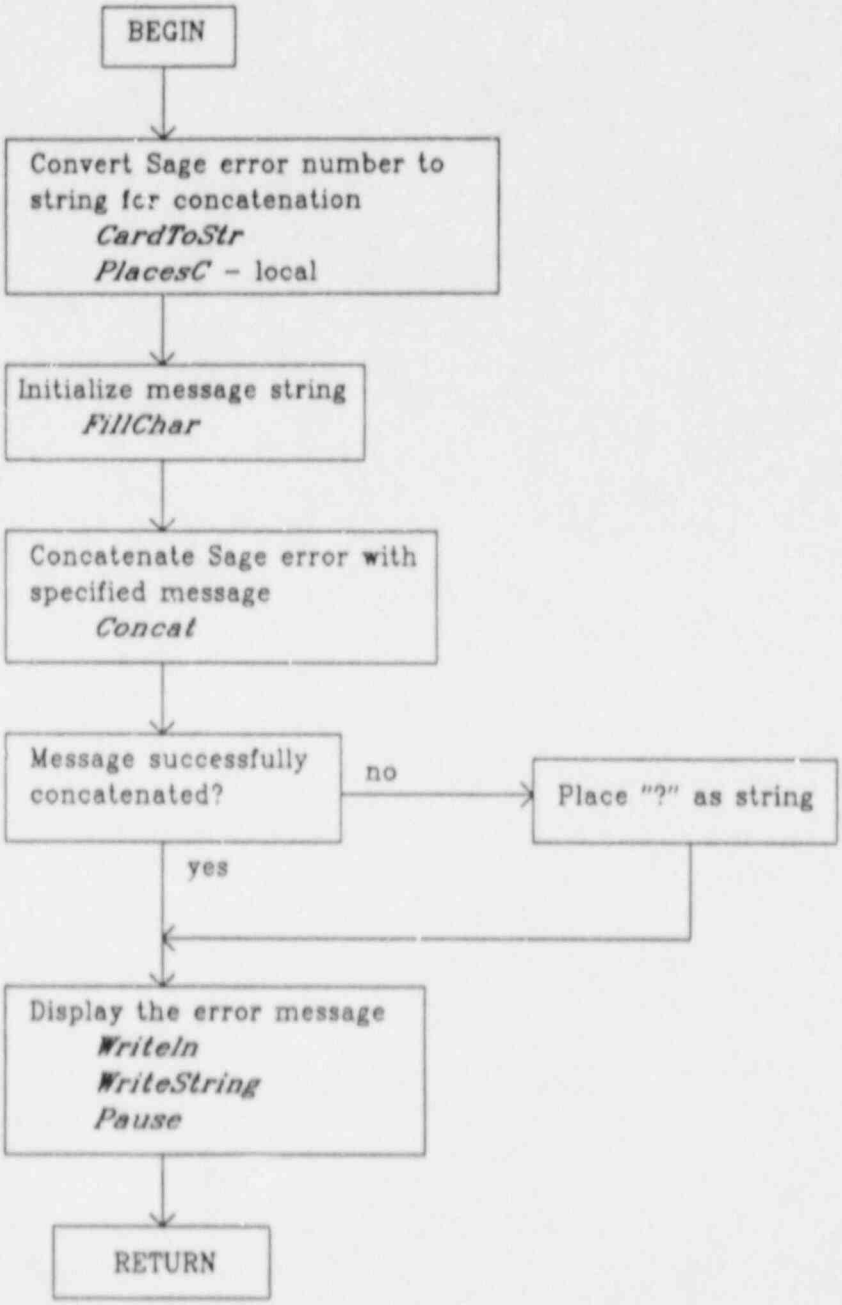

BEGIN
Prepare relation "CmpValid" for access
MyOpenRelation - local
Position at specified record and initialize failure-group array and its count
ReadRecordA
FillChar
Get failure group for current count
GetRepeatName
GetFieldA
Failure group found?
BlankData - local
yes
Put failure group into array
MoveLeft
Increment count
Count = 3?
no
yes
no
Return relation "CmpValid" to state at procedure entry
MyCloseRelation - local
RETURN


BEGIN
Get current date from system
CurrentDate1
String, "str", for millitary date < 11 characters?
yes
Fill remainder of string with nul character
FillChar
no
String > 11 characters?
yes
Inform user of string error
no
Initialize "str" array
FillChar
Move day and year into "str"
MoveLeft
Change month to millitay format and move into "str"
MoveLeft
FillChar
StrToCard
FillChar
error
Fill month with dashes
FillChar
RETURN


BEGIN
Check initial status of specified relation
open
closed
CloseRelation
CloseRelationFiles
RETURN


BEGIN
Specified relation open?
FindRecord
yes
RETURN TRUE
no
OpenRelation
RETURN FALSE


BEGIN
Determine the number of character places needed by cardinal value
RETURN


BEGIN
Convert Sage error number to string for concatenation
CardToStr
PlacesC - local
Initialize message string
FillChar
Concatenate Sage error with specified message
Concat
Message successfully concatenated?
no
Place "?" as string
yes
Display the error message
WriteIn
WriteString
Pause
RETURN

LIBRARY StoreMan

Local Procedures

Exported

ClearSBuffer
FreezeSBuffer
IncludeNewKey
RefreshSBuffer
RemoveKey
RetrieveKey

Other

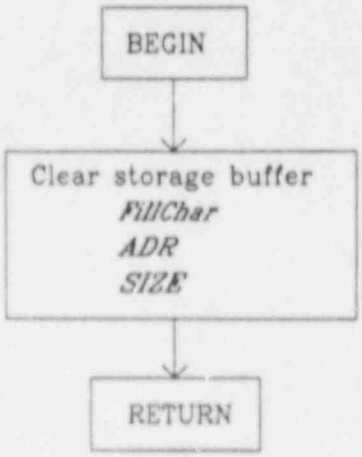
-none-

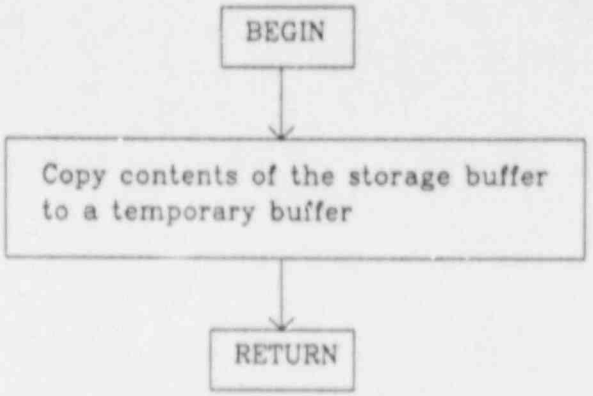
Procedures Imported

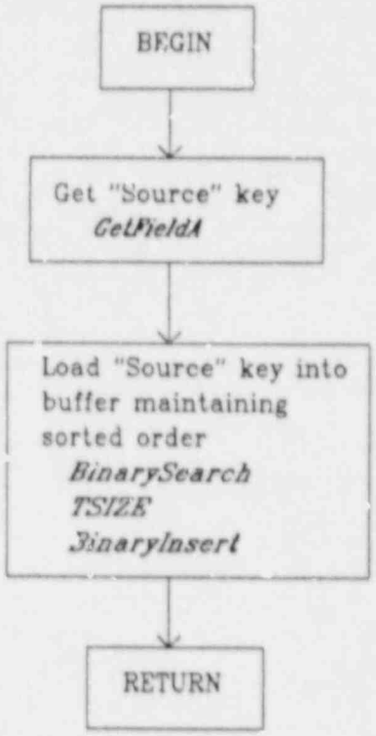
ADR SYSTEM
BinaryDelete SortLib
BinaryInsert SortLib
BinarySearch SortLib
FillChar MoveLib
GetFieldA Sage
SIZE SYSTEM
TSIZE SYSTEM

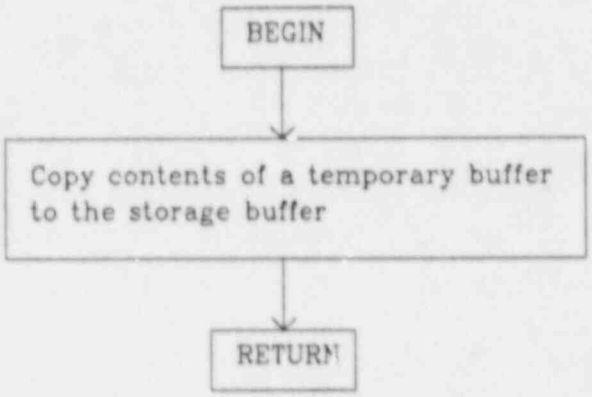
Messages

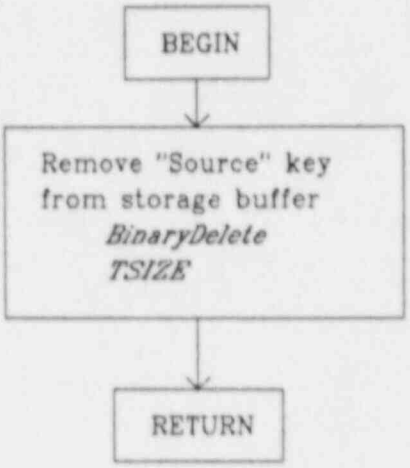
-none-

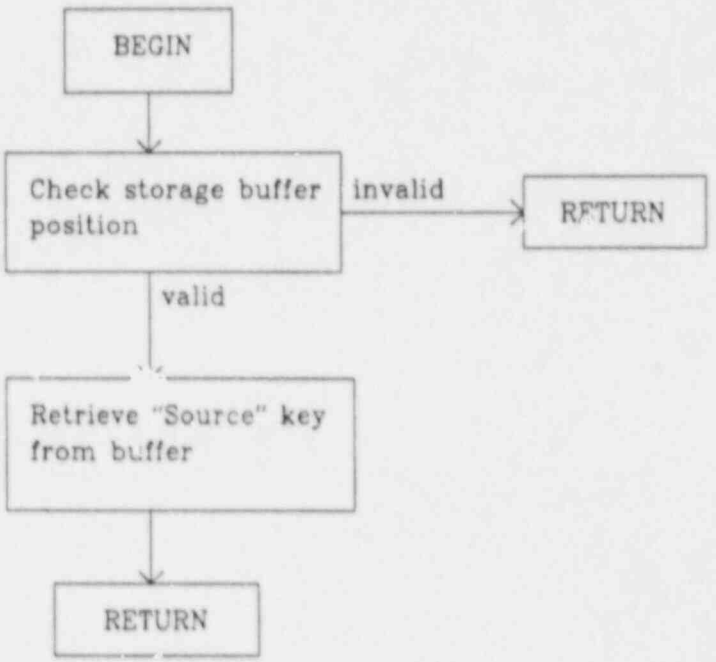

BEGIN
Clear storage buffer
FillChar
ADR
SIZE
RETURN


BEGIN
Copy contents of the storage buffer to a temporary buffer
RETURN


BEGIN
Get "Source" key
GetFieldA
Load "Source" key into
buffer maintaining
sorted order
BinarySearch
TSIZE
BinaryInsert
RETURN


BEGIN
Copy contents of a temporary buffer to the storage buffer
RETURN


BEGIN
Remove "Source" key
from storage buffer
BinaryDelete
TSIZE
RETURN


BEGIN
Check storage buffer position
invalid
RETURN
valid
Retrieve "Source" key from buffer
RETURN

APPENDIX C2

HCF DATA PROCESSING MODULE CHARTS

APPENDIX C2

HCF DATA PROCESSING MODULE CHARTS

This appendix contains high level flow charts for some modules used in HCF data processing. These charts provide an overview of what each module and each of its procedures is doing and how it is done; consequently, these charts can serve as road maps to the source code.

These charts are organized by module. For a particular module, the chart of the module is first, followed by charts of that module's procedures presented in alphabetical order. Three lists precede the charts for each module. An alphabetic, italicized list of procedures local to the module is given first. These local procedures are the procedures charted. The next list identifies procedures imported and names the library from which each procedure is imported. This list is alphabetized by procedure name with the procedure name italicized. If the libraries named here are not listed in Appendix A, they may be charted in Appendix C1. The final list is of messages referenced in the charts. Each message is assigned a number by which the messages are referenced and by which the list is ordered.

The elements in these charts are boxes and circles. Boxes contain descriptive comments on the processes being charted. Circled numbers are used to show chart continuation. Circled letters are used to show looping structures. Path lines, used to connect chart elements, have an arrow head at one end, thus showing the direction of activity flow from element to element.

Within boxes of the charts, procedure names are frequently used. These names are bolded and italicized. If a procedure name is considered descriptive enough, it may be used as a description of the process being shown; otherwise, the name will occur under a description and be indented showing that this procedure is used in the process described. If a local procedure is referenced, its name is followed by the dash character and then the word 'local'.

When multiple lines exit from a box, the condition determining the use of a path line will be indicated at the line. Any unlabeled path line exiting a box indicates the normal path to be taken.

When data base forms or relations are mentioned, they are shown within quotation marks.

MODULE RetrH .!

Local Procedures

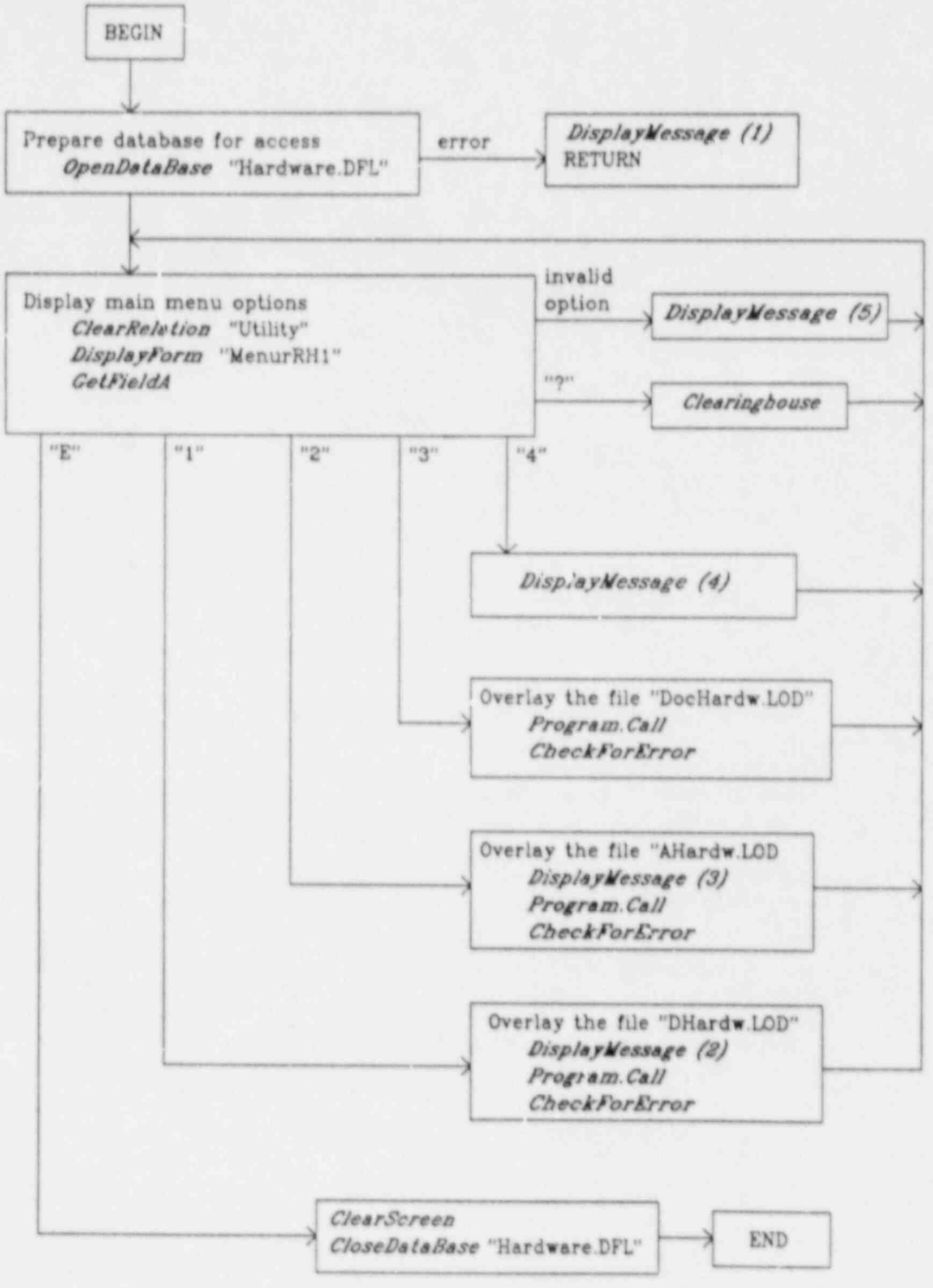
-none-

Procedures Imported

Call Program	*ClearScreen* ThorPort	*GetFieldA* Sage
CheckForError OlayHw	*CloseDataBase* Sage	*OpenDataBase* Sage
Clearinghouse ClrHouse	*DisplayForm* Sage	
ClearRelation Sage	*DisplayMessage* Sage	

Messages

(1) - "The schema file, 'Hardware.DFL', is not available"
(2) - "Loading Descriptive Search Programs ... please standby"
(3) - "Loading Ad Hoc Search Programs ... please standby"
(4) - "* Hardware Glossary is not yet complete *"
(5) - "Invalid Option"


BEGIN
Prepare database for access
OpenDataBase "Hardware.DFL"
error
DisplayMessage (1)
RETURN
Display main menu options
ClearRelation "Utility"
DisplayForm "MenurRH1"
GetFieldA
invalid option
DisplayMessage (5)
"?"
Clearinghouse
"E"
"1"
"2"
"3"
"4"
DisplayMessage (4)
Overlay the file "DocHardw.LOD"
Program.Call
CheckForError
Overlay the file "AHardw.LOD
DisplayMessage (3)
Program.Call
CheckForError
Overlay the file "DHardw.LOD"
DisplayMessage (2)
Program.Call
CheckForError
ClearScreen
CloseDataBase "Hardware.DFL"
END

MODULE DHardw

Local Procedures

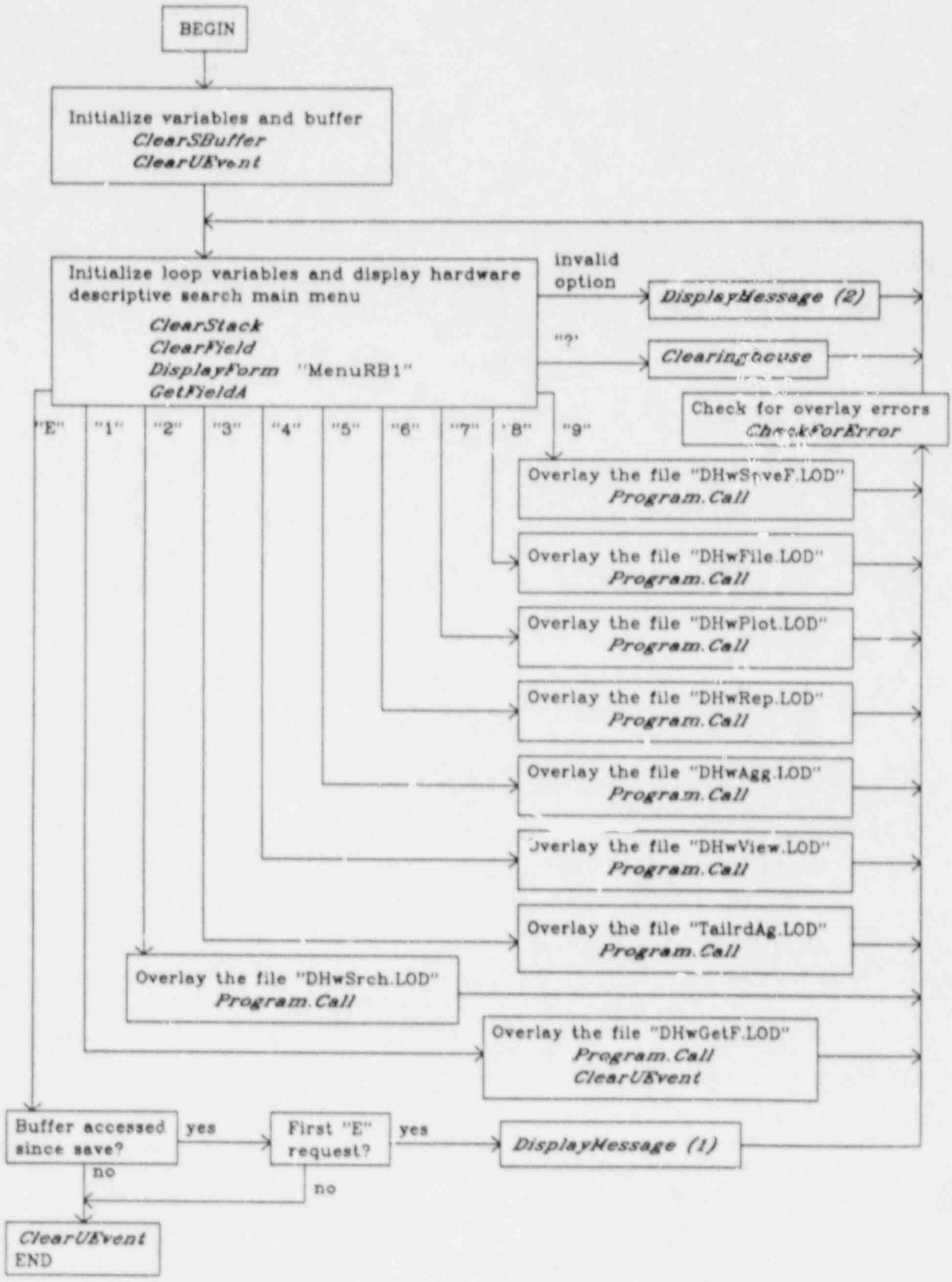
-none-

Procedures Imported

Call Program
CheckForError OlayHw
ClearField Sage
Clearinghouse ClrHouse
ClearSBuffer StoreMan
ClearStack General
ClearUEvent General
DisplayForm Sage
DisplayMessage Sage
GetFieldA Sage

Messages

(1) - "Located records not saved, select 'E' again for Exit WITHOUT Save!"
(2) - "Selected option is not provided"


BEGIN
Initialize variables and buffer
ClearSBuffer
ClearUEvent
Initialize loop variables and display hardware descriptive search main menu
ClearStack
ClearField
DisplayForm "MenuRB1"
GetFieldA
invalid option
DisplayMessage (2)
"?"
ClearingHouse
Check for overlay errors
CheckForError
"E"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"
Overlay the file "DHwSaveF.LOD"
Program.Call
Overlay the file "DHwFile.LOD"
Program.Call
Overlay the file "DHwPlot.LOD"
Program.Call
Overlay the file "DHwRep.LOD"
Program.Call
Overlay the file "DHwAgg.LOD"
Program.Call
Overlay the file "DHwView.LOD"
Program.Call
Overlay the file "TailrdAg.LOD"
Program.Call
Overlay the file "DHwSrch.LOD"
Program.Call
Overlay the file "DHwGetF.LOD"
Program.Call
ClearUEvent
Buffer accessed since save?
yes
First "E" request?
yes
DisplayMessage (1)
no
no
ClearUEvent
END

MODULE DHwAgg

Local Procedures

-none-

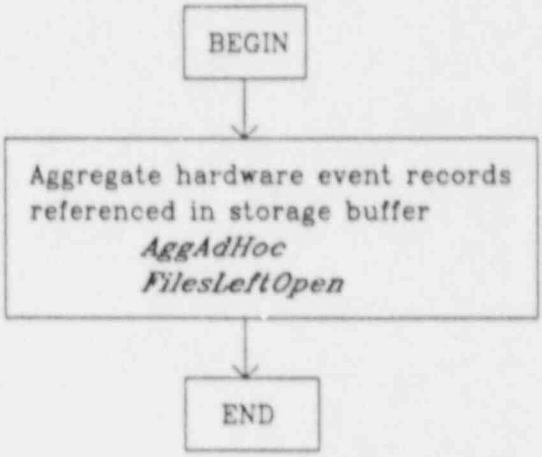
Procedures Imported

AggAdHoc HardAg

FilesLeftOpen OlayHw

Messages

-none-


BEGIN
Aggregate hardware event records referenced in storage buffer
AggAdHoc
FilesLeftOpen
END

MODULE AHardw

Local Procedures

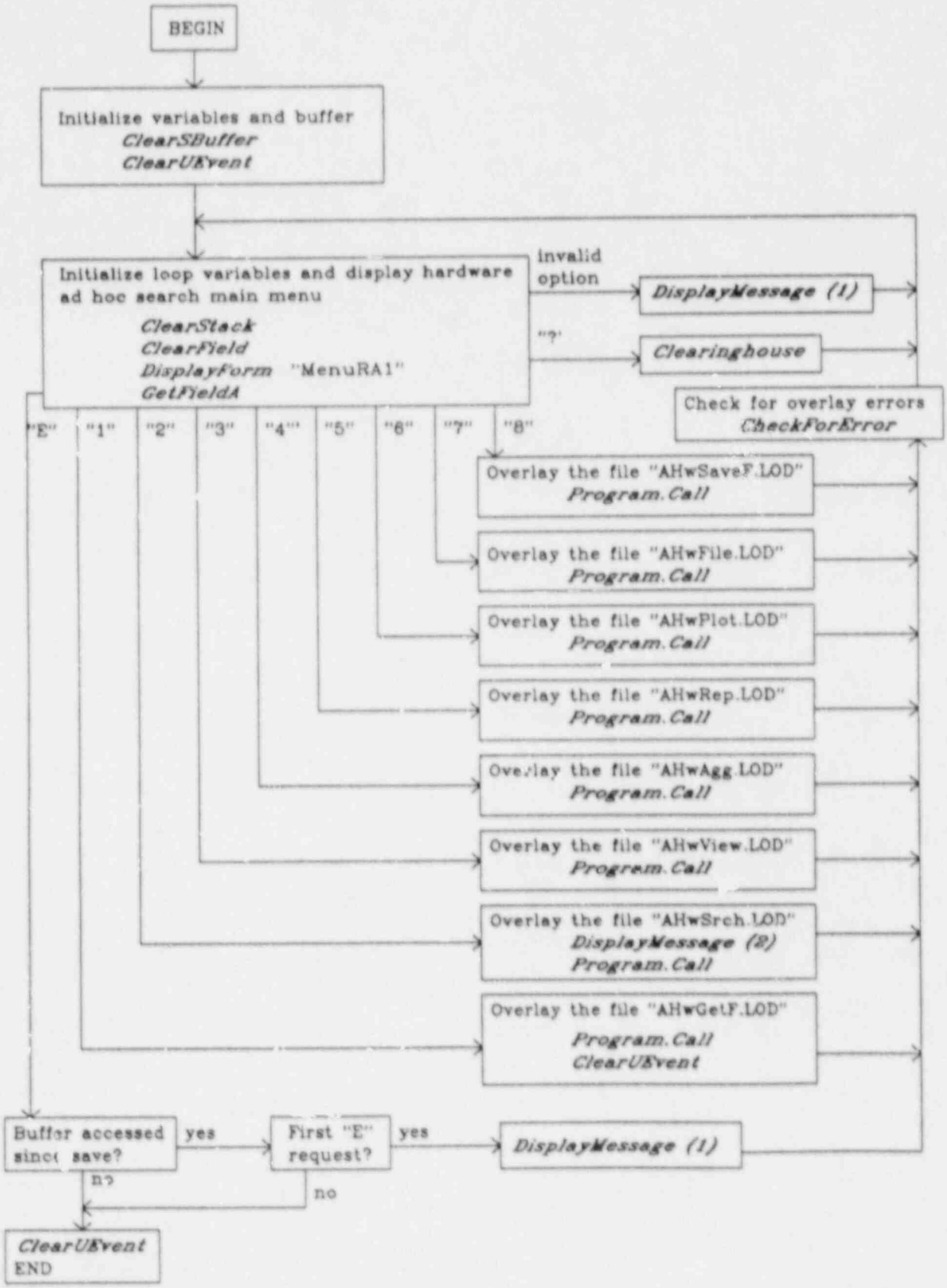
-none-

Procedures Imported

Call Program
CheckForError OlayHw
ClearField Sage
Clearinghouse ClrHouse
ClearSBuffer StoreMan
ClearStack General
ClearUEvent General
DisplayForm Sage
DisplayMessage Sage
GetFieldA Sage

Messages

(1) - "Located records not saved, select 'E' again for Exit WITHCUT Save!"
(2) - "Preparing ad hoc search screen ... standby"
(3) - "Selected option is not provided"


BEGIN
Initialize variables and buffer
ClearSBuffer
ClearUEvent
Initialize loop variables and display hardware
ad hoc search main menu
ClearStack
ClearField
DisplayForm "MenuRA1"
GetFieldA
invalid option
DisplayMessage (1)
"?"
Clearinghouse
Check for overlay errors
CheckForError
"E"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
Overlay the file "AHwSaveF.LOD"
Program.Call
Overlay the file "AHwFile.LOD"
Program.Call
Overlay the file "AHwPlot.LOD"
Program.Call
Overlay the file "AHwRep.LOD"
Program.Call
Overlay the file "AHwAgg.LOD"
Program.Call
Overlay the file "AHwView.LOD"
Program.Call
Overlay the file "AHwSrch.LOD"
DisplayMessage (2)
Program.Call
Overlay the file "AHwGetF.LOD"
Program.Call
ClearUEvent
Buffer accessed since save?
yes
First "E" request?
yes
DisplayMessage (1)
no
no
ClearUEvent
END

MODULE AHwAgg

Local Procedures

-none-

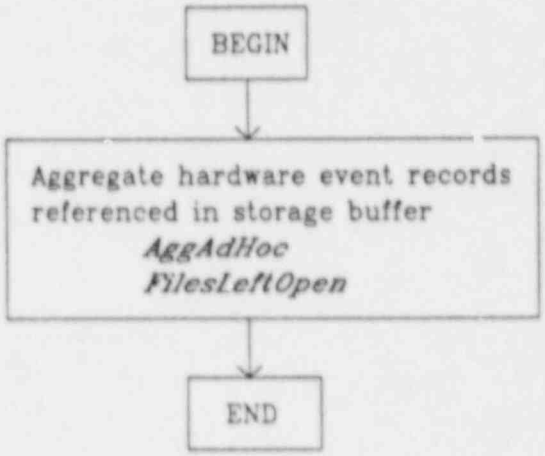
Procedures Imported

AggAdHoc HardAg

FilesLeftOpen OlayHw

Messages

-none-


BEGIN
Aggregate hardware event records referenced in storage buffer
AggAdHoc
FilesLeftOpen
END

APPENDIX D

RETRIEVAL ACCESS

APPENDIX D

RETRIEVAL ACCESS

This appendix presents the software elements involved with the set up of retrieval access. These elements are the DOS batch files used, module charts for each module, THOR reports for the data base named MASTER, and source code listings for the three programs used in providing retrieval access.

CONTENTS

Batch Files
Used for the Setup of Retrieval Access
in the NUCLARR System

There are seven DOS batch files used to set up retrieval access for users of the NUCLARR system. NUCLARR.BAT is used to initiate an access request. ZOPT.BAT is generated by the program MainMenu. NUCLARR.BAT executes MainMenu and then ZOPT.BAT. ZOPT.BAT always consists of a single DOS command which executes one of the other 5 batch files: Q.BAT, 1.BAT, 2.BAT, 3.BAT, or 4.BAT.

Q.BAT execution returns the user to the DOS root directory.

1.BAT execution provides access to data in the data base DATABANK and then executes NUCLARR.BAT again.

2.BAT execution provides access to data in the data base HARDWARE and then executes NUCLARR.BAT again.

3.BAT execution provides access to information on aggregation processing for HEP data and then executes NUCLARR.BAT again.

4.BAT execution provides access to information on aggregation processing for HCF data and then executes NUCLARR.BAT again.

A listing of file statements follows for each of these seven DOS batch files.

NUCLARR.BAT (in DOS root directory)

```
echo off
cls
cd\nuclarr
mainmenu
zopt
```

ZOPT.BAT (in DOS directory NUCLARR)

an example:

Q.BAT

Q.BAT (in DOS directory NUCLARR)

```
echo off
cls
echo  ****
echo  ****
echo  ****
echo  ****
echo  **** Reboot your PC to remove HALO Graphics Driver ****
echo  ****                                                  ****
echo  **** before proceeding to other PC applications.      ****
echo  ****
echo  ****
echo  ****
cd\
prompt $p$g
```

1.BAT (in DOS directory NUCLARR)

```
echo off
cls
\halo\thalo
IF ERRORLEVEL 1 GOTO NOLOD
\halo\halorlm
:NOLOD
cd\human
retrieve
cd\
nuclarr
```

2.BAT (in DOS directory NUCLARR)

```
echo off
cls
\halo\thalo
IF ERRORLEVEL 1 GOTO NOLOD
\halo\halorlm
:NOLOD
cd\hardware
retrhard
cd\
nuclarr
```

3.BAT (in DOS directory NUCLARR)

```
echo off
cls
cd\hep
hepnotes
cd\
nuclarr
```

4.BAT (in DOS directory NUCLARR)

```
echo off
cls
cd\hardware
hwnotes
cd\
nuclarr
```

Module Charts

This section contains high level flow charts for modules used to set up retrieval access. These charts provide an overview of what each module is doing and how it is done; consequently, these charts can serve as road maps to the source code.

For the first module, the chart of the module is followed by the chart of that module's only procedure. The other modules have no procedures. Three lists precede the charts for each module. An italicized list of procedures local to the module is given first. The next list identifies procedures imported and names the library from which each procedure is imported. This list is alphabetized by procedure name with the procedure name italicized. The libraries named here are listed in Appendix A or charted in Appendices B1 or C1. The final list is of messages referenced in the charts. Each message is assigned a number by which the messages are referenced and by which the list is ordered.

The boxes used in these charts contain descriptive comments on the processes being charted. Path lines, used to connect boxes, have an arrow head at one end, thus showing the direction of activity flow.

Within boxes of the charts, procedure names are frequently used. These names are bolded and italicized. If a procedure name is considered descriptive enough, it may be used as a description of the process being shown; otherwise, the name will occur under a description and be indented showing that this procedure is used in the process described. If a local procedure is referenced, its name is followed by the dash character and then the word 'local'.

When multiple lines exit from a box, the condition determining the use of a path line will be indicated at the line. Any unlabeled path line exiting a box indicates the normal path to be taken.

When data base forms or relations are mentioned, they are shown within quotation marks.

MODULE MainMenu

Local Procedures

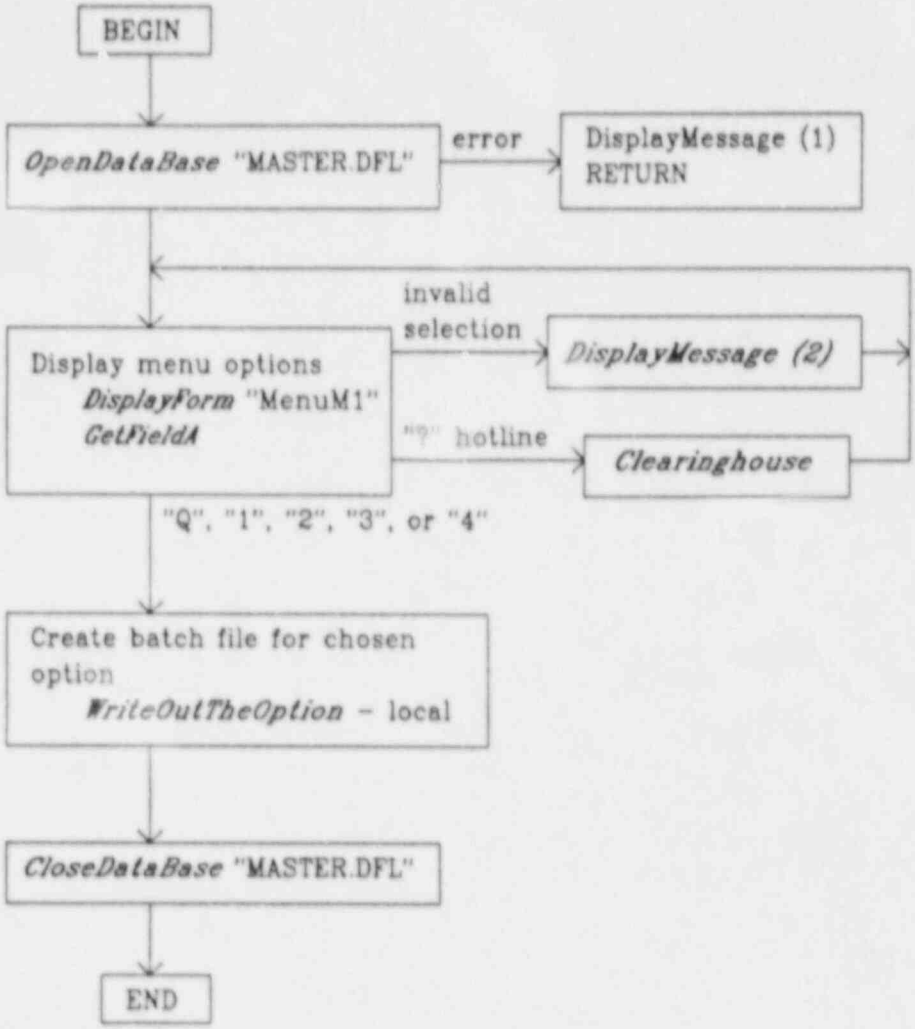
WriteOutTheOption

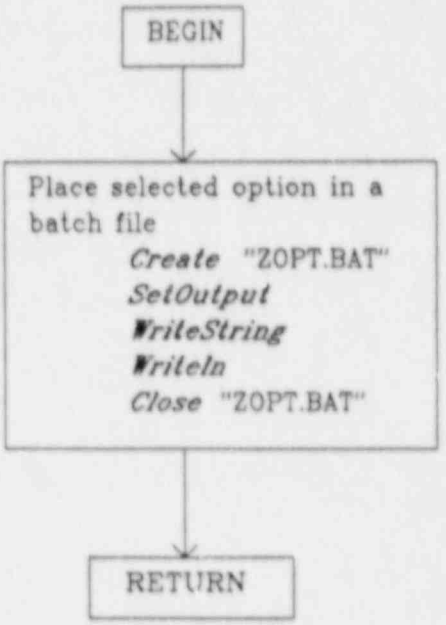
Procedures Imported

BinTextMode Files
Clearinghouse ClrHouse
Close Files
CloseDataBase Sage
Create Files
DisplayForm Sage
DisplayMessage Sage
GetFieldA Sage
OpenDataBase Sage
RecordEdit SageLib
ReplaceMode Files
Reset Files
SageOperations Sage
SetOutPut StandardIO
WriteLn SimpleIO
WriteString SimpleIO

Messages

(1) - "DataBase not opened"
(2) - "Invalid option selected"


BEGIN
OpenDataBase "MASTER.DFL"
error
DisplayMessage (1)
RETURN
invalid
selection
Display menu options
DisplayForm "MenuM1"
GetFieldA
DisplayMessage (2)
"?" hotline
Clearinghouse
"Q", "1", "2", "3", or "4"
Create batch file for chosen
option
WriteOutTheOption - local
CloseDataBase "MASTER.DFL"
END


BEGIN
Place selected option in a batch file
Create "ZOPT.BAT"
SetOutput
WriteString
WriteIn
Close "ZOPT.BAT"
RETURN

MODULE HEPNotes

Local Procedures

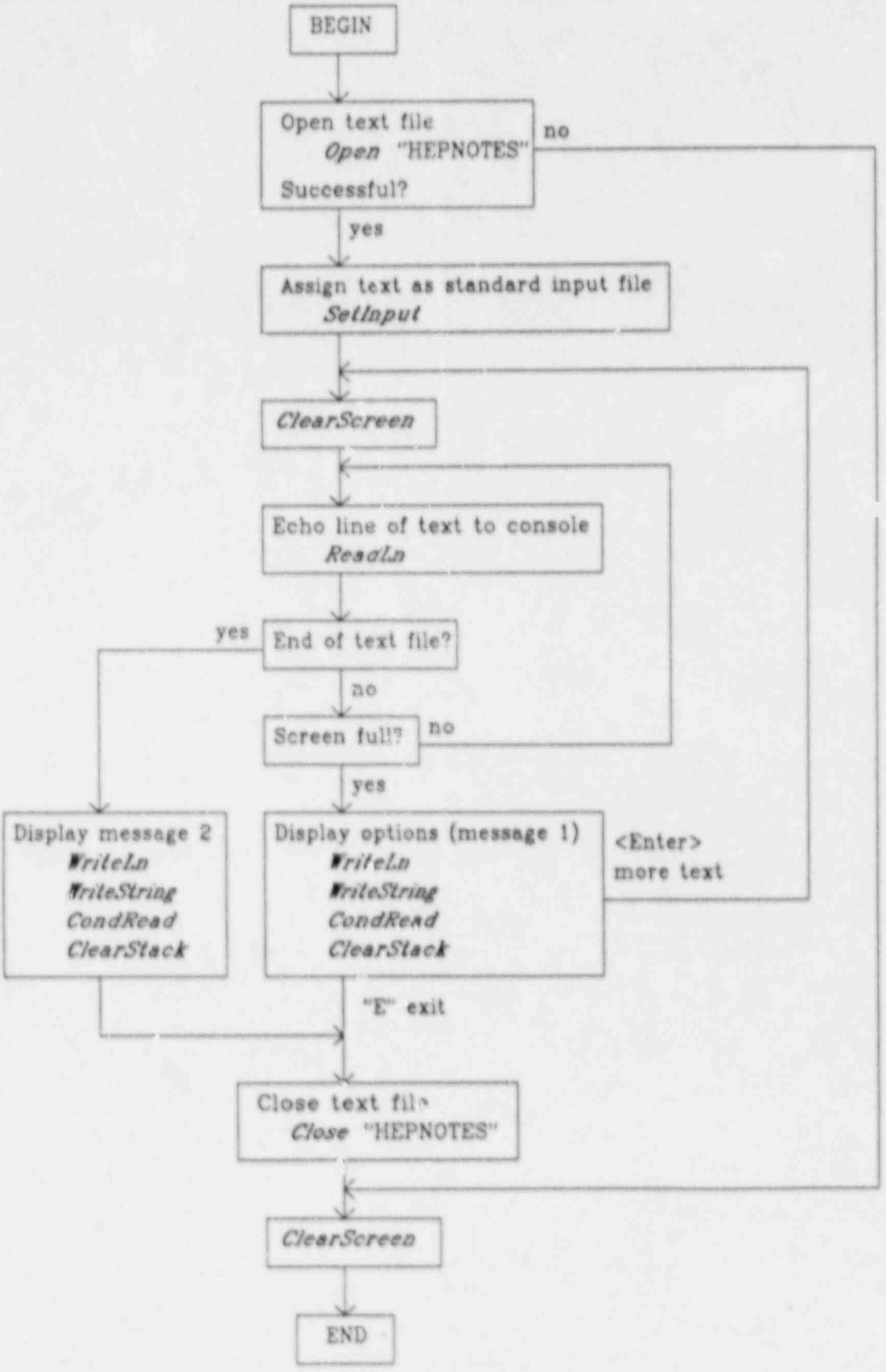
-none-

Procedures Imported

BinTextMode Files
ClearScreen ThorPort
ClearStack General
Close Files
CondRead Terminal
Open Sage
ReadChar SimpleIO
ReadLn SimpleIO
ReadWriteMode Files
Reset Files
SetInput StandardIO
WriteLn SimpleIO
WriteString SimpleIO

Messages

(1) - "Press <Enter> for More Text or <E> to Exit to Main Menu"
(2) - "Press <Enter> to return to Main Menu"


BEGIN
Open text file
Open "HEPNOTES"
Successful?
no
yes
Assign text as standard input file
SetInput
ClearScreen
Echo line of text to console
ReadLn
End of text file?
yes
no
Screen full?
no
yes
Display message 2
WriteLn
WriteString
CondRead
ClearStack
Display options (message 1)
WriteLn
WriteString
CondRead
ClearStack
<Enter>
more text
"E" exit
Close text file
Close "HEPNOTES"
ClearScreen
END

MODULE HWNotes

Local Procedures

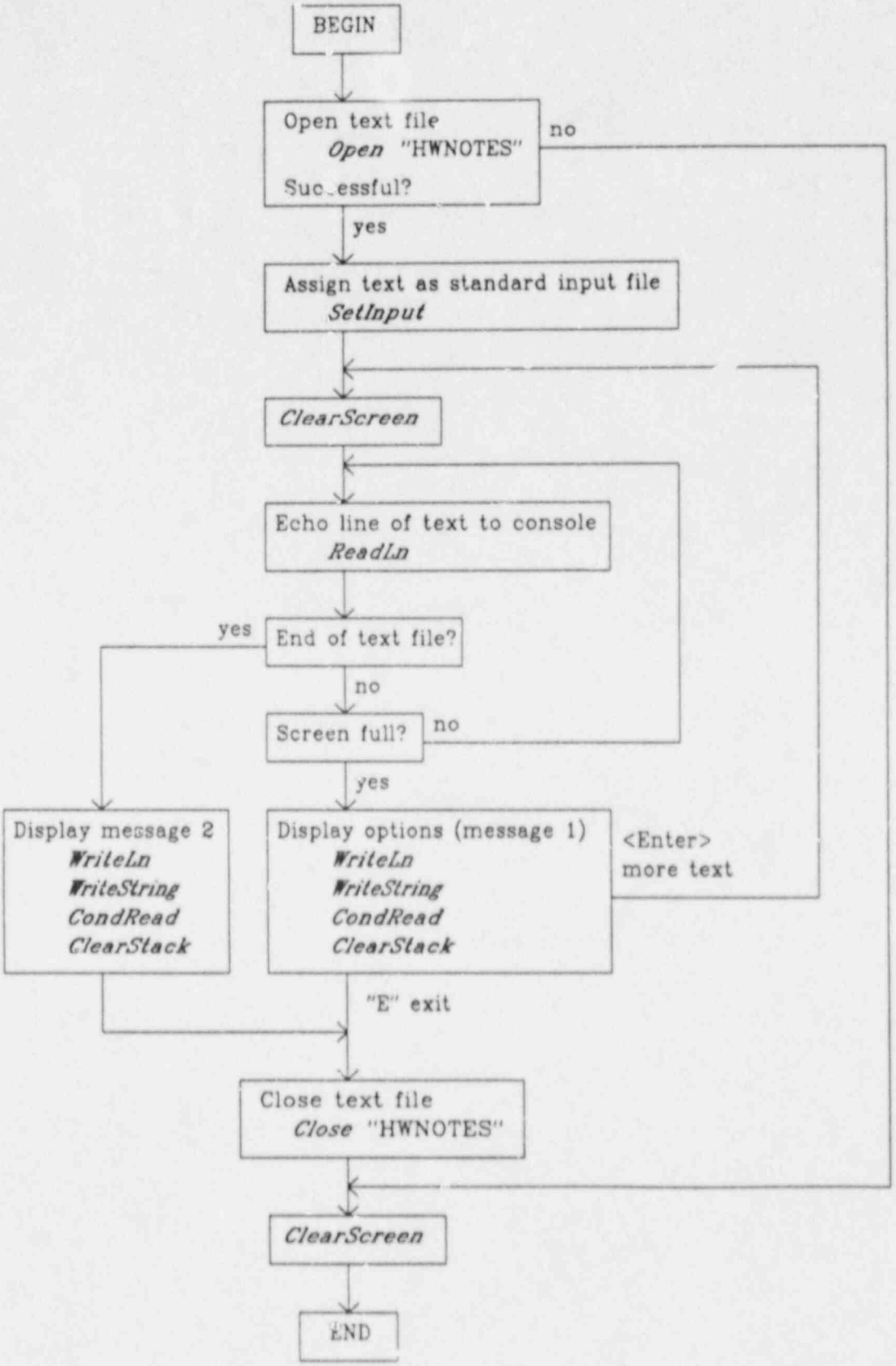
-none-

Procedures Imported

BinTextMode Files	*Open* Files	*SetInput* StandardIO
ClearScreen ThorPort	*ReadChar* SimpleIO	*WriteLn* SimpleIO
ClearStack General	*ReadLn* SimpleIO	*WriteString* SimpleIO
Close Files	*ReadWriteMode* Files	
CondRead Terminal	*Reset* Files	

Messages

(1) - "Press <Enter> for More Text or <E> to Exit to Main Menu"
(2) - "Press <Enter> to return to Main Menu"


BEGIN
Open text file
Open "HWNOTES"
Suc_essful?
no
yes
Assign text as standard input file
SetInput
ClearScreen
Echo line of text to console
ReadLn
yes
End of text file?
no
Screen full?
no
yes
Display message 2
WriteLn
WriteString
CondRead
ClearStack
Display options (message 1)
WriteLn
WriteString
CondRead
ClearStack
<Enter>
more text
"E" exit
Close text file
Close "HWNOTES"
ClearScreen
END

THOR Reports

This section contains THOR reports for the data base named MASTER. Two reports are included: a record report and a form report. There is no file summary report included because there are no files for MASTER to maintain.

A record report shows the field structure defined for records in those relations selected for the report. In other words, this is a report of some relation schemas. In the case of MASTER, there is only one relation and it has only one field.

A form report gives defining and appearance information for those forms selected for the report. In MASTER, there is only one form.

One or more THOR reports may be generated and sent to a DOS file for review or printing. For each such file written, a table of contents and index is included by THOR.

Table of Contents

RECORD REPORT FOR MASTER

--- 1 ---

Record Name : Utility
Description : Utility relation for main menu
Fields : 1
Byte Length : 6

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
UA1	1 UCA character	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			

FORM REPORT FOR MASTER

MenuM1 Main menu for NUCLARR 6 Field(s)
packed picture size -> 503 bytes
predominant color -> light cyan on black

NUclear Computerized Library for Assessing Reactor Reliability
Main Menu
Version 1.1

Q - Quit NUCLARR System
? - NUCLARR Hotline #

1 - Human Reliability Data
2 - Hardware Failure Data
3 - HEP Aggregation Help File
4 - Hardware Aggregation Help File

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
2	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
3	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
4	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
5	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
6	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	

Index

Source Code

This section contains source code listings for the three programs used in providing retrieval access. The DOCUPROC utility, which was used to generate these listings, provides line numbering, a table of contents, and an index. The line numbering is restarted for each module.

Table of Contents

```
MODULE MainMenu;

(**
    MainMenu - Module to display the main NUCLARR menu and capture choice

    G. H. Beers - April 1988

    Module displays main NUCLARR menu, reads the selected choice and puts it
    out to a file for future capture by HEP or Hardware execution control

    CALLING SEQUENCE -

      MainMenu

    ENTRY -

      data in

    EXIT  -

      data out

    FILES -


*)
FROM Sage      IMPORT OpenDataBase, CloseDataBase, GetFieldA, SageError,
                      SageOperations, DisplayForm, DisplayMessage;

FROM Files       IMPORT  BinTextMode, Close, EOF, File, FileState, Create,
                         ReplaceMode, Reset;

FROM SageLib     IMPORT  RecordEdit;

FROM SimpleIO    IMPORT  WriteString, WriteLn;

FROM StandardIO  IMPORT  SetOutput;

FROM ClrHouse    IMPORT  Clearinghouse;


VAR
  option :CHAR;


PROCEDURE WriteOutTheOption (ch :CHAR);

(**
    WriteOutTheOption - procedure which writes mainmenu option to a file;

    G. H. Beers - April 1988

    Procedure is passed the valid option and then puts it to a file.

    CALLING SEQUENCE -

      WriteOutTheOption (ch);

    ENTRY -

      ch :CHAR - option character with value 1,2,3,4.

    EXIT  -

    FILES -

      zoptfile

*)

VAR
  state  :FileState;
  btMode :BinTextMode;
  roMode :ReplaceMode;
  output :File;
```

```
 78: BEGIN
 79:   roMode := replace;
 80:   btMode := textMode;
 81:
 82:   Create (output, "ZOPT.BAT", btMode, roMode, state);
 83:   IF state = ok THEN
 84:     SetOutput (output);
 85:     WriteString (ch);
 86:     WriteString (".BAT");
 87:     WriteLn;
 88:     Close (output,state);
 89:   END;
 90:
 91: END WriteOutTheOption;
 92:
 93:
 94:
 95: BEGIN  (* MainMenu *)
 96:   OpenDataBase ("MASTER.DFL", 5, 5, 10000);
 97:   IF SageError # 0 THEN
 98:     DisplayMessage ("DataBase not opened", TRUE);
 99:     RETURN;
100:   END;
101:
102:   LOOP
103:     DisplayForm ("MenuM1", "Utility", "UA1", FALSE);
104:     GetFieldA ("Utility", "UA1", option);
105:
106:     CASE option OF
107:
108:           "0",
109:           "1",
110:           "2",
111:           "3",
112:           "4" : WriteOutTheOption (option) ;
113:                 EXIT|
114:
115:           "?" : Clearinghouse;
116:
117:     ELSE
118:       DisplayMessage ("Invalid option selected", TRUE);
119:     END;
120:   END;  (* LOOP *)
121:
122:   CloseDataBase;
123:
124: END MainMenu.
```

```
MODULE HEPNotes;

(**
    HEPNotes - Control program for the viewing of aggregation help file.

    G. H. Beers - April 1988

    Program reads and displays aggregation helps one page at a time.  The reader
    is able to quit at the end of any page and return to the main Nuclarr menu.
    Uses SimpleIO to read and echo each line.

    CALLING SEQUENCE -

      HEPNotes ;

    ENTRY -

    EXIT  -

    FILES -

      File HEPNotes

*)

FROM Files      IMPORT BinTextMode, Close, EOF, File, FileState, Open,
                       ReadWriteMode, Reset;

FROM NUCGEN     IMPORT ClearStack;

FROM SimpleIO   IMPORT ReadChar, ReadLn, WriteLn, WriteString, EOT, EOL;

FROM StandardIO IMPORT SetInput;

FROM Terminal   IMPORT CondRead;

FROM ThorPort   IMPORT ClearScreen;


VAR
  termCommand :CHAR;
  success     :BOOLEAN;
  endOfFile   :BOOLEAN;
  readyToExit :BOOLEAN;
  thisPage    :CARDINAL;

  state       :FileState;
  btMode      :BinTextMode;
  roMode      :ReadWriteMode;
  input       :File;


BEGIN
  roMode := readOnly;
  btMode := textMode;

  readyToExit := FALSE;

  Open (input, "HEPNOTES", btMode, roMode, state);
  IF state = ok THEN
    SetInput (input);

    WHILE NOT readyToExit DO
      WriteLn;
      ClearScreen;
      thisPage := 1;
      LOOP
        ReadLn ;
        IF EOT() THEN
          readyToExit := TRUE;
          LOOP
            WriteLn;
            IF thisPage < 23 THEN
              INC(thisPage);
```

```
75:             ELSE
76:               EXIT;
77:             END;
78:           END;
79:           EXIT;
80:         END;
81:         IF thisPage < 23 THEN
82:           INC(thisPage);
83:         ELSE
84:           EXIT;
85:         END;
86:       END;  (* LOOP *)
87:       WriteLn;
88:
89:       IF readyToExit THEN
90:         WriteString ("Press <Enter> to return to Main Menu");
91:       ELSE
92:         WriteString ("Press <Enter> for More Text or <E> to Exit to Main Menu");
93:       END;
94:
95:       REPEAT
96:        CondRead (termCommand, success);
97:         IF success THEN
98:           IF NOT readyToExit THEN
99:             IF (termCommand = "E") OR (termCommand = "e") THEN
100:              readyToExit := TRUE;
101:            ELSE
102:            END;
103:          ELSE
104:          END;
105:        END;
106:      UNTIL success;
107:
108:    ClearStack;
109:
110:    END;  (* WHILE *)
111:
112:    Close (input, state);
113:
114:  END;
115:
116:  ClearScreen;
117:
118: END HEPNotes.
```

```
1: MODULE HWNotes;
2:
3: (**
4:     HWNotes - Control program for the viewing of aggregation help file.
5:
6:     G. ... Beers - April 1988
7:
8:     Program reads and displays aggregation helps one page at a time.  The reader
9:     is able to quit at the end of any page and return to the main Nuclarr menu.
10:    Uses SimpleIO to read and echo each line.
11:
12:    CALLING SEQUENCE -
13:
14:      HWNotes ;
15:
16:    ENTRY -
17:
18:    EXIT  -
19:
20:    FILES -
21:
22:      File HWNotes
23:
24: *)
25:
26: FROM Files      IMPORT BinTextMode, Close, EOF, File, FileState, Open,
27:                       ReadWriteMode, Reset;
28:
29: FROM General    IMPORT ClearStack;
30:
31: FROM SimpleIO   IMPORT ReadChar, ReadLn, WriteLn, WriteString, EOT, EOL;
32:
33: FROM StandardIO IMPORT SetInput;
34:
35: FROM Terminal   IMPORT CondRead;
36:
37: FROM ThorPort    MPORT ClearScreen;
38:
39:
40: VAR
41:   termCommand :CHAR;
42:   success     :BOOLEAN;
43:   endOfFile   :BOOLEAN;
44:   readyToExit :BOOLEAN;
45:   thisPage    :CARDINAL;
46:
47:   state       :FileState;
48:   btMode      :BinTextMode;
49:   roMode      :ReadWriteMode;
50:   input       :File;
51:
52:
53: BEGIN
54:   roMode := readOnly;
55:   btMode := textMode;
56:
57:   readyToExit := FALSE;
58:
59:   Open (input, "HWNOTES", btMode, roMode, state);
60:   IF state = ok THEN
61:     SetInput (input);
62:
63:     WHILE NOT readyToExit DO
64:       WriteLn;
65:       ClearScreen;
66:       thisPage := 1;
67:       LOOP
68:         ReadLn ;
69:         IF EOT() THEN
70:           readyToExit := TRUE;
71:           LOOP
72:             WriteLn;
73:             IF thisPage < 23 THEN
74:               INC(thisPage);
```

```
 75:             ELSE
 76:               EXIT;
 77:             END;
 78:           END;
 79:           EXIT;
 80:         END;
 81:         IF thisPage < 23 THEN
 82:           INC(thisPage);
 83:         ELSE
 84:           EXIT;
 85:         END;
 86:       END;  (* LOOP *)
 87:       WriteLn;
 88:
 89:       IF readyToExit THEN
 90:         WriteString ("Press <Enter> to return to Main Menu");
 91:       ELSE
 92:         WriteString ("Press <Enter> for More Text or <E> to Exit to Main Menu");
 93:       END;
 94:
 95:       REPEAT
 96:        CondRead (termCommand, success);
 97:         IF success THEN
 98:           IF NOT readyToExit THEN
 99:             IF (termCommand = "E") OR (termCommand = "e") THEN
100:               readyToExit := TRUE;
101:             ELSE
102:             END;
103:           ELSE
104:           END;
105:         END;
106:       UNTIL success;
107:
108:     ClearStack;
109:
110:     END;  (* WHILE *)
111:
112:     Close (input, state);
113:
114:   END;
115:
116:   ClearScreen;
117:
118: END HWNotes.
```

APPENDIX E

SAMPLE THOR REPORTS FOR DATABANK DATA BASE

APPENDIX E

SAMPLE THOR REPORTS FOR DATABANK DATA BASE

This appendix presents a sample of some THOR reports for the data base named DATABANK. Three report types are included: a file summary report, a record report, and a form report.

A file summary report identifies all files maintained by a particular data base.

A record report shows the field structure defined for records in those relations selected for the report. In other words, this is a report of some relation schemas.

A form report gives defining and appearance information for those forms selected for the report.

It should be noted that names of files, records (relations), fields, and forms are all arbitrarily determined and not necessarily meaningful. However, there is also a description associated with each name, except for files. Although it is intended that the names selected be meaningful, their meaning may be obscure because of the need for so many names and because of their length being limited to eight characters.

One or more THOR reports may be generated and sent to a DOS file for review or printing. For each such file written, a table of contents and index is included by THOR.

Table of Contents

FILE SUMMARY FOR DATABANK

	Data File Name	Index File Name	Block Name
1	CellData.DAT	CellData.IDX	
2	CELLVAL.DAT	CELLVAL.IDX	
3	Columns.DAT	Columns.IDX	
4	HEP.DAT	HEP.IDX	
5	MATRCOL.DAT	MATRCOL.IDX	
6	MATRROW.DAT	MATRROW.IDX	
7	REFERENC.DAT	REFERENC.IDX	
8	ROWS.DAT	ROWS.IDX	ROWS.BLK
9	TaskStmt.DAT	TaskStmt.IDX	
10	MATRDESC.DAT	MATRDESC.IDX	
11	PLANT.DAT	PLANT.IDX	
12	DOCUMENT.DAT	DOCUMENT.IDX	
13	ERROR.DAT	ERROR.IDX	
14	LER.DAT	LER.IDX	

Total Number of Files = 14

--- 1 ---

Record Name : CellData
Description : Source Statement Information
Fields : 49
Byte Length : 125
File Number : 1
Data File : CellData.DAT
Index File : CellData.IDX

Field Name / Repeat	Description / Type	Length / Decimals	Key Type / K/Nod N/Cr	Sub-Fields
AggFlag	Aggregation flag (-,T,C,F)	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
CellColu	Column ID	2	Non-keyed	
1	3 (Integer)			
CellKey	Primary Key	11	Primary	CellMatr
1	0 (Concatenated)		20 5	CellRow
				CellColu
				CellTask
				CellNumb
CellMatr	Matrix Number	2	Non-keyed	
1	3 (Integer)			
CellNumb	Source Statement Number	2	Non-keyed	
1	3 (Integer)			
CellRow	Row ID	3	Non-keyed	
1	3 (Integer)			
CellTask	Task Statement Number	2	Non-keyed	
1	3 (Integer)			
DateAdd	Date Added to Data Bank	10	Alternate	
1	7 (Date) (yyyy/mm/dd)		20 5	
DocID	Document Number xxx of (xxx-yy	3	Non-keyed	
1	3 (Integer)			
DocItem	Page, Paragraph, Figure, Line	20	Non-keyed	
1	1 (Alphanumeric)			
DocKey	Alternate Key for DocYear + ID	5	Alternate	DocYear
1	0 (Concatenated)		20 5	DocID

RECORD REPORT FOR DATABANK

Field Name / Repeat	Description / Type	Length / Decimals	Key Type K/Nod N/Cr	Sub-Fields
DocYear	Document Year yy of (xxx-yy)	2	Non-keyed	
1	3 (Integer)			
E	Number of Errors	8	Non-keyed	
1	4 (Real)	1		
EI	Number of Errors - calculated	8	Non-keyed	
1	4 (Real)	1		
Erf	Error Factor for Median HEP	3	Non-keyed	
1	3 (Integer)			
ErfI	Err Factr for Median HEP Input	3	Non-keyed	
1	3 (Integer)			
ErrFail	1=rc/o 2=rc/c 3=rnc/o 4=rnc/c	1	Non-keyed	
1	3 (Integer)			
HEP	Human Error Probability	9	Non-keyed	
1	4 (Real)	7		
HEPI	Human Error Probability Input	9	Non-keyed	
1	4 (Real)	7		
LCB	Lower Confidence Bound	9	Non-keyed	
1	4 (Real)	7		
LCBI	Lower Confidence Bound Input	9	Non-keyed	
1	4 (Real)	7		
Median	Median HEP	9	Non-keyed	
1	4 (Real)	7		
MedianI	Median HEP Input	9	Non-keyed	
1	4 (Real)	7		
N	No. Opportunities for Error	8	Non-keyed	
1	4 (Real)	1		
NI	No.Opportunities for Error Inp	8	Non-keyed	
1	4 (Real)	1		
PSFExper	Experience Perform Shaping Fac	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFFb	Feedback PSF	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			

RECORD REPORT FOR DATABANK

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
PSFPT	Performance Time PSF	8	Non-keyed	
1	14 (Short Time)(w/o microsec.)			
PSFProc	Procedure Perform Shaping Fact	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFStf	Staffing PSF	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFStres	Stress Perform Shaping Factor	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFSup	Supervision PSF	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFTA	Time Availalbe PSF	8	Non-keyed	
1	14 (Short Time)(w/o microsec.)			
PSFTag	Tagging Perform. Shaping Factr	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PSFTrn	Training PSF	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
PlantCod	Plant Code	4	Non-keyed	
1	2 (Upper Case Alphanumeric)			
RefID	Cited Document ID xxx of xxxyy	3	Non-keyed	
1	3 (Integer)			
RefKey	Alt. Key for Reference Doc Err	5	Alternate	RefYear
1	0 (Concatenated)		20 5	RefID
RefPage	Cited Reference Document Page	6	Non-keyed	
1	1 (Alphanumeric)			
RefYear	Cited Document Year yy (xxx-yy	2	Non-keyed	
1	3 (Integer)			
Source	Data Origin (lab, field, etc.)	2	Non-keyed	
1	3 (Integer)			
UCB	Upper Confidence Bound	9	Non-keyed	
1	4 (Real)	7		
UCBI	Upper Confidence Bound Input	9	Non-keyed	
1	4 (Real)	7		

RECORD REPORT FOR DATABANK

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
UKeyMC	Alt. matrix + col	4	Alternate	CellMatr
1	0 (Concatenated)		20 5	CellColu
UKeyMR	Alt. matrix + row	5	Alternate	CellMatr
1	0 (Concatenated)		20 5	CellRow
UKeyMRC	Alt. matrix+ row + col	7	Alternate	CellMatr
1	0 (Concatenated)		20 5	CellRow
				CellColu
UKeyMRCT	Alt. matrix+ row + col + task	9	Alternate	CellMatr
1	0 (Concatenated)		20 5	CellRow
				CellColu
				CellTask
UKyMRCTE	Alt. Key for Cell + Task + Err	10	Alternate	CellMatr
1	0 (Concatenated)		20 5	CellRow
				CellColu
				CellTask
				ErrFail
cellUsed	Source used in calculations	1	Non-keyed	
1	1 (Alphanumeric)			

Field Validation Report

--- 1 ---

Record Name : CellData
Description : Source Statement Information

1) AggFlag Aggregation flag (-,T,C,F) (Upper Case Alphanumeric)

Enumerated List - 4 value(s)

- C F T

2) CellColu Column ID (Integer)

Minimum Value -> 0
Maximum Value ->99

4) CellMatr Matrix Number (Integer)

Minimum Value -> 0
Maximum Value ->99

6) CellRow Row ID (Integer)

Minimum Value -> 0
Maximum Value ->999

7) CellTask Task Statement Number (Integer)

Minimum Value -> 0

13) E Number of Errors (Real)

Minimum Value -> .0

14) EI Number of Errors - calculated (Real)

Minimum Value -> .0

17) ErrFail 1=rc/o 2=rc/c 3=rnc/o 4=rnc/c (Integer)

Minimum Value ->1
Maximum Value ->4

22) Median Median HEP (Real)

Minimum Value -> .0000000

23) MedianI Median HEP Input (Real)

Minimum Value -> .0000000

24) N No. Opportunities for Error (Real)

Minimum Value -> .0

25) NI No.Opportunities for Error Inp (Real)

Minimum Value -> .0

41) Source Data Origin (lab, field, etc.) (Integer)

Minimum Value -> 0

--- 2 ---

Record Name : CellVal
Description : Cell Validity
Fields : 5
Byte Length : 7
File Number : 2
Data File : CELLVAL.DAT
Index File : CELLVAL.IDX

Field Name / Repeat	Description / Type	Length / Decimals	Key Type K/Nod N/Cr	Sub-Fields
CellColu 1	Column ID 3 (Integer)	2	Non-keyed	
CellMatr 1	Matrix 3 (Integer)	2	Non-keyed	
CellRow 1	Row ID 3 (Integer)	3	Non-keyed	
PrimeKey 1	Primary Key = Cell location 0 (Concatenated)	7	Primary 20 5	CellMatr CellRow CellColu
Validity 1	0=invalid, 1=valid, 2=shaded 3 (Integer)	1	Non-keyed	

Field Validation Report

--- 2 ---

Record Name : CellVal
Description : Cell Validity

1) CellColu Column ID (Integer)

Minimum Value -> 0
Maximum Value ->99

2) CellMatr Matrix (Integer)

Minimum Value -> 0
Maximum Value ->99

3) CellRow Row ID (Integer)

Minimum Value -> 0
Maximum Value ->999

--- 3 ---

Record Name : Columns
Description : General Column Labels
Fields : 3
Byte Length : 231
File Number : 3
Data File : Columns.DAT
Index File : Columns.IDX

Field Name / Repeat	Description / Type	Length / Decimals	Key Type K/Nod N/Cr	Sub-Fields
ColDef	Definition for column verb	72	Non-keyed	
3	1 (Alphanumeric)			
ColDescr	Label for Column ID	16	Non-keyed	
1	2 (Upper Case Alphanumeric)			
ColNumbr	Column Number for Descriptions	2	Primary	
1	3 (Integer)		20 5	

Field Validation Report

--- 3 ---

Record Name : Columns
Description : General Column Labels

3) ColNumbr Column Number for Descriptions (Integer)

Minimum Value -> 0
Maximum Value ->99

FORM REPORT FOR DATABANK

Menur01 Retrieve - Main Menu 11 Field(s) Help Form -> zr01
packed picture size -> 957 bytes
predominant color -> light cyan on black

Retrieval of
NUclear Computerized Library for Assessing Reactor Reliability
Human Error Probability Data

Main Retrieve Menu
Version 1.1

E - Exit from HEP Retrieve Program
? - NUCLARR Hotline #

1 - Locate HEP Data Records by Description
2 - Review Documents
3 - Locate HEP Data Records by Ad Hoc Search
4 - Report on Located HEP Data Records
5 - Plot from Located HEP Data Records
6 - Calculate Aggregated HEP for Located Records
7 - Generate ASCII File for dBase III/SAS/SPSS
8 - Retrieve a Saved Data Records File
9 - Save Located Records to a File

Field	Display Type	Record & (Field) - rpt		Field Type	Help Frm
1	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
2	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
3	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
4	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
5	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
6	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
7	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
8	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
9	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
10	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	
11	Highlighted	UCommand(CommandA) -	1	Upper Case Alphanumeric	

zr001 Retrieve - MenurOi, continued 0 Field(s) Help Form -> zr002
packed picture size -> 1124 bytes
predominant color -> light yellow on blue

Main Menu

4 Report on Located Data - A report is output to a device to show the cell information, task statement, and source statement for all located data records currently referenced in the storage buffer.

5 Plot from Located Data - Generate log plots from aggregated HEPs for located data records (option 6 needs to be used before option 5). View plots on the console and print them on an Epson or LaserJet printer.

6 Calculate Aggregated HEP - Combine HEPs for Source Statements from the same Task Statement and then combine such Task Statement HEPs to determine an aggregated HEP value with confidence bounds for all located data records currently referenced in the storage buffer.

7 Generate ASCII File - Generate an ASCII file in a format compatible with the dBase III, SAS, and SPSS systems. Source Statements are converted for all located data records or for all data in the data base. For the dBase III system, Task Statements and Documents may also be converted.

Esc - View more help (options 8 & 9)
Enter - View previous help (options E, ?, 1 - 3)

---

zr002 Retrieve - zr001, continued 0 Field(s) Help Form -> EXIT
packed picture size -> 695 bytes
predominant color -> light yellow on blue

Main Menu

8 Retrieve a Saved Records File - Moves references to a set of previously located records from a file into the active storage buffer. Such a file must have been created by option 9 of this Main Menu. Any data in the buffer when this Retrieve option is selected will be replaced by that from the selected file.

9 Save Located Records to a File - Copies references (to located records), currently in the storage buffer, out to a disk file named by the user. If the named file currently exists, it will be overwritten with the data passed to it through this option.

Esc - Return to Main Menu
Enter - View previous help (options 4 - 7)

```
zr01      Retrieve - Main Menu Menur01      0 Field(s)  Help Form -> zr001
          packed picture size ->   864 bytes
          predominant color   -> light yellow on blue
```

Main Menu

E Exit - Exit the NUCLARR Retrieval system. All located data record references are cleared from the storage buffer.

? NUCLARR Hotline # - Display the NUCLARR Hotline telephone number for information concerning the data or the utilization of NUCLARR.

1 Descriptive Search - Locate NUCLARR data records by using a series of user-friendly menus. The storage buffer holds references to located records.

2 Review Documents - Review of the source documents referenced in source statements as original and reference documents. A report option and a search of all references to a document are available.

3 Ad Hoc Search - Locate NUCLARR data records by ad hoc search method. The storage buffer holds references to located records.

Esc - View more help (options 4 - 9)
Enter - Return to Main Menu

Index

APPENDIX F

SAMPLE THOR REPORTS FOR HARDWARE DATA BASE

APPENDIX F

SAMPLE THOR REPORTS FOR HARDWARE DATA BASE

This appendix presents a sample of some THOR reports for the data base named HARDWARE. Three report types are included: a file summary report, a record report, and a form report.

A file summary report identifies all files maintained by a particular data base.

A record report shows the field structure defined for records in those relations selected for the report. In other words, this is a report of some relation schemas.

A form report gives defining and appearance information for those forms selected for the report.

It should be noted that names of files, records (relations), fields, and forms are all arbitrarily determined and not necessarily meaningful. However, there is also a description associated with each name, except for files. Although it is intended that the names selected be meaningful, their meaning may be obscure because of the need for so many names and because of their length being limited to eight characters.

One or more THOR reports may be generated and sent to a DOS file for review or printing. For each such file written, a table of contents and index is included by THOR.

Table of Contents

FILE SUMMARY FOR HARDWARE

	Data File Name	Index File Name	Block Name
1	HCOMP.DAT	HCOMP.IDX	
2	FAILMODE.DAT	FAILMODE.IDX	
3	HARDAPPL.DAT	HARDAPPL.IDX	
4	HEVENT.DAT	HEVENT.IDX	
5	HSOURCE.DAT	HSOURCE.IDX	HSOURCE.BLK
6	HSYSTEM.DAT	HSYSTEM.IDX	
7	CMPVALID.DAT	CMPVALID.IDX	
8	DEGREE.DAT	DEGREE.IDX	
9	DISTTYPE.DAT	DISTTYPE.IDX	
10	EXPRTYPE.DAT	EXPRTYPE.IDX	
11	EXPOSROR.DAT	EXPOSROR.IDX	
12	FAILORIG.DAT	FAILORIG.IDX	
13	FAILRTYP.DAT	FAILRTYP.IDX	
14	FAILSEVR.DAT	FAILSEVR.IDX	
15	NORMSTAT.DAT	NORMSTAT.IDX	
16	PLANT.DAT	PLANT.IDX	
17	DOCUMENT.DAT	DOCUMENT.IDX	
18	AGSTAT.DAT	AGSTAT.IDX	
19	UEVENT.DAT	UEVENT.IDX	
20	REFERENC.DAT	REFERENC.IDX	
21	ARCHENGR.DAT	ARCHENGR.IDX	
22	RECCOUNT.DAT	RECCOUNT.IDX	

Total Number of Files = 22

--- 20 ---

Record Name : Source
Description : Source probability data
Fields : 75
Byte Length : 343
File Number : 5
Data File : HSOURCE.DAT
Index File : HSOURCE.IDX
Block File : HSOURCE.BLK

Field Name / Repeat	Description / Type	Length / Decimals	Key Type K/Nod N/Cr	Sub-Fields
AggFlag 1	Aggregation allowed? 11 (Boolean) (T/F)	1	Non-keyed	
AggType 1	Aggregation method type 2 (Upper Case Alphanumeric)	1	Non-keyed	
Appl 3	Application 2 (Upper Case Alphanumeric)	6	Non-keyed	
BayUpd 1	Bayesian Update? 2 (Upper Case Alphanumeric)	1	Non-keyed	
CatComp 1	Category/Component 0 (Concatenated)	4	Alternate 20 5	Category Comp
Category 1	Gen. component cat. (level 1) 2 (Upper Case Alphanumeric)	1	Non-keyed	
Comment 1	Additional source comments 12 (Variable Block Data)	4	Non-keyed	
Comp 1	Gen. component name (level 2) 2 (Upper Case Alphanumeric)	3	Non-keyed	
CompDmnd 1	Average indiv. comp. demands 3 (Integer)	8	Non-keyed	
CompF 1	Component//failure 0 (Concatenated)	6	Non-keyed	Comp FailID
CompFID 1	Component/design/failure 0 (Concatenated)	8	Non-keyed	Comp Design FailID

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
CompHrs 1	Component exposure time 3 (Integer)	12	Non-keyed	
CompID 1	Component identifier 0 (Concatenated)	5	Non-keyed	Comp Design
DataEnd 1	Latest year of failure data 2 (Upper Case Alphanumeric)	2	Non-keyed	
DataStrt 1	Earliest year of failure data 2 (Upper Case Alphanumeric)	2	Non-keyed	
Degree 1	Deg-degraded or incipient fail 2 (Upper Case Alphanumeric)	2	Non-keyed	
Demand 1	Demand source data? 11 (Boolean) (Y/N)	1	Non-keyed	
Design 1	Component design (level 3) 2 (Upper Case Alphanumeric)	2	Non-keyed	
Distrb 1	Distribution type 2 (Upper Case Alphanumeric)	8	Non-keyed	
DocPage 1	Document Reference page, etc. 2 (Upper Case Alphanumeric)	20	Non-keyed	
DocRef 1	Document Reference number 3 (Integer)	5	Non-keyed	
Domestic 1	Domestic Failure Data? 2 (Upper Case Alphanumeric)	1	Non-keyed	
ERecType 10	Exposure recording type 2 (Upper Case Alphanumeric)	4	Non-keyed	
ErrFacC 1	assigned error factor 5 (Floating Point Scientific)	5 3	Non-keyed	
ErrorFac 1	Error factor 5 (Floating Point Scientific)	5 3	Non-keyed	
EventID 1	Basic event identifier 0 (Concatenated)	11	Alternate 20 5	Category Comp Design FailID NormStat

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
ExpOrign	Exposure data origin fields	4	Non-keyed	
1	2 (Upper Case Alphanumeric)			
FID	Plant (facility) code	4	Non-keyed	
4	2 (Upper Case Alphanumeric)			
FRecType	Failure recording type	4	Non-keyed	
10	2 (Upper Case Alphanumeric)			
FStar	Assigned number of failures	8	Non-keyed	
1	4 (Real)	2		
FailID	Failure mode id.	3	Non-keyed	
1	2 (Upper Case Alphanumeric)			
FailOrgn	Failure data origin	4	Non-keyed	
1	2 (Upper Case Alphanumeric)			
IRADAP	IRADAP suitable data?	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
InclCirc	Circuit included?	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
InclSys	Include System	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
LowerCon	Lower confidence bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
LowerTol	Lower tolerance bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
MeanRate	Assigned mean from probability	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
MedRate	Assigned median from probility	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
NormStat	Normal state	2	Non-keyed	
1	2 (Upper Case Alphanumeric)			
Nuclear	Nuclear plan data origin?	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
NumComp	Number of components	6	Non-keyed	
1	3 (Integer)			

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
NumFail 1	Number of failures 3 (Integer)	6	Non-keyed	
PFOnly 1	Primary failure only? 2 (Upper Case Alphanumeric)	1	Non-keyed	
PreAgg 1	Pre-aggregated data? 11 (Boolean) (Y/N)	1	Non-keyed	
PrimeKey 1	Primary Source Key 0 (Concatenated)	14	Primary 20 5	Category Comp Design FailID NormStat SourceID
Prob 1	Failure probability 5 (Floating Point Scientific)	5 3	Non-keyed	
ProbCon 1	Confidence bounds probability 3 (Integer)	2	Non-keyed	
ProbEF 1	Error factor probability 3 (Integer)	2	Non-keyed	
ProbTol 1	Tolerance bounds probability 3 (Integer)	2	Non-keyed	
ProbType 1	Failure probability type 2 (Upper Case Alphanumeric)	6	Non-keyed	
ProbUnit 1	Failure probability units 2 (Upper Case Alphanumeric)	3	Non-keyed	
RateC 1	Assigned failure rate 5 (Floating Point Scientific)	5 1	Non-keyed	
RawData 1	Raw Data in recored? 11 (Boolean) (T/F)	1	Non-keyed	
SafetyGr 1	Safety grade comp. (Y,N, or U) 2 (Upper Case Alphanumeric)	1	Non-keyed	
Severity 1	Failure severity (C,D,I, or X) 2 (Upper Case Alphanumeric)	1	Non-keyed	

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
SourceID 1	Source identifier 3 (Integer)	3	Non-keyed	
SrceDEnt 1	Data entry code 2 (Upper Case Alphanumeric)	3	Non-keyed	
SrceNtry 1	Date source record entered 7 (Date) (yyyy/mm/dd)	10	Non-keyed	
StanDev 1	Standard deviation 5 (Floating Point Scientific)	5 3	Non-keyed	
StanDevC 1	Assigned Standard deviation 5 (Floating Point Scientific)	5 3	Non-keyed	
SubSysTn 1	Subsystem train 1 (Alphanumeric)	36	Non-keyed	
System 5	System code 1 (Alphanumeric)	2	Non-keyed	
TotCDmnd 1	Total component demands 3 (Integer)	8	Non-keyed	
TotCmpHr 1	Total component exposure time 3 (Integer)	12	Non-keyed	
TotDmndC 1	Assigned total demand 3 (Integer)	8	Non-keyed	
TotHrsC 1	Assigned total time 3 (Integer)	12	Non-keyed	
TypeEF 1	Error factor type 2 (Upper Case Alphanumeric)	1	Non-keyed	
UndrLS2 1	underlying variance 5 (Floating Point Scientific)	5 3	Non-keyed	
UpperCon 1	Upper confidence bound 5 (Floating Point Scientific)	5 3	Non-keyed	
UpperTol 1	Upper tolerance bound 5 (Floating Point Scientific)	5 3	Non-keyed	
UpprTolC 1	assigned upper tolerance limit 5 (Floating Point Scientific)	5 3	Non-keyed	

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
VType	Generic variance type	2	Non-keyed	
1	2 (Upper Case Alphanumeric)			
Variance	Variance of the failure rate	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
VariatnC	Assigned variation	5	Non-keyed	
1	5 (Floating Point Scientific)	3		

Field Validation Report

--- 20 ---

Record Name : Source
Description : Source probability data

4) BayUpd Bayesian Update? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

6) Category Gen. component cat. (level 1) (Upper Case Alphanumeric)

Minimum Value ->1
Maximum Value ->2

22) Domestic Domestic Failure Data? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

30) FStar Assigned number of failures (Real)

Minimum Value -> .00

33) IRADAP IRADAP suitable data? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

34) InclCirc Circuit included? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

35) InclSys Include System (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

- E I

41) Nuclear Nuclear plan data origin? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

- N Y

44) PFOnly Primary failure only? (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

55) SafetyGr Safety grade comp. (Y,N, or U) (Upper Case Alphanumeric)

Enumerated List - 3 value(s)

N U Y

--- 21 ---

Record Name : System
Description : System descriptions
Fields : 2
Byte Length : 57
File Number : 6
Data File : HSYSTEM.DAT
Index File : HSYSTEM.IDX

Field Name / Repeat	Description / Type	Length / Decimals	Key Type K/Nod N/Cr	Sub-Fields
Des	System description	70	Non-keyed	
1	2 (Upper Case Alphanumeric)			
SystemID	System identifier	2	Primary	
1	1 (Alphanumeric)		20 5	

--- 22 ---

Record Name : UEvent
Description : Basic event information
Fields : 22
Byte Length : 57
File Number : 19
Data File : UEVENT.DAT
Index File : UEVENT.IDX

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
CatComp	Component ID (level 1 + 3 + 2)	6	Alternate	Category
1	0 (Concatenated)		20 5	Design Comp
Category	Gen. component cat. (level 1)	1	Non-keyed	
1	2 (Upper Case Alphanumeric)			
Comp	Gen. component name (level 2)	3	Non-keyed	
1	2 (Upper Case Alphanumeric)			
CompFID	Component ID + Failure	8	Alternate	Comp
1	0 (Concatenated)		20 5	Design FailID
CompID	Component ID (level 2 + 3)	5	Alternate	Comp
1	0 (Concatenated)		20 5	Design
CompIDNS	Component ID + Normal State	7	Alternate	Comp
1	0 (Concatenated)		20 5	Design NormStat
CompNS	Component + Normal State	5	Alternate	Comp
1	0 (Concatenated)		20 5	NormStat
CompNSFM	Component+Normal State+Failure	8	Alternate	Comp
1	0 (Concatenated)		20 5	FailID NormStat
Design	Component design (level 3)	2	Non-keyed	
1	2 (Upper Case Alphanumeric)			
DmandAgP	Demand aggregated probability	5	Non-keyed	
1	5 (Floating Point Scientific)	3		

RECORD REPORT FOR HARDWARE

Field Name Repeat	Description Type	Length Decimals	Key Type K/Nod N/Cr	Sub-Fields
DmandAgU	Demand aggregated upper bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
DmandAsP	Demand assigned probability	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
DmandAsU	Demand assigned upper bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
EventID	Unique event identifier	11	Primary	Category
1	0 (Concatenated)		20 5	Comp Design FailID NormStat
FailID	Failure mode ID	3	Non-keyed	
1	2 (Upper Case Alphanumeric)			
HrAgP	Hourly aggregated probability	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
HrAgU	Hourly aggregated upper bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
HrAsP	Hourly assigned probability	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
HrAsU	Hourly assigned upper bound	5	Non-keyed	
1	5 (Floating Point Scientific)	3		
NormStat	Normal state of comp/des/failr	2	Non-keyed	
1	2 (Upper Case Alphanumeric)			
SrceCnt	No. sources for this event	3	Non-keyed	
1	3 (Integer)			
SrceCntA	No. sources aggregated in Evnt	3	Non-keyed	
1	3 (Integer)			

Field Validation Report

--- 22 ---

Record Name : UEvent
Description : Basic event information

2) Category Gen. component cat. (level 1) (Upper Case Alphanumeric)

Minimum Value ->1
Maximum Value ->2

FORM REPORT FOR HARDWARE

lenuRA1 Main Ad Hoc search menu (H/W) 10 Field(s) Help Form -> HMenuRA1
packed picture size -> 738 bytes
predominant color -> white on black

Ad Hoc Search of Hardware Data
Main Menu

E - Exit from Ad Hoc Search Program
? - NUCLARR Hotlinc #

1 - Retrieve a Saved Data Records File
2 - Ad Hoc Search
3 - View the Located Data Records
4 - Aggregate Probabilities for Located Records
5 - Report on Located Data Records
6 - Plot from Located Data Records
7 - Generate ASCII File for dBase III/SAS/SPSS
8 - Save Located Data Records to a File

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
2	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
3	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
4	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
5	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
6	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
7	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
8	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
9	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
10	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	

MenuRB1 Main Browse/search menu (H/W) 11 Field(s) Help Form -> HMenuRB1
packed picture size -> 814 bytes
predominant color -> white on black

Descriptive Search of Hardware Data
Main Menu

E - Exit from Descriptive Search Program
? - NUCLARR Hotline #

1 - Retrieve a Saved Data Records File
2 - Begin Descriptive Search
3 - "Tailored" Selection and Aggregation
4 - View the Located Data Records
5 - Aggregate Probabilities for Located Records
6 - Report on Located Data Records
7 - Plot from Located Data Records
8 - Generate ASCII File for dBase III/ AS/SPSS
9 - Save Located Data Records to a Fil.

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
2	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
3	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
4	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
5	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
6	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
7	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
8	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
9	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
10	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
11	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	

FORM REPORT FOR HARDWARE

MenuRH1 Main retrieve menu for hardwar 6 Field(s) Help Form -> HMenuRH1
packed picture size -> 752 bytes
predominant color -> white on black

Retrieval of
NUclear Computerized Library for Assessing Reactor Reliability
Hardware Component Failure Data

Main Retrieve Menu
Version 1.1

E - Exit from Hardware Retrieve Program
? - NUCLARR Hotline #

1 - Locate Data Records by Description
2 - Locate Data Records by Ad Hoc Search
3 - Review Documents
4 - Hardware Glossary of Terms

Press <Esc> for general NUCLARR Help

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
2	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
3	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
4	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
5	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	
6	Highlighted	Utility (UA1) - 1	Upper Case Alphanumeric	

HMenuRA1 Help for Ad Hoc Search Menu 1 Field(s) Help Form -> EXIT
packed picture size -> 865 bytes
predominant color -> light yellow on blue

Ad Hoc Search of Hardware Data

E - Exit from Ad Hoc Search Program: Exit to Main Search Program Menu. User will be reminded to save any existing search results not already saved by option number 7 below.

? - Data Clearinghouse: Display Data Clearinghouse telephone number.

1 - Retrieve a Saved Search File: A previously saved search file may be requested and brought into memory for reporting, aggregating, and ASCII file generation.

2 - Ad Hoc Search: Begin the Ad Hoc Search function which will display a series of screens on which specific values may be selected to initiate a search of the hardware source data.

1 Press <Esc> to continue <Enter> return to Ad Hoc Menu

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Entry/Display	Utility (DA1) - 1	Alphanumeric	HMnuRA-2

HMenuRB1 Help-Main Browse/search menu 1 Field(s) He'p Form -> EXIT
packed picture size -> 708 bytes
predominant color -> light yellow on blue

Descriptive Search of Hardware Data

E - Exit from Descriptive Search Program. Exit to Main Search Program Menu. User will be reminded to save any existing search results not already saved by option number 8 below.

? - NUCLARR Hotline # - Display Data Clearinghouse telephone number for assistance or further system information.

1 - Retrieve a Saved Search File: A previously saved search file may be requested and brought into memory for reporting, aggregating, and ASCII file generation.

1 Press <Esc> continue <Enter> return to Descriptive search

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Entry/Display	Utility (DA1) - 1	Alphanumeric	HMnuRB-2

HMnuRA-2 Cont. help for Ad Hoc Search 1 Field(s) Help Form -> EXIT
packed picture size -> 851 bytes
predominant color -> light yellow on blue

Ad Hoc Search of Hardware Data

3 - View Located Data Records: View the Source records currently in the buffer and all associated Event aggregations. If the aggregation option has not yet been selected, Auto aggregation values from the database will be displayed.

4 - Aggregate Probabilities for Located Records: Compute probabilities of the aggregated hardware source data which results from the Ad Hoc search option number 2 above.

5 - Report on Located Data Records: Report on the screen, printer, or to a file all events and data sources resulting from the Ad Hoc search option number 2 above.

1 Press <Esc> to continue <Enter> for previous Help page

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Entry/Display	Utility (DA1) - 1	Alphanumeric	HMnuRA-3

HMnuRA-3 Cont. help for Ad Hoc Search 0 Field(s) Help Form -> EXIT
packed picture size -> 763 bytes
predominant color -> light yellow on blue

Ad Hoc Search of Hardware Data

6 - Plot from Located Data Records: Generate log plots from located data records. View plots on the console and/or print on an Epson or laserjet printer.

7 - Generate ASCII File for dBase III/SAS/SPSS: Generate and save an ASCII file of aggregated event probabilities suitable for use by dBase III, or in SAS and SPSS programs.

8 - Save Located Records to a Search File: Save the results of the Ad Hoc search function number 2 in a user named data file for future use in NUCLARR.

<Esc> return to Ad Hoc menu <Enter> for previous Help page

HMnuRB-2 Cont.Help-Main Browse/srch mnu 1 Field(s) Help Form -> EXIT
packed picture size -> 1051 bytes
predominant color -> light yellow on blue

Descriptive Search of Hardware Data

2 - Begin Descriptive Search: Begin the Descriptive Search function which displays descriptions of all hardware components, designs, failures, normal states, applications, etc., following the taxonomy described in report number EGG-REQ-7742. Selections may be made at each taxonomy level thus narrowing the number of source records to be searched as each taxonomy level is descended.

3 - "Tailored" Aggregation and Selection: Extend the Descriptive Search option 2 or file option 1 with a quick selection screen for Failure Data Origin; Domestic or Foreign, Nuclear or Non-Nuclear, Safety or Non-Safety Grade, and Quality Coded Data Fields. This function will complete the selection, aggregate, and display the aggregated events and selected source records.

1 Press <Esc> to continue <Enter> for previous Help page

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Entry/Display	Utility (DA1) - 1	Alphanumeric	HMnuRB-3

HMnuRB-3 Cont.Help-Main Browse/srch mnu 1 Field(s) Help Form -> EXIT
packed picture size -> [illegible] bytes
predominant color -> light yellow on blue

Descriptive Search of Hardware Data

4 - View Located Data Records: View the Source records currently in the buffer and all associated Event aggregations. If the aggregation option has not yet been selected, Auto aggregation values from the database will displayed.

5 - Aggregate Probabilities for Located Records: Compute aggregated probabilites using the source data selected during the Descriptive Search option number 2.

6 - Report on Located Data Records: Report on the screen, printer, or to a file all events and data sources resulting from the Descriptive Search option number 2 above.

1 Press <Esc> to continue <Enter> for previous Help page

Field	Display Type	Record & (Field) - rpt	Field Type	Help Frm
1	Entry/Display	Utility (DA1) - 1	Alphanumeric	HMnuRB-4

HMnuRB-4 ..nt.Help-Main Browse/srch mnu 0 Field(s) Help Form -> EXIT
packed picture size -> 739 bytes
predominant color -> light yellow on blue

Descriptive Search of Hardware Data

7 - Plot from Located Data Records: Generate log plots from located data records. View plots on the console and/or on an Epson or laserjet printer.

8 - Gererate ASCII File for dBase ..../SAS/SPSS: Generate and save an ASCII file of aggregate event probabilities suitable for use by dBase III, or ... SAS and SPSS programs.

9 - Save Located Records to a Search File: Save the results of the Descriptive Search function number 2 in a user named data file.

<Esc> to return to menu <Enter> for previous Help page

APPENDIX G

SAMPLE SOURCE LISTING FOR HEP DATA PROCESSING

APPENDIX G

SAMPLE SOURCE LISTING FOR HEP DATA PROCESSING

This appendix presents a sample of some source listings for a library and a module used for HEP data processing. The DOCUPROC utility, which was used to generate these listings, provides line numbering, a table of contents, and an index. The line numbering is restarted for each module-level code element.

Table of Contents

```
 1: DEFINITION MODULE NUCGEN;
 2:
 3: FROM Reports IMPORT FileRP;
 4:
 5: EXPORT QUALIFIED (* procedures *)
 6:
 7:                   CalcMedHEPEF,
 8:                   Card2StrOFill,
 9:                   Clearinghouse,
10:                   ClearStack,
11:                   ConvertErrFailToUErrUFail,
12:                   ConvertToDataManualPage,
13:                   DataCheck,
14:                   DocCodes,
15:                   DocType,
16:                   GetMatrixDesc,
17:                   GetRowDescr,
18:                   GetVerb,
19:                   GetCellType,
20:                   GetCellValidity,
21:                   MyOpenRelation,
22:                   MyCloseRelation,
23:                   OutputFile,
24:                   SearchDoc,
25:
26:                   (* types *)
27:
28:                   CellType,
29:                   ValidType,
30:
31:                   (* VAR *)
32:
33:                   ftbl;
34:
35: TYPE
36:       ValidType    = (VALID, INVALID, SHADED);
37:       CellType     = (FUNCTIONALGROUP, CELL);
38:
39: VAR
40:       ftbl : FileRP;  (* In overlays - each overlay must have
41:                          it own ftbl declared and report files must be
42:                          OPENED and CLOSED in side the overlay.
43:                          this is a different file then in the main
44:                          program.
45:                       *)
46:
47:
48: PROCEDURE CalcMedHEPEF (VAR meanHEP     :REAL;
49:                            UCB, LCB     :REAL;
50:                         VAR ef          :CARDINAL;
51:                         VAR medianHEP   :REAL;
52:                         VAR success     :BOOLEAN);
53:
54: (**
55:        CalcMedHEPEF - Calculate Error-Factor & Median-HEP
56:
57:        T. H. Tucker - March 1987.
58:
59:        CalcMedHEPEF calculates the Median HEP & Error Factor values
60:                     when given the Mean HEP & Upper Confidence Bound
61:                     values.
62:
63:        CALLING SEQUENCE -
64:
65:           CalcMedHEPEF (meanHEP, UCB, ef, MedianHEP, success);
66:
67:        ENTRY - meanHEP  is the calling program provided Mean HEP value
68:                UCB      is the calling program provided Upper Confidence Bound
69:                LCB      is the calling program provided Lower Confidence Bound
70:
71:        EXIT  - ef         is the calculated Error Factor for Median HEP
72:                medianHEP  is the calculated Median HEP
73:                success    is a BOOLEAN flag indicating whether computation
74:                           was successful (TRUE = successful)
75:
76:        FILES -
77:
```

```
78:         ALGORITHM - medianHEP    = meanHEP / Exp (Power (sigma, 2.0) / 2.0)
79:                     ef           = Exp (k * sigma)  <-- SUPERCEDED!!!
80:                     ef           = Exp (ln(UCB/LCB)/2.0)
81:
82:                     where, sigma = (UCB - meanHEP) / k
83:                     and  k is a constant.
84: *)
85:
86:
87: PROCEDURE Card2StrOFill (i,width: CARDINAL;  VAR string: ARRAY OF CHAR);
88:
89: (**
90:         Card2StrOFill - Fill string with zeros.
91:
92:         T. H. Tucker  - March 1986.
93:
94:         This procedure fills the blanks in a string with zeroes.
95:
96:         CALLING SEQUENCE -
97:
98:           Card2StrOFill (i, width, string)
99:
100:        ENTRY -
101:
102:          i       - CARDINAL
103:                    The cardinal that is to change to a string.
104:          width   - CARDINAL
105:                    The width of the cardinal.
106:
107:        EXIT -
108:
109:          string - ARRAY OF CHAR
110:                   The string that represents the cardinal.
111:
112:        FILES -
113: *)
114:
115:
116: PROCEDURE Clearinghouse;
117:
118: (**
119:         Clearinghouse - Displays clearinghouse number.
120:
121:         P. M. McGuire - September 1986.
122:
123:         Display of the clearinghouse number.
124:
125:         CALLING SEQUENCE -
126:
127:           Clearinghouse ()
128:
129:         ENTRY -
130:
131:         EXIT -
132:
133:         FILES -
134: *)
135:
136:
137: PROCEDURE ClearStack
138:           ()        ;
139:
140: (**
141:     ClearStack - Clears off any extraneous junk from stack
142:
143:     G. H. Beers - December 1987
144:
145:     description
146:
147:     CALLING SEQUENCE -
148:
149:       ClearStack ()
150:
151:     ENTRY -
152:
```

```
153:
154:     EXIT -
155:
156:
157:     FILES -
158: *)
159:
160:
161: PROCEDURE ConvertErrFailToUErrUFail(ErrFail: CARDINAL;
162:                                     VAR UErr :ARRAY OF CHAR; VAR UFail :CHAR);
163:
164: (**
165:         ConvertErrFailToUErrUFail - Conversion of ErrFail to CHAR.
166:
167:         P. M. McGuire - September 1986
168:
169:         Converts the integer ErrFail to a two character representation.
170:
171:         CALLING SEQUENCE -
172:
173:           ConvertErrFailToUErrUFail(ErrFail, UErr, UFail)
174:
175:         ENTRY -
176:
177:           ErrFail - CARDINAL
178:                     The errfail for a source statement
179:
180:         EXIT -
181:
182:           UErr    - ARRAY OF CHAR
183:                     The data type : RecoveryConsidered / RecoveryNotConsidered
184:           UFail   - CHAR
185:                     The failure mode : Commission or Omission.
186:
187:
188:         FILES -
189: *)
190:
191:
192: PROCEDURE ConvertToDataManualPage(VAR dataManualPage: ARRAY OF CHAR;
193:                                   matrix, rowID, colID:CARDINAL);
194:
195:  (**
196:         ConvertToDataManualPage - Card to Char dataManualPage.
197:
198:         P. M. McGuire - September 1986.
199:
200:         Conversion of the cardinal matrix, row, and column to
201:         character representation of dataManualPage number is completed
202:         with the preceeding zeros filled in the dataManualPage number.
203:
204:         CALLING SEQUENCE -
205:
206:           ConvertToDataManualPage (dataManualPage, matrix, rowID,
207:                                    colID)
208:
209:
210:         ENTRY -
211:
212:           matrix          - CARDINAL
213:                             Matrix number
214:           rowID           - CARDINAL
215:                             Row number.
216:           colID           - CARDINAL
217:                             Column number.
218:
219:         EXIT -
220:
221:           dataManualPage - ARRAY OF CHAR
222:                            Character representation of matrix, row and
223:                            column with zeros filled for blanks.
224:
225:         FILES -
226: *)
227:
```

```
228:
229: PROCEDURE DataCheck(matr         : CARDINAL;
230:                     matrName     : ARRAY OF CHAR;
231:                     row          : CARDINAL;
232:                     rowName      : ARRAY OF CHAR;
233:                     col          : CARDINAL;
234:                     colName      : ARRAY OF CHAR;
235:                     relationName: ARRAY OF CHAR;
236:                     dataKey      : ARRAY OF CHAR):BOOLEAN;
237:
238: (**
239:        DataCheck - Check if data exists for a matrix/row/cell
240:
241:        P. M. McGuire  September 1986
242:
243:        Check if data exists in the relation specified in relationName,
244:        matrix, row, or cell using "dataKey" to read the relation.
245:        Return TRUE if data exists.  The relation is opened and closed
246:        in DataCheck with a no modify option.  The record is read on an
247:        alternate key with a EQ sage operation.  The dataKey can be one
248:        of three options:  UKeyM   - search on the matrix given only.
249:                           UKeyMR  - search on the matrix and row given.
250:                           UKeyMRC - search on the matrix and row and
251:                                     col given.
252:
253:        CALLING SEQUENCE -
254:
255:          DataCheck (matr, row, col, relationName, dataKey ): BOOLEAN
256:
257:        ENTRY -
258:
259:          matr         - CARDINAL
260:                         Matrix number
261:          matrName     - ARRAY OF CHAR
262:                         The field name for the matrix value.
263:          row          - CARDINAL
264:                         Row ID
265:          rowName      - ARRAY OF CHAR
266:                         The field name for the row value.
267:          col          - CARDINAL
268:                         Column ID
269:          colName      - ARRAY OF CHAR
270:                         The field name for the column value.
271:          relationName - ARRAY OF CHAR
272:                         The relation being checked
273:          dataKey      - ARRAY OF CHAR
274:                         Alternate key name in HEP.
275:                         It can be UKeyM, UKeyMR or UKeyMRC
276:
277:        EXIT -
278:
279:          BOOLEAN - TRUE = data exists, FALSE = no data
280:
281:        FILES -
282:
283:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
284:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
285:        ********    ********    *****  *****       *****       *******
286:
287:        relation-   O(F)/C      read   EQ          UKeyM
288:        Name                                       UKeyMR
289:                                                   UKeyMRC
290: *)
291:
292:
293: PROCEDURE DocCodes;
294:
295: (**
296:        DocCodes - Retrieves "Original" and "Reference".
297:
298:        P. M. McGuire  September 1986
299:
300:        Expands Codes from "Referenc" to the description of
301:        "Original" and "Reference" and loads it into the
302:        Utility fields "Text" and "Text2" respectively for display.
```

```
303:        DocCodes gets the codes from the current "Referenc" record.
304:
305:        CALLING SEQUENCE -
306:
307:          DocCodes()
308:
309:        ENTRY -
310:
311:        EXIT -
312:
313:
314:        FILES -
315:
316:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
317:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
318:        --------    ---------   -----  ----        -----       -------
319:
320:        Referenc    O(Prior)    read               Codes
321:
322:        Utility                 write                          Text
323:                                                               Text2
324: *)
325:
326:
327: PROCEDURE DocType;
328:
329: (**
330:        DocType - Expands the document type code into a description.
331:
332:        P. M. McGuire - September 1986
333:
334:        Using the 2 letter document code from "Referenc" the
335:        description is loaded into the Utility "Text30" field.
336:        DocType gets docType of the current "Referenc" record.
337:
338:        CALLING SEQUENCE -
339:
340:          DocType()
341:
342:        ENTRY -
343:
344:        EXIT -
345:
346:        FILES -
347:
348:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
349:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
350:        --------    ---------   -----  -----       -----       -------
351:
352:        Referenc    O(Prior)    read               docType
353:
354:        Utility                 write                          Text30
355: *)
356:
357:
358: PROCEDURE GetCellType (VAR typeOfCell : CellType;
359:                            rowID      : CARDINAL);
360:
361: (**
362:        GetCellType - Determination of cell type.
363:
364:        T. H. Tucker - March 1986.
365:
366:        GetCellType determines the cell type, either Functional
367:        Group Cell or an individual Cell.  Row numbers ending in
368:        0 are functional groups.
369:
370:        CALLING SEQUENCE -
371:
372:          GetCellType (typeOfCell, rowID)
373:
374:        ENTRY -
375:
376:          rowID      - CARDINAL
377:                       The row number of the cell selected.
```

```
378:
379:        EXIT -
380:
381:          typeOfCell - CellType
382:                       CellType = (FUNCTIONALGROUP, CELL)
383:                       Returns the type of the cell selected.
384:
385:        FILES -
386: *)
387:
388:
389: PROCEDURE GetCellValidity (matrix, rowID, colID : CARDINAL;
390:                            VAR validity : ValidType);
391:
392: (**
393:        GetCellValidity - Cell validity check.
394:
395:        T. H. Tucker - March 1986.
396:
397:
398:        GetCellValidity determines if the cell requested by the
399:        user is valid. "CellVal" is opened and closed with a no modify
400:        option.  The record is read on the PrimeKey with an EQ sage
401:        operation.  The PrimeKey consisting of the matrix, rowID, and
402:        colID.
403:
404:        CALLING SEQUENCE -
405:
406:          GetCellValidity (matrix, rowID, colID, validity)
407:
408:        ENTRY -
409:
410:          matrix   - CARDINAL
411:                     matrix number
412:          rowID    - CARDINAL
413:                     row identification number
414:          colID    - CARDINAL
415:                     column identification number
416:
417:        EXIT -
418:
419:           validity - ValidType
420:                      ValidType    = (VALID, INVALID, SHADED)
421:                      Returns the validity of the cell selected.
422:
423:        FILES -
424:
425:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
426:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
427:        --------    ---------   -----  -----       -----       -------
428:
429:        CellVal     O(F)/C      read   EQ          PrimeKey
430: *)
431:
432:
433: PROCEDURE GetMatrixDesc (matrix : CARDINAL);
434:
435: (**
436:        GetMatrixDesc -  Gets plant ID, job classification & taxonomy level
437:
438:        D. J. Fink - July 1986.
439:
440:        Opens, reads, and closes the relation MatrDesc to get the
441:        matrix's plant ID, job classification, and taxonomy level.
442:
443:
444:        CALLING SEQUENCE -
445:
446:          GetMatrixDesc (matrix)
447:
448:        ENTRY -
449:
450:          matrix - CARDINAL
451:                   The matrix number associated with the description.
452:
```

```
453:         EXIT -
454:
455:         FILES -
456:
457:         RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
458:         NAME        CLOSE       WRITE  OP          KEYS        CHANGES
459:         --------    ---------   -----  -----       -----       -------
460:
461:         MatrDesc    O(F)/C      read   EQ          Matrix
462: *)
463:
464:
465: PROCEDURE GetRowDescr (matrix, rowID: CARDINAL);
466:
467: (**
468:         GetRowDescr - Retrieval of row description.
469:
470:         T. H. Tucker - March 1986.
471:
472:         Retrieval of row description.  MATRROW and Rows are opened.
473:         Sage operation GE and EQ is used in reading the records.
474:
475:         CALLING SEQUENCE -
476:
477:           GetRowDescr ( matrix, rowID)
478:
479:         ENTRY -
480:
481:           matrix    - CARDINAL
482:                       The matrix number.
483:           rowID     - CARDINAL
484:                       The row number.
485:
486:         EXIT -
487:
488:         FILES -
489:
490:         RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
491:         NAME        CLOSE       WRITE  OP          KEYS        CHANGES
492:         --------    ---------   -----  -----       -----       -------
493:
494:         MATRROW     O(F)        read   GE          PrimeKey
495:         Rows        O(F)/C      read   EQ          RowNumbr
496: *)
497:
498:
499: PROCEDURE GetVerb ( matrix, colID : CARDINAL;
500:                     VAR sageerror : CARDINAL);
501: (**
502:         GetVerb - Retrieves Human Action Verb.
503:
504:         P. M. McGuire - September 1986.
505:
506:         GetVerb retrieves the human action verb.  Human action verb
507:         is dependent on the matrix and colID.  MATRCOL and Columns are
508:         opened and sage operation GE and EQ is used in reading the
509:         records.
510:
511:         CALLING SEQUENCE -
512:
513:           GetVerb (matrix, colID, sageerror)
514:
515:         ENTRY -
516:              matrix - CARDINAL
517:                       The matrix number.
518:               colID - CARDINAL
519:                       The column number.
520:
521:         EXIT -
522:
523:            sageerror - CARDINAL
524:                        The sageerror from the MATRCOL read operation.
525:
526:
527:         FILES -
```

```
528:
529:         RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
530:         NAME        CLOSE       WRITE  OP          KEYS        CHANGES
531:         --------    ---------   -----  -----       -----       -------
532:
533:         MATRCOL     O(F)/C      read   GE          PrimeKey
534:         Columns     O(F)/C      read   EQ          ColNumbr
535: *)
536:
537:
538: PROCEDURE  MyCloseRelation (relation :ARRAY OF CHAR;
539:                             status   :BOOLEAN);
540:
541: (**
542:     MyCloseRelation - Closes relation or data file as necessary
543:
544:     G. H. Beers - December 1987
545:
546:     Checks the initial status of a relation and returns it to its initial state.
547:
548:     CALLING SEQUENCE -
549:
550:       MyCloseRelation (relation,status)
551:
552:     ENTRY -
553:
554:       relation - ARRAY OF CHAR
555:                  Name of relation to be opened or checked.
556:
557:       status   - BOOLEAN
558:                  inital relation status flag
559:                  TRUE  - Relation or index file was initially open
560:                  FALSE - Relation was initially closed
561:
562:     EXIT  -
563:
564:     FILES -
565: *)
566:
567:
568: PROCEDURE  MyOpenRelation (relation, key :ARRAY OF CHAR) :BOOLEAN;
569:
570: (**
571:     MyOpenRelation - Opens relation if necessary and returns initial status
572:
573:     G. H. Beers - December 1987
574:
575:     Checks the current status of a relation and opens it if necessary.
576:
577:     CALLING SEQUENCE -
578:
579:       MyOpenRelation (relation,key)
580:
581:     ENTRY -
582:
583:       relation - ARRAY OF CHAR
584:                  Name of relation to be opened or checked.
585:
586:       key      - ARRAY OF CHAR
587:                  Name of key to use for testing relation current status
588:
589:     EXIT  -
590:
591:       result   - BOOLEAN
592:                  initial relation status flag
593:                  TRUE - Relation or index file was initially open
594:
595:                  FALSE - Relation was closed
596:
597:      FILES -
598: *)
599:
600:
601: PROCEDURE OutputFile (name : ARRAY OF CHAR;
602:                       VAR ok : BOOLEAN;
```

```
603:                        dataManualPage: ARRAY OF CHAR;
604:                        VAR screenDisplay: BOOLEAN);
605:
606: (**
607:        OutputFile - Opens output file.
608:
609:        P. M. McGuire - September 1986.
610:
611:        Opens the WrForm file for output.
612:
613:        CALLING SEQUENCE -
614:
615:          OutputFile(name, ok, dataManualPage, screenDisplay)
616:
617:        ENTRY -
618:
619:          name            - ARRAY OF CHAR
620:                            Title of print menu
621:
622:          dataManualPage - ARRAY OF CHAR
623:                            The dataManualPage number for display.
624:
625:        EXIT -
626:
627:          ok              - BOOLEAN
628:                            Success of the opening operation
629:
630:          screenDisplay   - BOOLEAN
631:                            Display to the screen = True
632:                            Write to a file or PRN = False
633:
634:        FILES -
635:
636:
637:        FILE        OPEN        DEVICE              FORMS
638:        NAME        CLOSE                           WRITTEN   DESCRIPTION
639:        *****       *****       *******             *******   ***********
640:
641:        ftbl        O           filename
642: *)
643:
644:
645: PROCEDURE SearchDoc;
646:
647: (**
648:        SearchDoc - Search of source statements by document.
649:
650:        P. M. McGuire - September 1986.
651:
652:        Generation of document listing by source statement.
653:
654:        CALLING SEQUENCE -
655:
656:          SearchDoc()
657:
658:        ENTRY -
659:
660:        EXIT -
661:
662:        FILES -
663:
664:        RELATION    OPEN(T/F)  READ/    SAGE      FIELDS/           FIELD
665:        NAME        CLOSE      WRITE    OP        KEYS              CHANGES
666:        ********    *********  *****    *****     *****             *******
667:
668:        CellData    O(F)/C     read     EQ        DocKey
669:                                                  RefKey
670:
671:        Reference   O(prior)   read               DocNumbr
672:
673:        FILE        OPEN       DEVICE   FORMS
674:        NAME        CLOSE                WRITTEN   DESCRIPTION
675:        *****       *****      *******  *******   ***********
676:
677:        ftbl        O/C        PRN      RecDoc1   doc. description
```

```
678:                                    RepDoc2   header for
679:                                              Celldata
680:                                    RepDoc3
681: *)
682:
683:
684: END NUCGEN.
```

```
 1: IMPLEMENTATION MODULE NUCGEN;
 2:
 3: FROM Sage     IMPORT
 4:                DisplayMessage, DisplayForm,    GetFieldA,   GetFieldC,
 5:                PutFieldC,      WriteRecord,    CompareFieldC,
 6:                OpenRelation,   CloseRelation,  CloseRelationFiles,
 7:                CopyField,      ClearRelation,  ClearField,  GetFieldB,
 8:                PutFieldA,      ReadRecord,     SageError,
 9:                SageOperations, GetRepeatName,  FindRecord;
10:
11: FROM SortLib  IMPORT
12:                FillChar;
13:
14: FROM Reports  IMPORT
15:                OpenReport, DefineHeader, WrForm, TopOfPage,
16:                CloseReport, WrString, WrLn;
17:
18: FROM Convert  IMPORT
19:                StrToCard,   CardToStr;
20:
21: FROM Files    IMPORT
22:                File, FileState, GetFileName, State;
23:
24: FROM MathLib  IMPORT
25:                Exp, Ln, Power, Sqrt;
26:
27: FROM String   IMPORT
28:                Concat;
29:
30: FROM Terminal IMPORT
31:                CondRead;
32:
33: FROM ThorPort IMPORT
34:                Pause;
35:
36:
37: CONST
38:   RingBell  = TRUE;
39:   NoBell    = FALSE;
40:   Modify    = TRUE;
41:   NoModify  = FALSE;
42:   Default   = FALSE;
43:
44:
45: PROCEDURE CalcMedHEPEF (VAR meanHEP   :REAL;
46:                            UCB, LCB  :REAL;
47:                        VAR EF        :CARDINAL;
48:                        VAR medianHEP :REAL;
49:                        VAR success   :BOOLEAN);
50:
51: (**
52:        CalcMedHEPEF - Calculate Error-Factor & Mean-HEP
53:
54:        T. H. Tucker - March 1987.
55:
56:        CalcMedHEPEF calculates the Mean HEP & Error Factor values when
57:                     given the Median HEP & Confidence Bound values.
58:
59:        CALLING SEQUENCE -
60:
61:           CalcMedHEPEF (meanHEP,UCB,LCB,EF,MedianHEP, success);
62:
63:        ENTRY -
64:                medianHEP is the calculated Median HEP
65:                UCB       is the calling program provided Upper Confidence Bound
66:                LCB       is the calling program provided Lower Confidence Bound
67:
68:        EXIT  - EF        is the calculated Error Factor for Median HEP
69:                meanHEP   is the calling program provided Mean HEP value
70:                success   is a BOOLEAN flag indicating whether computation
71:                          was successful (TRUE = successful)
72:
73:        FILES -
74:
```

```
75:         ALGORITHM - meanHEP = medianHEP * Exp (Power (sigma, 2.0) / 2.0)
76:                     EF      = Sqrt (UCB/LCB))
77:
78:                     where, sigma = (UCB - meanHEP) / k
79:                     and  k is a constant.
80: *)
81:
82: CONST          (* ********************************* *)
83:   k = 1.6449;  (* <-- NEVER ALLOWED TO BE ZERO *** *)
84:                (* ********************************* *)
85: VAR
86:   sigma :REAL;
87:
88: BEGIN
89:   success     := TRUE;
90:   EF := 0;
91:
92:   IF (LCB > 0.0) THEN
93:     EF := TRUNC (Sqrt(UCB/LCB) + 0.5);
94:   ELSIF (medianHEP > 0.0) THEN
95:     EF := TRUNC ((UCB/medianHEP) + 0.5);
96:   ELSE
97:     DisplayMessage ("* ERROR - error factor could not be calculated",TRUE);
98:     Pause (2000);
99:   END;
100:
101:                                    (* Use A6 algorithm to get meanHEP *)
102:   IF (EF # 0) THEN
103:     meanHEP := medianHEP * Exp(0.5 * (Power((Ln( FLOAT(EF) )/1.6449),2.0
104:   ELSE
105:     success := FALSE;
106:     DisplayMessage ("* ERROR - CalcMedHEPEF, attempted:  Ln(0)", TRUE);
107:     Pause (2000);
108:   END;
109:
110: END CalcMedHEPEF;
111:
112:
113: PROCEDURE Card2StrOFill (i,width: CARDINAL;  VAR string: ARRAY OF CHAR);
114:
115: (**
116:         Card2StrOFill - Fill string with zeros.
117:
118:         T. H. Tucker  - March 1986.
119:
120:         This procedure fills the blanks in a string with zeroes.
121:
122:         CALLING SEQUENCE -
123:
124:           Card2StrOFill (i, width, string)
125:
126:         ENTRY -
127:
128:           i      - CARDINAL
129:                    The cardinal that is to change to a string.
130:           width  - CARDINAL
131:                    The width of the cardinal.
132:
133:         EXIT -
134:
135:           string - ARRAY OF CHAR
136:                    The string that represents the cardinal.
137:
138:
139:         FILES -
140: *)
141:
142:
143: VAR
144:   done:  BOOLEAN;
145:   j:  CARDINAL;
146:
147:
148: BEGIN
149:   CardToStr (i, string, width, done);
```

```
150:   j := 0;
151:   LOOP
152:     IF (string[j] = " ") THEN
153:       string[j] := "0";
154:     ELSE
155:       EXIT;
156:     END;
157:     INC (j);
158:     IF (j >= width) THEN EXIT; END;
159:   END;
160:
161: END Card2StrOFill;
162:
163:
164: PROCEDURE Clearinghouse;
165:
166: (**
167:         Clearinghouse - Displays clearinghouse number.
168:
169:         P. M. McGuire - September 1986.
170:
171:         Display of the clearinghouse number.
172:
173:         CALLING SEQUENCE -
174:
175:           Clearinghouse ()
176:
177:         ENTRY -
178:
179:         EXIT -
180:
181:         FILES -
182: *)
183:
184:
185: CONST
186:   mess =
187: "Telephone Clearinghouse for assistance at (208)526-0735 or FTS 583-0735"
188:                                                                        ;
189:
190: BEGIN
191:   DisplayMessage(mess,NoBell);
192:
193: END Clearinghouse;
194:
195:
196: PROCEDURE ClearStack
197:           ()       ;
198:
199: (**
200:     ClearStack - Clears off any extraneous junk from stack
201:
202:     G. H. Beers - December 1987
203:
204:     description
205:
206:     CALLING SEQUENCE -
207:
208:       ClearStack ()
209:
210:     ENTRY -
211:
212:
213:     EXIT  -
214:
215:
216:     FILES -
217: *)
218:
219:
220: VAR
221:   ch       :CHAR;
222:   success  :BOOLEAN;
223:
224: BEGIN
```

```
225:
226:   LOOP
227:     CondRead (ch, success);
228:     IF NOT success THEN
229:       EXIT;
230:     END;
231:   END;  (* LOOP *)
232:
233: END ClearStack;
234:
235:
236: PROCEDURE ConvertErrFailToUErrUFail
237:           (    ErrFail : CARDINAL;
238:            VAR UErr    : ARRAY OF CHAR;
239:            VAR UFail   : CHAR          );
240:
241: (**
242:         ConvertErrFailToUErrUFail - Conversion of ErrFail to CHAR.
243:
244:         P. M. McGuire - September 1986
245:
246:         Modified by T.H. Tucker, April 1987, for new Data-Type / Failure Mode
247:         format:
248:
249:                      = 1) RecoveryConsidered    / omission
250:            ErrFail   = 2) RecoveryConsidered    / commission
251:                      = 3) RecoveryNotConsidered / omission
252:                      = 4) RecoveryNotConsidered / commission
253:
254:         Converts the integer ErrFail to a two character representation.
255:
256:         CALLING SEQUENCE -
257:
258:           ConvertErrFailToUErrUFail(ErrFail, UErr, UFail)
259:
260:         ENTRY -
261:
262:           ErrFail  - CARDINAL
263:                      The errfail for a source statement
264:
265:         EXIT -
266:
267:           UErr     - ARRAY OF CHAR
268:                      The Data Type : RecoveryConsidered / RecoveryNotConsidered
269:           UFail    - CHAR
270:                      The Failure Mode : Commission / Omission.
271:
272:
273:
274:         FILES -
275: *)
276:
277: BEGIN
278:
279:   FillChar (UErr,  " ",  4);
280:   FillChar (UFail, " ",  1);
281:
282:   IF ErrFail = 1 THEN
283:     FillChar (UErr[0], "R",   1);
284:     FillChar (UErr[1], "C",   1);
285:     UFail := "O";
286:   ELSIF ErrFail = 2 THEN
287:     FillChar (UErr[0], "R",   1);
288:     FillChar (UErr[1], "C",   1);
289:     UFail := "C";
290:   ELSIF ErrFail = 3 THEN
291:     FillChar (UErr[0], "R",   1);
292:     FillChar (UErr[1], "N",   1);
293:     FillChar (UErr[2], "C",   1);
294:     UFail := "O";
295:   ELSIF ErrFail = 4 THEN
296:     FillChar (UErr[0], "R",   1);
297:     FillChar (UErr[1], "N",   1);
298:     FillChar (UErr[2], "C",   1);
299:     UFail := "C";
```

```
300:   END;
301:
302: END ConvertErrFailToUErrUFail;
303:
304:
305: PROCEDURE ConvertToDataManualPage(VAR dataManualPage: ARRAY OF CHAR;
306:                                       matrix, rowID, colID:CARDINAL);
307:
308:  (**
309:         ConvertToDataManualPage - Card to Char dataManualPage.
310:
311:         P. M. McGuire - September 1986.
312:
313:         Conversion of the cardinal matrix, row, and column to
314:         character representation of dataManualPage number is completed
315:         with the preceeding zeros filled in the dataManualPage number.
316:
317:         CALLING SEQUENCE -
318:
319:           ConvertToDataManualPage (dataManualPage, matrix, rowID,
320:                                    colID)
321:
322:
323:         ENTRY -
324:
325:           matrix          - CARDINAL
326:                             Matrix number
327:           rowID           - CARDINAL
328:                             Row number.
329:           colID           - CARDINAL
330:                             Column number.
331:
332:         EXIT -
333:
334:           dataManualPage - ARRAY OF CHAR
335:                            Character representation of matrix, row and
336:                            column with zeros filled for blanks.
337:
338:         FILES -
339: *)
340:
341:
342: VAR
343:   tempmat : ARRAY [0..1] OF CHAR;
344:   temprow : ARRAY [0..2] OF CHAR;
345:   tempcol : ARRAY [0..1] OF CHAR;
346:
347:
348: BEGIN
349:
350: Card2StrOFill (matrix, 2,tempmat);
351: dataManualPage[0] := tempmat[0];
352: dataManualPage[1] := tempmat[1];
353:
354: Card2StrOFill (rowID, 3,temprow);
355: dataManualPage[2] := temprow[0];
356: dataManualPage[3] := temprow[1];
357: dataManualPage[4] := temprow[2];
358:
359: Card2StrOFill (colID, 2,tempcol);
360: dataManualPage[5] := tempcol[0];
361: dataManualPage[6] := tempcol[1];
362:
363: END ConvertToDataManualPage;
364:
365:
366: PROCEDURE DataCheck(matr        : CARDINAL;
367:                     matrName    : ARRAY OF CHAR;
368:                     row         : CARDINAL;
369:                     rowName     : ARRAY OF CHAR;
370:                     col         : CARDINAL;
371:                     colName     : ARRAY OF CHAR;
372:                     relationName: ARRAY OF CHAR;
373:                     dataKey     : ARRAY OF CHAR):BOOLEAN;
374: (**
```

```
375:        DataCheck - Check if data exists for a matrix/row/cell
376:
377:        P. M. McGuire  September 1986
378:
379:        Check if data exists in the relation specified in relationName,
380:        matrix, row, or cell using "dataKey" to read the relation.
381:        Return TRUE if data exists.  The relation is opened and closed
382:        in DataCheck with a no modify option.  The record is read on an
383:        alternate key with a EQ sage operation.  The dataKey can be one
384:        of three options:  UKeyM   - search on the matrix given only.
385:                           UKeyMR  - search on the matrix and row given.
386:                           UKeyMRC - search on the matrix and row and
387:                                     col given.
388:
389:        CALLING SEQUENCE -
390:
391:          DataCheck (matr, row, col, relationName, dataKey ): BOOLEAN
392:
393:        ENTRY -
394:
395:          matr         - CARDINAL
396:                         Matrix number
397:          matrName     - ARRAY OF CHAR
398:                         The field name for the matrix value.
399:          row          - CARDINAL
400:                         Row ID
401:          rowName      - ARRAY OF CHAR
402:                         The field name for the row value.
403:          col          - CARDINAL
404:                         Column ID
405:          colName      - ARRAY OF CHAR
406:                         The field name for the column value.
407:          relationName - ARRAY OF CHAR
408:                         The relation being checked
409:          dataKey      - ARRAY OF CHAR
410:                         Alternate key name in HEP.
411:                         It can be UKeyM, UKeyMR or UKeyMRC
412:
413:        EXIT -
414:
415:          BOOLEAN - TRUE = data exists, FALSE = no data
416:
417:        FILES -
418:
419:        RELATION    OPEN(T/F)   READ/  SAGE       FIELDS/     FIELD
420:        NAME        CLOSE       WRITE  OP         KEYS        CHANGES
421:        --------    ---------   -----  -----      -----       -------
422:
423:        relation-   O(F)/C      read   EQ         UKeyM
424:        Name                                      UKeyMR
425:                                                  UKeyMRC
426: *)
427:
428:
429: VAR
430:   relationOpen :BOOLEAN;
431:
432:
433: BEGIN
434:   relationOpen := MyOpenRelation(relationName, dataKey);
435:
436:   PutFieldC(relationName, matrName, matr);
437:   PutFieldC(relationName, rowName, row);
438:   PutFieldC(relationName, colName, col);
439:
440:   FindRecord(relationName, dataKey, EQ);
441:
442:   IF SageError = 0 THEN
443:      MyCloseRelation(relationName, relationOpen);
444:      RETURN TRUE;
445:   ELSE
446:      MyCloseRelation(relationName, relationOpen);
447:      RETURN FALSE;
448:   END; (* end if- SageError *)
449:
```

```
END DataCheck;


PROCEDURE DocCodes;

(**
        DocCodes - Retrieves "Original" and "Reference".

        P. M. McGuire  September 1986

        Expands Codes from "Referenc" to the description of
        "Original" and "Reference" and loads it into the
        Utility fields "Text" and "Text2" respectively for display.
        DocCodes gets the codes from the current "Referenc" record.

        CALLING SEQUENCE -

          DocCodes()

        ENTRY -

        EXIT -

        FILES -

        RELATION    OPEN(T/F)   READ/   SAGE        FIELDS/     FIELD
        NAME        CLOSE       WRITE   OP          KEYS        CHANGES
        --------    ---------   -----   -----       -----       -------

        Referenc    O(Prior)    read                Codes
        Utility                 write                           Text
                                                                Text2
*)

BEGIN
        (* check for original document *)

        IF GetFieldB("Referenc", "Codes[1]") THEN
           PutFieldA("Utility", "Text", "Original");
        ELSE
           ClearField("Utility", "Text");
        END;

        (* check for reference document *)

        IF GetFieldB("Referenc", "Codes[2]") THEN
           PutFieldA("Utility", "Text2", "Reference");
        ELSE
           ClearField("Utility", "Text2");
        END;

END DocCodes;


PROCEDURE DocType;

(**
        DocType - Expands the document type code into a description

        P. M. McGuire - September 1986

        Using the 2 letter document code from "Referenc" the
        description is loaded into the Utility "Text30" field.
        DocType gets docType of the current "Referenc" record.

        CALLING SEQUENCE -

          DocType()

        ENTRY -

        EXIT -

        FILES -
```

```
525:         RELATION    OPEN(./F)   READ/  DAGE        FIELDS/     FIELD
526:         NAME        CLOSE       WRITE  OP          KEYS        CHANGES
527:         --------    ---------   -----  -----       -----       -------
528:
529:         Referenc    O(Pr.or)    read               docType
530:         Utility                 write                          Text3J
531: *)
532:
533:
534: VAR
535:   codeText : ARRAY [0. 29] OF CHAR;
536:   docType  : ARRAY [0..1] OF CHAR;
537:
538: BEGIN
539:       GetFieldA("Referenc", "DocType", docType);
540:         IF (docTy.e[0] = "L") AND (docType[1] = "D") THEN    (* LD *)
541:            codeText := "Applicant/Licensee Document";
542:         ELSIF
543:            (docType[0] = "B") AND (docType'1] = "K") THEN    (* BK *)
544:            codeText := "Book";
545:         ELSIF
546:            (docType[0] = "C") AND (docType[1] = "P") THEN    (* CP *)
547:            codeText := "Conference Proceeding / Paper";
548:         ELSIF
549:            (docType[0] = "C") AND (docType[1] = "M") THEN    (* CM *)
550:            codeText := "Correspondence and Memoranda";
551:         ELSIF
552:            (docType[0] = "J") AND (docType[1] = "A") THEN    (* JA *)
553:            codeText := "Journal Article";
554:         ELSIF
555:            (docType[0] = "U") AND (docType[1] = "R") THEN    (* UR *)
556:             codeText := "Unpublished Report";
557:         ELSIF
558:            (docType[0] = "N") AND (docType[1] = "R") THEN    (* NR *)
559:             codeText := "NUREG Report";
560:         ELSIF
561:            (docType[0] = "O") AND (docType[1] = "P") THEN    (* OP *)
562:             codeText := "Other Federal Agency Pub.";
563:         ELSIF
564:            (docType[0] = "O") AND (docType[1] = "D") THEN    (* OD *)
565:             codeText := "General document";
566:         ELSE
567:             codeText := " Unknown ";
568:         END;
569:         PutFieldA("Utility", "Text30", codeText);
570:
571: END DocType;
572:
573:
574: PROCEDURE GetCellType (VAR typeOfCell : CellType;
575:                            rowID      : CARDINAL);
576:
577: (**
578:         GetCellType - Determination of cell type.
579:
580:         T. H. Tucker - March 1986.
581:
582:         GetCellType determines the cell type, either Functional
583:         Group Cell or an individual Cell.  Row numbers ending in
584:         0 are functional groups.
585:
586:         CALLING SEQUENCE -
587:
588:           GetCell[illegible] (typeOfCell, rowID)
589:
590:         ENTRY -
591:
592:           rowID          CARDINAL
593:                          [illegible] selected.
594:
595:         EXIT -
596:
597:           typeOfCell     [illegible]
598:                          [illegible] (GROUP, CELL)
599:                          [illegible] the cell selected.
```

```
600:
601:        FILES -
602: *)
603:
604: BEGIN
605:   IF (((rowID DIV 10) * 10) = rowID) THEN
606:     typeOfCell := FUNCTIONALGROUP;
607:   ELSE
608:     typeOfCell := CELL;
609:   END;
610:
611: END GetCellType;
612:
613:
614: PROCEDURE GetCellValidity (matrix, rowID, colID : CARDINAL;
615:                            VAR validity : ValidType);
616:
617: (**
618:        GetCellValidity - Cell validity check.
619:
620:        T. H. Tucker - March 1986.
621:
622:
623:        GetCellValidity determines if the cell requested by the
624:        user is valid. "CellVal" is opened and closed with a no modify
625:        option.  The record is read on the PrimeKey with an EQ sage
626:        operation.  The PrimeKey consisting of the matrix, rowID, and
627:        colID.
628:
629:        CALLING SEQUENCE -
630:
631:          GetCellValidity (matrix, rowID, colID, validity)
632:
633:        ENTRY -
634:
635:          matrix   - CARDINAL
636:                     The matrix number
637:          rowID    - CARDINAL
638:                     The row identification number
639:          colID    - CARDINAL
640:                     The column identification number
641:
642:        EXIT -
643:
644:          validity - ValidType
645:                     ValidType    = (VALID, INVALID, SHADED)
646:                     Returns the validity of the cell selected.
647:
648:        FILES -
649:
650:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
651:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
652:        --------    ---------   -----  -----       -----       -------
653:
654:        CellVal     O(F)/C      read   EQ          PrimeKey
655: *)
656:
657:
658: VAR
659:   i      :  CARDINAL;
660:   CVOpen :BOOLEAN;
661:
662:
663: BEGIN
664:
665:    CVOpen := MyOpenRelation ("CellVal", "PrimeKey");
666:    IF (SageError = 0) THEN
667:       validity := INVALID;
668:
669:       PutFieldC ("CellVal", "CellMatr", matrix);
670:       PutFieldC ("CellVal", "CellRow",  rowID);
671:       PutFieldC ("CellVal", "CellColu", colID);
672:
673:       ReadRecord ("CellVal", "PrimeKey", EQ);
674:
```

```
675:      IF (SageError = 0) THEN
676:        GetFieldC ("CellVal", "Validity", i);
677:        IF (i = 0) THEN
678:          validity := INVALID;
679:        ELSIF (i = 1) THEN
680:          validity := VALID;
681:        ELSIF (i = 2) THEN
682:          validity := SHADED;
683:        END;
684:      END;
685:
686:
687:      MyCloseRelation("CellVal", CVOpen);
688:    ELSE
689:      DisplayMessage ("** ERROR - could not open CellVal",TRUE);
690:    END;
691:
692: END GetCellValidity;
693:
694:
695: PROCEDURE GetMatrixDesc (matrix : CARDINAL);
696:
697: (**
698:        GetMatrixDesc -  Gets plant ID, job classification & taxonomy level
699:
700:        D. J. Fink - July 1986.
701:
702:        Opens, reads, and closes the relation MatrDesc to get the
703:        matrix's plant ID, job classification, and taxonomy level.
704:
705:
706:        CALLING SEQUENCE -
707:
708:          GetMatrixDesc (matrix)
709:
710:        ENTRY -
711:
712:          matrix - CARDINAL
713:                   The matrix number associated with the description.
714:
715:        EXIT -
716:
717:        FILES -
718:
719:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
720:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
721:        --------    ---------   -----  -----       -----       -------
722:
723:        MatrDesc    O(F)/C      read   EQ          Matrix
724: *)
725:
726:
727: VAR
728:   i : CARDINAL;
729:
730:
731: BEGIN
732:   OpenRelation("MatrDesc", NoModify);
733:
734:     PutFieldC   ("MatrDesc","Matrix", matrix);
735:     ReadRecord  ("MatrDesc", "Matrix", EQ);
736:
737:   CloseRelation("MatrDesc");
738:
739: END GetMatrixDesc;
740:
741:
742: PROCEDURE GetRowDescr (matrix, rowID: CARDINAL);
743:
744: (**
745:        GetRowDescr - Retrieval of row description.
746:
747:        T. H. Tucker - March 1986.
748:
749:        Retrieval of row description.  MATRROW and Rows are opened.
```

```
750:        Sage operation GE and EQ is used in reading the records.
751:
752:        CALLING SEQUENCE -
753:
754:          GetRowDescr ( matrix, rowID)
755:
756:        ENTRY -
757:
758:          matrix    - CARDINAL
759:                      The matrix number.
760:          rowID     - CARDINAL
761:                      The row number.
762:
763:        EXIT -
764:
765:        FILES -
766:
767:        RELATION    OPEN(T/F)   READ/  SAGE        FIELDS/     FIELD
768:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
769:        --------    ---------   -----  -----       -----       -------
770:
771:        MATRROW     O(F)        read   GE          PrimeKey
772:        Rows        O(F)/C      read   EQ          RowNumbr
773: *)
774:
775:
776: VAR
777:   rowNumbr:  CARDINAL;
778:   MATRROWOpen :BOOLEAN;
779:   RowsOpen    :BOOLEAN;
780:
781:
782: BEGIN
783:   MATRROWOpen := MyOpenRelation ("MATRROW", "PrimeKey");
784:   RowsOpen    := MyOpenRelation ("Rows", "RowNumbr");
785:
786:   ClearRelation("MATRROW");
787:   PutFieldC ("MATRROW", "Matrix", matrix);
788:   PutFieldC ("MATRROW", "RowID",  rowID);
789:   ReadRecord ("MATRROW", "PrimeKey", GE);
790:
791:   IF (SageError = 0) THEN
792:     CopyField ("MATRROW", "RowNumbr", "Rows", "RowNumbr");
793:     ReadRecord ("Rows", "RowNumbr", EQ);
794:   END;
795:
796:   MyCloseRelation ("Rows", RowsOpen);
797:   MyCloseRelation ("MATRROW", MATRROWOpen);
798:
799: END GetRowDescr;
800:
801:
802: PROCEDURE GetVerb ( matrix, colID : CARDINAL;
803:                     VAR sageerror : CARDINAL);
804: (**
805:        GetVerb - Retrieves Human Action Verb.
806:
807:        P. N. McGuire - September 1986.
808:
809:        GetVerb retrieves the human action verb.  Human action verb
810:        is dependent on the matrix and colID.  MATRCOL and Columns are
811:        opened and sage operation GE and EQ is used in reading the
812:        records.
813:
814:        CALLING SEQUENCE -
815:
816:          GetVerb (matrix, colID, sageerror)
817:
818:        ENTRY -
819:
820:          matrix    - CARDINAL
821:                      The matrix number.
822:          colID     - CARDINAL
823:                      The column number.
824:
```

```
825:        EXIT -
826:
827:          sageerror - CARDINAL
828:                      The sageerror from the MATRCOL read operation.
829:
830:
831:        FILES -
832:
833:        RELATION    OPEN(I/F)   READ/  SAGE        FIELDS/     FIELD
834:        NAME        CLOSE       WRITE  OP          KEYS        CHANGES
835:        --------    ---------   -----  ----        -----       -------
836:
837:        MATRCOL     O(F)/C      read   GE          PrimeKey
838:        Columns     O(F)/C      read   EQ          ColNumbr
839: *)
840:
841: BEGIN
842:  OpenRelation ("MATRCOL", NoModify);
843:   IF (SageError = 0) THEN
844:    OpenRelation ("Columns", NoModify);
845:     IF (SageError = 0) THEN
846.       ClearRelation("MATRCOL");
847:       PutFieldC ("MATRCOL", "Matrix", matrix);
848:       PutFieldC ("MATRCOL", "ColID",  colID);
849:       ReadRecord ("MATRCOL", "PrimeKey", GE);
850:       sageerror := SageError;
851:
852:       IF (SageError = 0) AND
853:          (CompareFieldC ("MATRCOL", "Matrix", EQ, matrix)) THEN
854:           CopyField ("MATRCOL", "ColNumbr", "Columns", "ColNumbr");
855:           ReadRecord ("Columns", "ColNumbr", EQ);
856:       ELSE
857:         sageerror := 200;
858:       END;     (* IF-SageError *)
859:
860:    ELSE
861:     DisplayMessage("* ERROR - could not open Columns", RingBell);
862:    END;
863:   ELSE
864:    DisplayMessage("* ERROR - could not open MATRCOL", RingBell);
865:  END;
866:
867:  CloseRelation("MATRCOL");
868:  CloseRelation("Columns");
869:
870: END GetVerb;
871:
872:
873: PROCEDURE  MyCloseRelation (relation :ARRAY OF CHAR;
874:                             status   :BOOLEAN);
875:
876: (**
877:    MyCloseRelation - Closes relation or data file as necessary
878:
879:    G. H. Beers - December 1987
880:
881:    Checks the initial status of a relation and returns it to its initial state.
882:
883:    CALLING SEQUENCE -
884:
885:      MyCloseRelation (relation,status)
886:
887:    ENTRY -
888:
889:      relation - ARRAY OF CHAR
890:                 Name of relation to be opened or checked.
891:
892:      status   - BOOLEAN
893:                 initial relation status flag
894:                 TRUE    Relation or index file was initially open
895:                 FALSE - Relation was initially closed
896:
897:    EXIT -
898:
899:    FILES -
```

```
900: *)
901:
902: BEGIN
903:   IF status THEN
904:     CloseRelationFiles (relation);
905:   ELSE
906:     CloseRelation (relation);
907:   END;
908:
909: END MyCloseRelation;
910:
911:
912: PROCEDURE  MyOpenRelation (relation, key :ARRAY OF CHAR) :BOOLEAN;
913:
914: (**
915:     MyOpenRelation - Opens relation if necessary and returns initial status
916:
917:     G. H. Beers - December 1987
918:
919:     Checks the current status of a relation and opens it if necessary.
920:
921:     CALLING SEQUENCE -
922:
923:       MyOpenRelation (relation,key)
924:
925:     ENTRY -
926:
927:       relation - ARRAY OF CHAR
928:                  Name of relation to be opened or checked.
929:
930:       key      - ARRAY OF CHAR
931:                  Name of key to use for testing relation current status
932:
933:     EXIT  -
934:
935:       result   - BOOLEAN
936:                  Initial relation status flag
937:                  TRUE - Relation or index file was initially open
938:
939:                  FALSE - Relation was closed
940:
941:      FILES -
942: *)
943:
944:
945: VAR
946:   result  :BOOLEAN;
947:
948:
949: BEGIN
950:   result := TRUE;
951:   FindRecord (relation, key, Next);
952:   IF SageError = 106  THEN
953:     OpenRelation (relation, FALSE);
954:     result := FALSE
955:   END;
956:   RETURN result;
957:
958: END MyOpenRelation;
959:
960:
961: PROCEDURE OutputFile (name : ARRAY OF CHAR;
962:                       VAR ok : BOOLEAN;
963:                       dataManual!Page: ARRAY OF CHAR;
964:                       VAR screenDisplay: BOOLEAN);
965:
966: (**
967:         OutputFile - Opens output file.
968:
969:         P. M. McGuire - September 1986.
970:
971:         Opens the WrForm file for output.
972:
973:         CALLING SEQUENCE -
974:
```

```
975:           OutputFile(name, ok, dataManualPage, screenDisplay)
976:
977:         ENTRY -
978:
979:           name            - ARRAY OF CHAR
980:                             Title of print menu
981:
982:           dataManualPage  - ARRAY OF CHAR
983:                             The dataManualPage number for display.
984:
985:         EXIT -
986:
987:           ok              - BOOLEAN
988:                             Success of the opening operation
989:           screenDisplay   - BOOLEAN
990:                             Display to the screen = True
991:                             Write to a file or PRN = False
992:
993:         FILES -
994:
995:
996:         FILE        OPEN        DEVICE              FORMS
997:         NAME        CLOSE                           WRITTEN   DESCRIPTION
998:         -----       -----       -------             -------   -----------
999:
1000:        ftbl        0           filename
1001: *)
1002:
1003:
1004: VAR
1005:   select : CHAR;
1006:   filename: ARRAY [0..9] OF CHAR;
1007:   state: FileState;
1008:
1009:
1010: BEGIN
1011:
1012: LOOP
1013:  PutFieldA ("Utility", "DataMan", dataManualPage);
1014:  PutFieldA ("Utility", "Text2", name);
1015:  PutFieldA ("Utility", "CommandA", "S");
1016:  PutFieldA ("Utility", "Text25", "Cell Page Number        : ");
1017:
1018:  ClearField("Utility", "Text");
1019:
1020:  ClearStack ();
1021:  DisplayForm ("Menun1", "Utility", "Text", FALSE);
1022:
1023:  GetFieldA ("Utility", "CommandA", select);
1024:  GetFieldA ("Utility", "Text", filename );
1025:
1026:  CASE select OF
1027:   "S" :  IF (filename[0] = "C") AND (filename[1] = "O") AND
1028:             (filename[2] = "N") AND (filename[3] = " ") THEN
1029:
1030:             screenDisplay := TRUE;
1031:          ELSIF (filename[0] = " ") THEN
1032:             filename := "CON";
1033:             screenDisplay := TRUE;
1034:          ELSE
1035:             screenDisplay := FALSE;
1036:          END;
1037:
1038:          OpenReport (ftbl, filename, state);
1039:          ok := TRUE;
1040:          EXIT|
1041:   "E" : ok := FALSE;
1042:         EXIT
1043:  ELSE
1044:    DisplayMessage("Invalid selection", TRUE);
1045:  END; (* if- select *)
1046:
1047: END;
1048:
1049: END OutputFile;
```

```
1050:
1051:
1052: PROCEDURE SearchDoc;
1053:
1054: (**
1055:         SearchDoc - Search of source statements by document.
1056:
1057:         P. M. McGuire - September 1986.
1058:
1059:         Generation of document listing by source statement.
1060:
1061:         CALLING SEQUENCE -
1062:
1063:           SearchDoc()
1064:
1065:         ENTRY -
1066:
1067:         EXIT -
1068:
1069:         FILES -
1070:
1071:         RELATION   OPEN(T/F)  READ/   SAGE      FIELDS/          FIELD
1072:         NAME       CLOSE      WRITE   OP        KEYS             CHANGES
1073:         --------   ---------  -----   -----     -----            -------
1074:
1075:         CellData   O(F)/C     read    EQ        DocKey
1076:                                                 RefKey
1077:
1078:         Referenc   O(prior)   read              DocNumbr
1079:
1080:         FILE       OPEN       DEVICE  FORMS
1081:         NAME       CLOSE              WRITTEN   DESCRIPTION
1082:         -----      -----      ------- -------   -----------
1083:
1084:         ftbl       O/C        PRN     RepDoc1   doc. description
1085:                                       RepDoc2   header for
1086:                                                 Celldata
1087:                                       RepDoc3
1088: *)
1089:
1090:
1091: VAR
1092:   fileName: ARRAY [0..9] OF CHAR;
1093:   select: CHAR;
1094:   state: FileState;
1095:   error: CARDINAL;
1096:   newPage: BOOLEAN;
1097:
1098:
1099: BEGIN
1100:   LOOP                (* Open printer file (device) *)
1101:     ClearField("Utility", "DataMan");
1102:     ClearField("Utility", "Text25");
1103:     PutFieldA("Utility", "CommandA", "S");
1104:     PutFieldA ("Utility", "Text2", "Search Doc.");
1105:     ClearField("Utility", "Text");
1106:
1107:     ClearStack ();
1108:     DisplayForm ("MenuN1", "Utility", "Text", FALSE);
1109:
1110:     GetFieldA ("Utility", "CommandA", select);
1111:     GetFieldA("Utility", "Text", fileName);
1112:
1113:     IF select = "S" THEN            (* start *)
1114:           EXIT;
1115:     ELSIF select = 'E' THEN         (* exit *)
1116:         RETURN;
1117:     ELSE
1118:      DisplayMessage (" Invalid Selection ", TRUE);
1119:     END;
1120:
1121:   END; (* LOOP *)
1122:
1123:   IF (fileName[0] = " ") THEN
1124:       fileName := "CON";
```

```
1125:    END;
1126:    OpenReport(ftbl, fileName, state);
1127:
1128:    DefineHeader(ftbl,"Search Report", 25);
1129:    WrForm(ftbl,"RepDoc1"," "," ",TRUE,error);  (* document description *)
1130:    WrForm(ftbl,"RepDoc2"," "," ",TRUE,error);  (* headers for CellData *)
1131:
1132:    (* output original document references *)
1133:
1134:    OpenRelation("CellData",NoModify);
1135:    ClearRelation("CellData");
1136:    CopyField("Referenc","DocNumbr","CellData","DocKey");
1137:    ReadRecord("CellData","DocKey",EQ);
1138:    IF SageError = 0 THEN
1139:       WrString(ftbl,"                ");  (* blank line *)
1140:       WrLn(ftbl,newPage);
1141:
1142:       WrString(ftbl," < ORIGINAL >");
1143:       WrLn(ftbl,newPage);
1144:
1145:       WrString(ftbl,"                ");  (* blank line *)
1146:       WrLn(ftbl,newPage);
1147:
1148:    END;
1149:    WHILE SageError = 0 DO
1150:       WrForm(ftbl,"RepDoc3","RepDoc2"," ",TRUE,error);
1151:       ReadRecord("CellData","DocKey",NextEQ);
1152:    END; (* while for original data *)
1153:
1154:    (* output referenced document *)
1155:
1156:    CopyField("Referenc","DocNumbr","CellData","RefKey");
1157:    ReadRecord("CellData","RefKey",EQ);
1158:    IF SageError = 0 THEN
1159:
1160:       WrLn(ftbl,newPage);                (* blank line *)
1161:
1162:       WrString(ftbl," < REFERENCE >");
1163:       WrLn(ftbl,newPage);
1164:
1165:       WrString(ftbl,"                ");
1166:       WrLn(ftbl,newPage);
1167:    END;
1168:    WHILE SageError = 0 DO
1169:       WrForm(ftbl,"RepDoc4","RepDoc2"," ",TRUE,error);
1170:       ReadRecord("CellData","RefKey",NextEQ);
1171:    END; (* while for original data *)
1172:
1173:    CloseRelation("CellData");
1174:    CloseReport(ftbl,state);
1175:
1176: END SearchDoc;
1177:
1178: END NUCGEN.
```

```
MODULE HEPAgg;

(**
        HEPAgg - Aggregation access to storage buffer

        Provides for aggregation on Human Error Probability (HEP)
        records referenced in the storage buffer.
*)


                IMPORT (* module     *)
                 Program
                ;

FROM AsCII      IMPORT (* constant   *)
                 nul
                ;

FROM Binary     IMPORT (* procedures *)
                 ReadBytes ,WriteBytes
                ;

FROM BReports   IMPORT (* procedure  *)
                 CreateBufferReport
                ;

FROM Calc       IMPORT (* type       *)
                 HEPType
                ,

                       (* variable   *)
                case3UserAbort
                ,

                       (* procedures *)
                 CalcCellorFuncGrpHEP ,CalcTaskHEP
                ;

FROM DiskLib    IMPORT (* procedure  *)
                 Lookup
                ;

FROM Files      IMPORT (* types      *)
                 FileState ,File
                ,

                       (* procedures *)
                 Reset ,Remove ,Close ,Rewrite
                ;

FROM MoveLib    IMPORT (* procedure  *)
                 FillChar
                ;

FROM NUCGEN     IMPORT (* variable   *)
                 ftbl
                ,

                       (* procedures *)
                 Clearinghouse ,ClearStack ,ConvertToDataManualPage
                ;

FROM NUCPRINT   IMPORT (* constant   *)
                 recordsLimit
                ,

                       (* variables  *)
                 recordsLocatedTotal
                ,

                       (* procedure  *)
                 DisplaySource
                ;
```

```
75: FROM Reports   IMPORT (* procedures *)
76:                 OpenReport, CloseReport, DefineReport, TopOfPage, WrForm
77:                ;
78:
79: FROM Retrieve  IMPORT dataManualPage ,bufferAggregated ,bufferHEP
80:                ,bufferUCB ,bufferLCB
81:                ;
82:
83: FROM Sage      IMPORT (* type       *)
84:                 SageOperations
85:                ,
86:
87:                       (* variable   *)
88:                 SageError
89:                ,
90:
91:                       (* procedures *)
92:                 OpenDataBase ,CloseDataBase
93:                ,OpenRelation ,CloseRelation ,CloseRelationFiles
94:                ,ClearField   ,ClearRelation
95:                ,DisplayMessage
96:                ,DisplayForm
97:                ,GetFieldA ,GetFieldC ,GetFieldF
98:                ,PutFieldA ,PutFieldC ,PutFieldF
99:                ,ReadRecord   ,ReadRecordC   ,ReadRecordR
100:               ,FindRecord
101:               ;
102:
103: FROM SortLib   IMPORT (* procedure  *)
104:                 CompareKey
105:                ;
106:
107: FROM StorageM  IMPORT (* variable   *)
108:                 describeBuffer
109:                ,
110:
111:                       (* procedures *)
112:                 ShowBufferDescription ,DecodeCellDataKey
113:                ;
114:
115: FROM String    IMPORT (* procedure  *)
116:                 Concat
117:                ;
118:
119: FROM SYSTEM    IMPORT (* functions  *)
120:                 ADR ,SIZE
121:                ;
122:
123: FROM SYSTEMToBeImpl
124:                IMPORT (* type       *)
125:                 BYTECOUNT
126:                ;
127:
128: FROM Terminal  IMPORT (* procedure  *)
129:                 CondRead
130:                ;
131:
132: FROM ThorPort  IMPORT (* procedures *)
133:                 ClearScreen, Pause
134:                ;
135:
136:
137: CONST
138:   NoModify  = FALSE;
139:   NoBell    = FALSE;
140:   RingBell  = TRUE;
141:   NoDefault = FALSE;
142:
143: VAR
144:   index : CARDINAL;
145:   state : FileState;
146:   input : CHAR;
147:
148:
149: PROCEDURE AggregateHEP
```

```
150:          ()          ;
151: (**
152:        AggregateHEP - Calculate and display aggregated HEP
153:
154:        O. J. Call - April 1987
155:
156:        Calculates an aggregated HEP for located data.  First, an
157:        aggregation is done for each task represented in the search
158:        buffer.  Then an aggregation is done using all the task HEP
159:        values just calculated/identified.  The final result is
160:        displayed on the screen.  The user may select to generate a
161:        report of this aggregation process.
162:
163:
164:        CALLING SEQUENCE -
165:
166:           AggregateHEP ()
167:
168:
169:        ENTRY -
170:
171:        EXIT -
172:
173:        FILES -
174:
175:        FILE        OPEN        DEVICE              FORMS
176:        NAME        CLOSE                           WRITTEN   DESCRIPTION
177:        ****        *****       *******             *******   ***********
178:        ftbl        O/C         fileName
179:
180: *)
181:
182:
183: CONST
184:   methodRecord = "              Task (a single record)              ";
185:   methodTask   = "        Task (Maximum Likelihood Estimate)        ";
186:   methodCell   = "     Cell (Median based on Geometric Average)     ";
187:   methodFun    = "  Functional (Median based on Geometric Average)  ";
188:
189:   noDataMess    = "Can not access one of the located data records";
190:   case3StopMess = "Aggregation of located records stopped as requested";
191:   noReportMess  =
192:     "Buffer aggregation completed, no report was requested";
193:
194: VAR
195:   fileName     : ARRAY [0..11] OF CHAR;
196:   printer      : ARRAY [0..11] OF CHAR;
197:   console      : ARRAY [0..11] OF CHAR;
198:   blankName    : ARRAY [0..11] OF CHAR;
199:   completeMess : ARRAY [0..79] OF CHAR;
200:   method       : ARRAY [0..49] OF CHAR; (* aggregation technique *)
201:
202:   sourceHEPData : ARRAY [0..(recordsLimit - 1)] OF HEPType;
203:   taskHEPData   : ARRAY [0..(recordsLimit - 1)] OF HEPType;
204:
205:   success : BOOLEAN;
206:
207:   case    : CARDINAL;
208:   index   : CARDINAL;
209:   iTask   : CARDINAL;
210:   iSource : CARDINAL;
211:   iStart  : CARDINAL;
212:   iEnd    : CARDINAL;
213:
214:   (* current              previous              initial        *)
215:   matrix : CARDINAL;  matrix0 : CARDINAL;  matrix1 : CARDINAL;
216:   row    : CARDINAL;  row0    : CARDINAL;  row1    : CARDINAL;
217:   column : CARDINAL;  column0 : CARDINAL;  column1 : CARDINAL;
218:   task   : CARDINAL;  task0   : CARDINAL;
219:   source : CARDINAL;  source0 : CARDINAL;
220:
221:
222: BEGIN (* AggregateHEP *)
223:   bufferAggregated := FALSE;
224:
```

```
225:   IF  recordsLocatedTotal = 0  THEN
226:     DisplayMessage ("No data in buffer, therefore no aggregation"
227:                                                                  ,RingBell);
228:     RETURN;
229:   END; (* IF *)
230:
231:   ShowBufferDescription ();
232:   Pause (1000);
233:
234:   OpenRelation ("CellData",NoModify);
235:   IF  SageError <> 0  THEN
236:     DisplayMessage ("Failed opening cell data for HEP aggregation"
237:                                                                  ,RingBell);
238:     RETURN;
239:   END;
240:
241:   ClearScreen;
242:   DisplayMessage ("Aggregation in process - please standby ...",NoBell);
243:
244:                                        (* initialize HEP data arrays *)
245:   FOR  index := 0 TO (recordsLimit - 1)  DO
246:     sourceHEPData[index].hep := 0.0;
247:     sourceHEPData[index].ucb := 0.0;
248:     sourceHEPData[index].lcb := 0.0;
249:     sourceHEPData[index].E   := 0.0;
250:     sourceHEPData[index].N   := 0.0;
251:     sourceHEPData[index].med := 0.0;
252:     sourceHEPData[index].erf       := 0;
253:     sourceHEPData[index].sourceNumb := 0;
254:
255:     taskHEPData[index].hep := 0.0;
256:     taskHEPData[index].ucb := 0.0;
257:     taskHEPData[index].lcb := 0.0;
258:     taskHEPData[index].E   := 0.0;
259:     taskHEPData[index].N   := 0.0;
260:     taskHEPData[index].med := 0.0;
261:     taskHEPData[index].erf        := 0;
262:     taskHEPData[index].sourceNumb := 0;
263:   END; (* FOR *)
264:
265:                                  (* get initial source HEP data values *)
266:   GetCellDataRecord (
267:     0,matrix0,row0,column0,task0,source0,sourceHEPData,success);
268:   IF  NOT success  THEN
269:     DisplayMessage (noDataMess,RingBell);
270:     RETURN;
271:   END;
272:
273:                               (* set initial matrix, row, column values *)
274:   matrix1 := matrix0;
275:   row1    := row0;
276:   column1 := column0;
277:
278:                        (* get any remaining values into HEP data arrays *)
279:   IF  (recordsLocatedTotal = 1)  THEN  (* set aggregated HEP *)
280:     bufferHEP := sourceHEPData[0].med;
281:     bufferUCB := sourceHEPData[0].ucb;
282:     bufferLCB := sourceHEPData[0].lcb;
283:     method := methodRecord;
284:     CloseRelation ("CellData");
285:
286:   ELSE  (* get remaining source HEP data *)
287:     iStart := 0;
288:     iEnd   := iStart;
289:     iTask  := iStart;
290:     FOR  index := 1 TO (recordsLocatedTotal - 1) BY 1  DO
291:       GetCellDataRecord (
292:         index,matrix,row,column,task,source,sourceHEPData,success);
293:       IF  NOT success  THEN
294:         DisplayMessage (noDataMess,RingBell);
295:         RETURN;
296:       END; (* success IF *)
297:
298:       IF  ((matrix0 = matrix) AND (row0 = row) AND (column0 = column)
299:                                          AND (task0 = task))  THEN
```

```
300:        INC (iEnd);                                          (* same task *)
301:        IF (index >= (recordsLocatedTotal-1)) THEN
302:                                  (* load final task HEP data values *)
303:          ConvertToDataManualPage (dataManualPage,matrix0,row0,column0);
304:          PutFieldA ("Utility","DataMan",dataManualPage);
305:          PutFieldC ("Utility","task[1]",task0          );
306:          CalculateForTaskInBuffer (iStart,iEnd,sourceHEPData,iTask,
307:                                    taskHEPData);
308:          IF  case3UserAbort  THEN
309:            CloseRelation ("CellData");
310:            DisplayMessage (case3StopMess,NoBell);
311:            RETURN;
312:          END; (* case3UserAbort IF *)
313:
314:          INC (iEnd);
315:          iStart := iEnd;
316:          INC (iTask);
317:        END;  (* index IF *)
318:
319:      ELSE  (* a new task found *)
320:        IF  (iEnd = iStart)  THEN
321:                            (* only one source for previous task *)
322:          iSource := index - 1;
323:          taskHEPData[iTask].mod := sourceHEPData[iSource].mod;
324:          taskHEPData[iTask].ucb := sourceHEPData[iSource].ucb;
325:          taskHEPData[iTask].lcb := sourceHEPData[iSource].lcb;
326:        ELSE  (* load previous task HEP data values *)
327:          ConvertToDataManualPage (dataManualPage,matrix0,row0,column0);
328:          PutFieldA ("Utility","DataMan",dataManualPage);
329:          PutFieldC ("Utility","task[1]",task0          );
330:          CalculateForTaskInBuffer (iStart,iEnd,sourceHEPData,
331:                                    iTask,taskHEPData);
332:          IF  case3UserAbort  THEN
333:            CloseRelation ("CellData");
334:            DisplayMessage (case3StopMess,NoBell);
335:            RETURN;
336:          END; (* case3UserAbort IF *)
337:
338:        END; (* iEnd IF *)
339:
340:        INC (iEnd);
341:        iStart := iEnd;
342:        INC (iTask);
343:
344:      END; (* matrix,row,column IF *)
345:
346:              (* current becomes previous *)
347:      matrix0 := matrix;
348:      row0    := row;
349:      column0 := column;
350:      task0   := task;
351:      source0 := source;
352:    END; (* FOR *)
353:
354:    CloseRelation ("CellData");
355:
356:                                  (* set aggregation technique label *)
357:    IF  iTask = 1  THEN
358:      method := methodTask;
359:    ELSIF  (matrix = matrix1) AND
360:           (row    = row1   ) AND
361:           (column = column1)      THEN
362:      method := methodCell;
363:    ELSE
364:      method := methodFun;
365:    END; (* iTask IF *)
366:
367:             (* call 4010 Procedure A10 to calculate aggregated HEP *)
368:    case := 0;
369:    CalcCellOrFuncGrpHEP (
370:      case,iTask,taskHEPData,bufferHEP,bufferUCB,bufferLCB);
371:
372:  END; (* recordsLocatedTotal IF *)
373:
374:  bufferAggregated := TRUE;
```

```
375:                                        (* finish set-up for form display *)
376:   PutFieldA ("UtilAhoc","Describe",describeBuffer);
377:   PutFieldA ("UtilAhoc","Text50"  ,method        );
378:   PutFieldF ("UtilAhoc","HEP"     ,bufferHEP     );
379:   PutFieldF ("UtilAhoc","UCB"     ,bufferUCB     );
380:   PutFieldF ("UtilAhoc","LCB"     ,bufferLCB     );
381:
382:   ClearField ("UtilAhoc","CommandA");
383:   DisplayMessage (" ",NoBell);
384:
385:   LOOP                                      (* select to report or not *)
386:     ClearStack ();
387:     DisplayForm ("Menur43","UtilAhoc","CommandA",NoDefault);
388:     GetFieldA ("UtilAhoc","CommandA",input);
389:
390:     CASE  input  OF
391:       "?" : Clearinghouse;
392:             PutFieldA ("UtilAhoc","CommandA"," ")|
393:
394:       "E" : DisplayMessage (noReportMess,NoBell);
395:             RETURN|
396:
397:       "R" : EXIT;
398:     ELSE
399:       DisplayMessage ("Invalid Selection",RingBell);
400:     END; (* CASE *)
401:
402:   END; (* LOOP *)
403:
404:                                             (* set-up for a report *)
405:   printer   := "PRN         ";
406:   console   := "CON         ";
407:   blankName := "            ";
408:
409:   PutFieldA ("Utility","CommandA","S");
410:   PutFieldA ("Utility","Text40","    Aggregated HEP for Located Data");
411:   ClearField ("Utility","Text2");
412:
413:   LOOP                                      (* select to start report or not *)
414:     ClearStack ();
415:     DisplayForm ("Menur41","Utility","Text2",NoDefault);
416:     GetFieldA ("Utility","CommandA",input);
417:
418:     CASE  input  OF
419:       "?" : Clearinghouse ();
420:             PutFieldA ("Utility","CommandA"," ")|
421:
422:       "E" : DisplayMessage (noReportMess,NoBell);
423:             RETURN|
424:
425:       "S" : GetFieldA ("Utility","Text2",fileName);
426:             IF  (fileName[0] = nul) OR
427:                 (CompareKey (fileName,blankName,12) = 0)  THEN
428:               DisplayMessage (
429:                 "A valid DOS file name is needed for this option"
430:                                                              ,RingBell);
431:             ELSE
432:               EXIT;
433:             END; (* IF *)
434:     ELSE
435:       DisplayMessage ("Invalid Selection",RingBell);
436:     END; (* CASE *)
437:
438:   END; (* LOOP *)
439:
440:                                             (* generate buffer report *)
441:   OpenReport (ftbl,fileName,state);
442:   IF  state <> ok  THEN
443:     DisplayMessage ("Error trying to open report file",RingBell);
444:   ELSE
445:     CreateBufferReport (fileName,console,TRUE);
446:     IF  ((CompareKey (fileName,printer  ,12) <> 0) AND
447:          (CompareKey (fileName,console  ,12) <> 0) AND
448:          (CompareKey (fileName,blankName,12) <> 0) AND
449:          (fileName[0] <> nul)   )  THEN
```

```
450:       Concat ("Report with aggregate HEP written to file ",
451:               fileName,completeMess,success);
452:       DisplayMessage (completeMess,NoBell);
453:     ELSIF  (CompareKey (fileName,printer,12) = 0)  THEN
454:       DisplayMessage (
455:         "Report with aggregate HEP sent to printer",NoBell);
456:     END; (* CompareKey IF *)
457:   END; (* state IF *)
458:   CloseReport(ftbl,state);
459:
460: END AggregateHEP;
461:
462:
463: PROCEDURE CalculateForTaskInBuffer
464:           (    start       : CARDINAL
465:           :    end         : CARDINAL
466:           ;VAR sourceArray : ARRAY OF HEPType
467:           ;    iTask       : CARDINAL
468:           ;VAR taskArray   : ARRAY OF HEPType);
469:
470: (**
471:         CalculateForTaskInBuffer - Calculate task HEP on buffer records
472:
473:         O. J. Call - May 1987
474:
475:         Calculates task HEP for search buffer source statement records.
476:         Procedure A9 in 4010 is used for calculation.  Source HEPs are
477:         the input for this procedure.
478:
479:
480:         CALLING SEQUENCE -
481:
482:            CalculateForTaskInBuffer (
483:                                start,end,sourceArray,iTask,taskArray)
484:
485:
486:         ENTRY -
487:
488:            start - CARDINAL
489:                    Starting position, in sourceArray, to use for
490:                    calculation.
491:
492:            end   - CARDINAL
493:                    Ending position, in sourceArray, to use for
494:                    calculation.
495:
496:            sourceArray - ARRAY OF HEPType
497:                    Array of source statement HEP data.
498:
499:            iTask - CARDINAL
500:                    Pointer to next available entry position in
501:                    taskArray.
502:
503:         EXIT -
504:
505:            taskArray - ARRAY OF HEPType
506:                    Array of task statement HEP data.
507:
508:         FILES -
509: *)
510:
511:
512: VAR
513:   case       :CARDINAL;
514:   Med        :REAL;
515:   i          :CARDINAL;
516:   Ucb        :REAL;
517:   Lcb        :REAL;
518:   size       :CARDINAL;
519:   sourceData :ARRAY [0..recordsLimit-1] OF HEPType;
520:   sourceUsed :ARRAY [0..recordsLimit-1] OF BOOLEAN;
521:   standAlone :BOOLEAN;
522:
523:
524: BEGIN (* CalculateForTaskInBuffer *)
```

```
525:   FOR  i := start TO end  DO
526:     sourceData[i-start] := sourceArray[i];
527:   END; (* FOR *)
528:
529:   size       := end - start + 1;
530:   standAlone := FALSE;
531:   case       := 0;
532:                                              (* call 4010 Procedure A9 *)
533:
534:   CalcTaskHEP (Med,Ucb,Lcb,size,case,sourceData,standAlone,sourceUsed);
535:
536:   taskArray[iTask].med := Med;
537:   taskArray[iTask].ucb := Ucb;
538:   taskArray[iTask].lcb := Lcb;
539:
540: END CalculateForTaskInBuffer;
541:
542:
543: PROCEDURE GetCellDataRecord
544:           (    position  : CARDINAL
545:           ;VAR matrix    : CARDINAL
546:           ;VAR row       : CARDINAL
547:           ;VAR column    : CARDINAL
548:           ;VAR task      : CARDINAL
549:           ;VAR source    : CARDINAL
550:           ;VAR dataArray : ARRAY OF HEPType
551:           ;VAR success   : BOOLEAN         );
552:
553: (**
554:         GetCellDataRecord - Get CellData record HEP data into an array
555:
556:         O. J. Call - June 1987
557:
558:         A CellData record is read into memory after determining a
559:         key-value identified in the ad hoc buffer.  HEP data from the
560:         current CellData record is moved into dataArray.
561:
562:
563:         CALLING SEQUENCE -
564:
565:            GetCellDataRecord (position,
566:              matrix,row,column,task,source,dataArray,success);
567:
568:
569:         ENTRY -
570:
571:            position  - CARDINAL
572:                        Index of position within bufferAdHoc array.
573:
574:            matrix    - CARDINAL
575:                        Numeric matrix identifier.
576:
577:            row       - CARDINAL
578:                        Numeric row identifier for identified matrix.
579:
580:            column    - CARDINAL
581:                        Numeric column identifier for identified matrix.
582:
583:            task      - CARDINAL
584:                        Numeric task statement identifier.
585:
586:            source    - CARDINAL
587:                        Numeric source identifier.
588:
589:         EXIT -
590:
591:            matrix    - CARDINAL
592:                        Numeric matrix identifier.
593:
594:            row       - CARDINAL
595:                        Numeric row identifier for identified matrix.
596:
597:            column    - CARDINAL
598:                        Numeric column identifier for identified matrix.
599:
```

```
600:            task      - CARDINAL
601:                        Numeric task statement identifier.
602:
603:            source    - CARDINAL
604:                        Numeric source identifier.
605:
606:            dataArray - ARRAY OF HEPType
607:                        Array for source HEP data values located.  Index
608:                        positions in this array match up with positions
609:                        in the ad hoc buffer.
610:
611:            success   - BOOLEAN
612:                        TRUE  denotes success in reading cell data.
613:                        FALSE denotes failure in reading cell data.
614:
615:          FILES -
616: *)
617:
618: BEGIN (* GetCellDataRecord *)
619:   IF  NOT DecodeCellDataKey (position + 1,matrix,row,column,task,source)
620:                                                                      THEN
621:     CloseRelation ("CellData");
622:     success := FALSE;
623:     RETURN;
624:   END; (* DecodeCellDataKey *)
625:
626:   ReadCellDataRecord (matrix,row,column,task,source);
627:   IF  SageError <> 0  THEN
628:     DisplayMessage ("Failed in reading cell data for HEP aggregation"
629:                                                              ,RingBell);
630:     CloseRelation ("CellData");
631:     success := FALSE;
632:     RETURN;
633:   END; (*SageError IF *)
634:   success := TRUE;
635:
636:   GetFieldF ("CellData","HEP"     ,dataArray[position].hep       );
637:   GetFieldF ("CellData","UCB"     ,dataArray[position].ucb       );
638:   GetFieldF ("CellData","LCB"     ,dataArray[position].lcb       );
639:   GetFieldF ("CellData","E"       ,dataArray[position].E         );
640:   GetFieldF ("CellData","N"       ,dataArray[position].N         );
641:   GetFieldF ("CellData","Median"  ,dataArray[position].med       );
642:   GetFieldF ("CellData","Erf"     ,dataArray[position].erf       );
643:   GetFieldC ("CellData","CellNumb",dataArray[position].sourceNumb);
644:
645: END GetCellDataRecord;
646:
647:
648: PROCEDURE ReadCellDataRecord
649:           (matrix : CARDINAL
650:           ;row    : CARDINAL
651:           ;column : CARDINAL
652:           ;task   : CARDINAL
653:           ;source : CARDINAL);
654:
655: (**
656:         ReadCellDataRecord - Reads a CellData record into memory
657:
658:         O. J. Call - June 1987
659:
660:         Reads a CellData record into memory after specifying the
661:         relation key-value.
662:
663:
664:         CALLING SEQUENCE -
665:
666:            ReadCellDataRecord (matrix,row,column,task,source)
667:
668:
669:         ENTRY -
670:
671:            matrix    - CARDINAL
672:                        Numeric matrix identifier.
673:
674:            row       - CARDINAL
```

```
675:                        Numeric row identifier for identified matrix.
676:
677:            column   - CARDINAL
678:                        Numeric column identifier for identified matrix.
679:
680:            task     - CARDINAL
681:                        Numeric task statement identifier.
682:
683:            source   - CARDINAL
684:                        Numeric source identifier.
685:
686:         EXIT -
687:
688:         FILES -
689: *)
690:
691: BEGIN (* readCellDataRecord *)
692:   PutFieldC ("CellData","CellMatr",matrix);
693:   PutFieldC ("CellData","CellRow" ,row   );
694:   PutFieldC ("CellData","CellColu",column);
695:   PutFieldC ("CellData","CellTask",task  );
696:   PutFieldC ("CellData","CellNumb",source);
697:
698:   ReadRecord ("CellData","CellKey",EQ);
699:
700: END ReadCellDataRecord;
701:
702:
703: BEGIN (* HEPAgg module *)
704:
705:   AggregateHEP ();
706:
707: END HEPAgg.
```

APPENDIX H

SAMPLE SOURCE LISTING FOR HCF DATA PROCESSING

APPENDIX H

SAMPLE SOURCE LISTING FOR HCF DATA PROCESSING

This appendix presents a sample of some source listings for a library and a module used for HCF data processing. The DOCUPROC utility, which was used to generate these listings, provides line numbering, a table of contents, and an index. The line numbering is restarted for each module-level code element.

Table of Contents

```
1: DEFINITION MODULE General;
2:
3: EXPORT QUALIFIED (* procedures *)
4:         ClearUEvent          ,ConvertMilitaryDate ,DisplaySageError
5:        ,GetCatName           ,GetFailureGroups    ,MilitaryDate
6:        ,MyCloseRelation      ,ShowSageError
7:        ,ClearStack           ,ClearMessage        ,
8:
9:                 (* functions  *)
10:        Among                ,BlankData           ,CheckRelationStatus
11:       ,FailureGroupMember   ,MyOpenRelation      ,PlacesC
12:       ;
13:
14:
15:
16: PROCEDURE Among (VAR f,l :ARRAY OF CHAR;
17:                      w,n :CARDINAL) :BOOLEAN;
18:
19: (**
20:     Among - Boolean function to check presence of an item in a list
21:
22:     G. H. Beers - November 1987
23:
24:     CALLING SEQUENCE - Among (f,l,w,n)
25:
26:     ENTRY - f  the item to test for
27:             l  the list to check
28:             w  width of each character string
29:             n  number of elements in list
30:
31:     EXIT -
32:
33: *)
34:
35:
36: PROCEDURE BlankData (string :ARRAY OF CHAR; nChar :CARDINAL) :BOOLEAN;
37:
38: (**
39:         BlankData - Tests string for being all blanks (or blank-nul)
40:
41:         T. H. Tucker - Aug 1987.
42:
43:
44:         ENTRY - string : array of characters to be tested
45:                 nChar  : number of characters to be tested in string
46:
47:         EXIT  - result : BOOLEAN function value
48:                          TRUE means string contains all blanks
49:                          FALSE means a non-blank resides in string
50:
51:
52: *)
53:
54:
55: PROCEDURE CheckRelationStatus (relation :ARRAY OF CHAR;
56:                                key      :ARRAY OF CHAR) : BOOLEAN;
57:
58: (**
59:     CheckRelationStatus - Checks for already open or closed relations
60:
61:     G. H. Beers - December 1987
62:
63:     Tests the initial state of a relation and returns status in a boolean
64:     variable after opening the relation if not already open.
65:
66:     CALLING SEQUENCE -
67:
68:       CheckRelationStatus (relation; key) : BOOLEAN
69:
70:     ENTRY -
71:
72:       relation : ARRAY OF CHAR
73:         Relation to test state and open if necessary
74:
75:       key      : ARRAY OF CHAR
76:         Key to use to test state
77:
```

```
78:     EXIT  -
79:       result : BOOLEAN
80:         TRUE  - if relation already open
81:         FALSE - if relation closed initially
82:
83:         relation is opened
84:
85:     FILES -
86:
87: *)
88:
89:
90: PROCEDURE ClearMessage;
91:
92: (**
93:     ClearMessage - Clears the currently displayed message from the screen.
94:
95:     G. H. Beers - April 1988
96:
97:     Issues a blank DisplayMessage to clear the message line on the screen.
98:
99:     CALLING SEQUENCE -
100:
101:       ClearMessage ')
102:
103:    ENTRY -
104:
105:    EXIT  -
106:
107: *)
108:
109:
110:
111: PROCEDURE ClearStack ()
112:         ;
113:
114: (**
115:    ClearStack - Clears off any extraneous junk from stack
116:
117:    G. H. Beers - December 1987
118:
119:    description
120:
121:    CALLING SEQUENCE -
122:
123:      ClearStack ()
124:
125:    ENTRY -
126:
127:    EXIT -
128:
129: *)
130:
131:
132: PROCEDURE ClearUEvent ();
133:
134: (**
135:    ClearUEvent - Deletes the UEvent relation for adhoc aggregations
136:
137:    G. H. Beers - December 1987
138:
139:    Opens, deletes, and closes the UEvent relation for new set of buffer
140:    records.
141:
142:
143:    CALLING SEQUENCE -
144:
145:      ClearUEvent ()
146:
147:
148:    ENTRY -
149:
150:      UEvent relation with or with records
151:
152:    EXIT  -
```

```
153: *)
154:
155:
156: PROCEDURE ConvertMilitaryDate (   date :ARRAY OF CHAR;
157:                                 VAR str :ARRAY OF CHAR);
158:
159: (**
160:         ConvertMilitaryDate - Convert date (yyyy/mm/dd) to Military form
161:
162:         T. H. Tucker
163:
164:         Converts given date (yyyy/mm/dd) to Military: dd mon yyyy; i.e.,
165:         change 1987/10/30  to  30 OCT 1987.
166:
167:
168:         ENTRY   - ARRAY OF CHAR:  yyyy/mm/dd (i.e., 1987/10/30)
169:
170:         EXIT  - str   military date equivalent (i.e., 30 OCT 1987)
171:                       str is ARRAY OF CHAR, at least 11-chars long.
172:
173: *)
174:
175:
176: PROCEDURE DisplaySageError (message :ARRAY OF CHAR);
177:
178: (**
179:         DisplaySageError - Display provided message & associated SageError value
180:
181:         T. H. Tucker
182:
183:         ENTRY   - message   messsage to be displayed
184: *)
185:
186:
187: PROCEDURE FailureGroupMember (VAR failID, failGrp :ARRAY OF CHAR) : BOOLEAN;
188:
189: (**
190:     FailureGroupMember - Returns flag indicating if failure mode in given group.
191:
192:     G. H. Beers - December 1987
193:
194:     Returns BOOLEAN flag indicating if the given failure mode belongs to the
195:     given failure group by locating the mode in relation FailureM and then
196:     determining if the given group matches the group FailGrp in the relation.
197:
198:     CALLING SEQUENCE -
199:
200:       FailureGroupMember (failID, failGrp) : BOOLEAN;
201:
202:     ENTRY -
203:
204:       failID - ARRAY OF CHAR
205:         The given failure Mode which is to be matched to the given group
206:
207:       failGrp - ARRAY OF CHAR
208:         The given failure Group which must match the given failure mode's
209:         group in the relation FailureM if a TRUE result is returned.
210:
211:     EXIT  -
212:
213:       result : BOOLEAN
214:         TRUE  - If the failure group of the given failure mode in FailureM
215:                 matches the given failure group "failGrp"
216:
217:         FALSE - If the failure group in FailureM does not match the given
218:                 failure group.
219:
220:      FILES -
221:
222: *)
223:
224:
225: PROCEDURE GetCatName (    category : CARDINAL;
226:                       VAR catName  : ARRAY OF CHAR);
227:
```

```
228: (**
229:         GetCatName - Return the category name
230:
231:         H.D. Stewart    07/10/87
232:
233:         This procedure returns the category name based on the
234:         given category number.
235:
236:         CALLING SEQUENCE -
237:
238:            GetCatName (category,catName)
239:
240:         ENTRY -
241:
242:            category : CARDINAL
243:                 The category number
244:
245:         EXIT -
246:
247:            catName : ARRAY OF CHAR
248:                 The name of the category (Mechanical, Electrical, etc.).
249:
250:         FILES -
251: *)
252:
253:
254: PROCEDURE GetFailureGroups(VAR comp :ARRAY OF CHAR;
255:                            VAR fgs  :ARRAY OF CHAR;
256:                            VAR fCnt :CARDINAL);
257: (**
258:     GetFailureGroups - assembles the valid failure groups for a component.
259:
260:     G. H. Beers - December 1987
261:
262:     Locates and counts valid failure groups for a given component and inserts
263:     the groups into an array.
264:
265:     CALLING SEQUENCE -
266:
267:       GetFailureGroups (comp, fgs; fCnt)
268:
269:     ENTRY -
270:
271:       comp - ARRAY OF CHAR
272:         Component for which failure groups will be located in the "CmpValid"
273:         relation.
274:
275:     EXIT  -
276:
277:       fgs : ARRAY OF CHAR
278:         Array of 9 characters which returns up to three valid failure groups
279:         of three characters each.
280:
281:       fCnt : CARDINAL
282:         Number of valid failure Groups located for given component.
283:
284:      FILES -
285:
286: *)
287:
288:
289: PROCEDURE MilitaryDate (VAR str :ARRAY OF CHAR);
290:
291: (**
292:         MilitaryDate - Return the current date in Military format
293:
294:         T. H. Tucker
295:
296:         Returns the current date in Military format: dd mmm yyyy
297:         e.g., 30 OCT 1987.
298:
299:
300:         EXIT  - str   value of current date in Military format.
301:                       str is ARRAY OF CHAR, at least 11-chars long.
302:
```

```
303: *)
304:
305:
306: PROCEDURE  MyCloseRelation (relation :ARRAY OF CHAR;
307:                             status   :BOOLEAN);
308:
309: (**
310:     MyCloseRelation - Closes relation or data file as necessary
311:
312:     G. H. Beers - December 1987
313:
314:     Checks the initial status of a relation and returns it to its initial state.
315:
316:     CALLING SEQUENCE -
317:
318:       MyCloseRelation (relation,status)
319:
320:     ENTRY -
321:
322:       relation - ARRAY OF CHAR
323:                  Name of relation to be opened or checked.
324:
325:       status   - BOOLEAN
326:                  inital relation status flag
327:                  TRUE  - Relation or index file was initially open
328:                  FALSE - Relation was initially closed
329:
330:     EXIT  -
331:
332:     FILES -
333:
334: *)
335:
336:
337: PROCEDURE  MyOpenRelation (relation, key :ARRAY OF CHAR;
338:                                    modify :BOOLEAN) :BOOLEAN;
339:
340: (**
341:     MyOpenRelation - Opens relation if necessary and returns initial status
342:
343:     G. H. Beers - December 1987
344:
345:     Checks the current status of a relation and opens it if necessary.
346:
347:     CALLING SEQUENCE -
348:
349:       MyOpenRelation (relation,key)
350:
351:     ENTRY -
352:
353:       relation - ARRAY OF CHAR
354:                  Name of relation to be opened or checked.
355:
356:       key      - ARRAY OF CHAR
357:                  Name of key to use for testing relation current status
358:
359:     EXIT  -
360:
361:       result   - BOOLEAN
362:                  Initial relation status flag
363:                  TRUE - Relation or index file was initially open
364:
365:                  FALSE - Relation was closed
366:
367:      FILES -
368:
369: *)
370:
371:
372: PROCEDURE PlacesC (m :CARDINAL) :CARDINAL;
373:
374: (**
375:         PlacesC - Determine character places needed for cardinal
376:
377:         T. H. Tucker
```

```
378:
379:        Determines how many character places are required to display the
380:        CARDINAL number, m.  This value is the number of digits  m
381:        takes up.
382:
383:
384:        ENTRY - m: CARDINAL number
385:
386:        EXIT  - (FUNCTION) result: number of character places to display m.
387: *)
388:
389:
390: PROCEDURE ShowSageError (string :ARRAY OF CHAR);
391:
392: (**
393:        ShowSageError - Write SageError (& string) below cursor position
394:
395:        T. H. Tucker
396:
397:        Writes the value of SageError & user-supplied string 1-line
398:        below the current cursor position.
399:
400:
401:        ENTRY - string - string User wants to display following SageError value
402:
403: *)
404:
405:
406: END General.
```

```
IMPLEMENTATION MODULE General;


FROM ASCII    IMPORT (* constant   *)
                nul
               ;

FROM Convert  IMPORT CardToStr, StrToCard;

FROM Sage     IMPORT ClearField,       ClearRelation,    CloseDataBase,
                     CloseRelation,    CompareFieldA,    CompareFieldC,
                     CopyField,        DefineFieldCheck, CloseRelationFiles,
                     DeleteRelation,
                     DeleteRecord,     DisplayMessage,   DisplayForm,
                     DisplayFormV,     DisplayFormVIP,   FindRecord,
                     GetFieldA,        GetFieldB,        GetFieldC,
                     GetFieldF,        GetFieldI,        GetFieldP,
                     GetRepeatName,    OpenDataBase,     OpenRelation,
                     PutFieldA,        PutFieldC,        PutFieldF,
                     PutFieldI,        ReWriteRecord,    ReadRecord,
                     SageError,        SageOperations,   WriteRecord,
                     GetBlock,         ReadRecordA,      RelationIsOpen;

FROM SortLib  IMPORT CompareKey, FillChar, MoveLeft, Sort, ScanChar;

FROM String   IMPORT Concat, Length, Position, Substring,
                     Compare, CompareResult;

FROM SYSTEM   IMPORT (* function   *)
                ADR
               ;

FROM Terminal IMPORT WriteLn, WriteString, CondRead;

FROM ThorPort IMPORT ClearScreen, CursorMove, FixFileName, Pause;

FROM ThorUtil IMPORT (* procedure  *)
                MovnString
               ;

FROM TimeLib  IMPORT CurrentDate1;


PROCEDURE Among (VAR f,l :ARRAY OF CHAR;
                     w,n :CARDINAL) :BOOLEAN;

(**
     Among - Boolean function to check presence of an item in a list

     G. H. Beers - November 1987

     CALLING SEQUENCE - Among (f,l,w,n)

     ENTRY - f  the item to test for
             l  the list to check
             w  width of each character string
             n  number of elements in list

     EXIT -

*)


VAR
  result :BOOLEAN;
  testit :ARRAY [0..19] OF CHAR;
  i,m    :CARDINAL;

BEGIN
  i := 0;
  result := FALSE;
  FillChar (testit, " ", 20);
  WHILE i < n DO
    MoveLeft ( l[i*w], testit, w);
```

```
75:     IF Compare (f, testit) = equal THEN
76:       result := TRUE;
77:       RETURN result;
78:     ELSE
79:       INC (i);
80:       END;
81:     END;  (* WHILE *)
82:   RETURN result;
83:   END Among;
84:
85:
86: PROCEDURE BlankData (string :ARRAY OF CHAR; nChar :CARDINAL) :BOOLEAN;
87:
88: (**
89:         BlankData - Tests string for being all blanks (or blank-nul)
90:
91:         T. H. Tucker - Aug 1987.
92:
93:
94:         ENTRY - string : array of characters to be tested
95:                 nChar  : number of characters to be tested in string
96:
97:         EXIT  - result : BOOLEAN function value
98:                          TRUE means string contains all blanks
99:                          FALSE means a non-blank resides in string
100:
101:
102: *)
103:
104:
105: VAR
106:   i        :CARDINAL;
107:   maxIndex :CARDINAL;
108:   result   :BOOLEAN;
109:
110: BEGIN
111:   result := TRUE;
112:
113:   IF (nChar > 0) THEN
114:     IF string[0] = CHR(0) THEN RETURN result; END;
115:     IF (string[0] = " ") THEN
116:       IF nChar > 1 THEN
117:         IF (string[1] = " ") THEN
118:           RETURN result;
119:         END;
120:       ELSE
121:         RETURN result;
122:       END;
123:     END;
124:
125:     maxIndex := HIGH (string);
126:     i := 0;
127:     LOOP
128:
129:       IF (i > maxIndex) THEN
130:         WriteLn;
131:         WriteString (" ERROR - exceeded HIGH (String) in BlankData");
132:         WriteLn;
133:       END;
134:
135:       IF (string[i] = CHR(0)) THEN RETURN result; END;
136:       IF (string[i] # " ") THEN
137:         result := FALSE;
138:         RETURN result;
139:       END;  (* IF *)
140:       INC (i);
141:       IF (i >= nChar) THEN EXIT; END;
142:     END;  (* LOOP *)
143:   END;  (* IF *)
144:   RETURN result;
145:
146: END BlankData;
147:
148:
149:
```

```
150: PROCEDURE CheckRelationStatus (relation :ARRAY OF CHAR;
151:                                key      :ARRAY OF CHAR) : BOOLEAN;
152:
153: (**
154:     CheckRelationStatus - Checks for already open or closed relations
155:
156:     G. H. Beers - December 1987
157:
158:     Tests the initial state of a relation and returns status in a boolean
159:     variable after opening the relation if not already open.
160:
161:     CALLING SEQUENCE -
162:
163:       CheckRelationStatus (relation; key) : BOOLEAN
164:
165:     ENTRY -
166:
167:       relation : ARRAY OF CHAR
168:         Relation to test state and open if necessary
169:
170:       key      : ARRAY OF CHAR
171:         Key to use to test state
172:
173:     EXIT  -
174:       result : BOOLEAN
175:         TRUE  - If relation already open
176:         FALSE - if relation closed initially
177:
178:         relation is opened
179:
180:     FILES -
181:
182: *)
183:
184: VAR
185:   result  :BOOLEAN;
186:
187: BEGIN
188:   result := TRUE;
189:   ReadRecord (relation, key, First);
190:   IF SageError = 106 THEN   (* relation not opened *)
191:     result := FALSE;
192:     END;
193:   RETURN result;
194:   END CheckRelationStatus ;
195:
196:
197: PROCEDURE ClearMessage;
198:
199: (**
200:     ClearMessage - Clears the currently displayed message from the screen.
201:
202:     G. H. Beers - April 1988
203:
204:     Issues a blank DisplayMessage to clear the message line on the screen.
205:
206:     CALLING SEQUENCE -
207:
208:       ClearMessage ()
209:
210:     ENTRY -
211:
212:       -
213:
214:     EXIT  -
215:
216:       -
217:
218:
219: *)
220:
221: BEGIN
222:   DisplayMessage (" ", FALSE);
223:
224: END ClearMessage;
```

```
225:
226:
227:
228: PROCEDURE ClearStack ()
229:         ;
230:
231: (**
232:     ClearStack - Clears off any extraneous junk from stack
233:
234:     G. H. Beers - December 1987
235:
236:     description
237:
238:     CALLING SEQUENCE -
239:
240:       ClearStack ()
241:
242:     ENTRY -
243:
244:
245:     EXIT  -
246:
247:
248:     FILES -
249:
250: *)
251:
252:
253: VAR
254:   ch        :CHAR;
255:   success   :BOOLEAN;
256:
257: BEGIN
258:
259:   LOOP
260:     CondRead (ch, success);
261:     IF NOT success THEN
262:       EXIT;
263:     END;
264:   END;  (* LOOP *)
265:
266: END ClearStack;
267:
268:
269: PROCEDURE ClearUEvent ();
270:
271: (**
272:     ClearUEvent - Deletes the UEvent relation for adhoc aggregations
273:
274:     G. H. Beers - December 1987
275:
276:     Opens, deletes, and closes the UEvent relation for new set of buffer
277:     records.
278:
279:     CALLING SEQUENCE -
280:
281:       ClearUEvent ()
282:
283:     ENTRY -
284:
285:        UEvent relation with or with records
286:
287:     EXIT  -
288:
289:     FILES -
290:
291: *)
292:
293: BEGIN
294:   OpenRelation ("UEvent",TRUE);
295:   DeleteRelation ("UEvent");
296:   CloseRelation ("UEvent");
297:
298: END ClearUEvent;
299:
```

```
300:
301: PROCEDURE ConvertMilitaryDate (   date :ARRAY OF CHAR;
302:                                 VAR str :ARRAY OF CHAR);
303:
304: (**
305:        ConvertMilitaryDate - Convert date (yyyy/mm/dd) to Military form
306:
307:        T. H. Tucker
308:
309:        Converts given date (yyyy/mm/dd) to Military: dd mon yyyy; i.e.,
310:        change 1987/10/30  to  30 OCT 1987.
311:
312:
313:        ENTRY   - /RRAY OF CHAR:  yyyy/mm/dd (i.e., 1987/10/30)
314:
315:        EXIT  - str   military date equivalent (i.e., 30 OCT 1987)
316:                      str is ARRAY OF CHAR, at least 11-chars long.
317:
318: *)
319:
320: VAR
321:   last    :CARDINAL;
322:   month   :ARRAY [0..3] OF CHAR;
323:   success :BOOLEAN;
324:
325: BEGIN
326:
327:   IF (HIGH(date) < 9) THEN
328:     WriteLn;
329:     WriteString ("PROCEDURE ConvertMilitaryDate: ");
330:     WriteString ("date is too small (10 ch. min.)");
331:     WriteLn;
332:     RETURN;
333:   END;
334:
335:   last := HIGH(str);
336:   IF (last > 10) THEN
337:     FillChar (str[11], CHR(0), 1);
338:     last := 10;
339:   END;
340:
341:   IF (last < 10) THEN
342:     WriteLn;
343:     WriteString ("PROCEDURE Military: argument is too small (11 ch. min.)");
344:     WriteLn;
345:     RETURN;
346:   END;
347:
348:   FillChar (str, CHR(32), 11);
349:   MoveLeft (date[8], str[0], 2);
350:   MoveLeft (date[0], str[7], 4);
351:
352:   MoveLeft (date[5], month, 2);
353:   FillChar (month[2], CHR(0), 1);
354:   StrToCard (month, last, success);
355:
356:   CASE last OF
357:      1:FillChar(str[3],"J",1); FillChar(str[4],"A",1); FillChar(str[5],"N",1)|
358:      2:FillChar(str[3],"F",1); FillChar(str[4],"E",1); FillChar(str[5],"B",1)|
359:      3:FillChar(str[3],"M",1); FillChar(str[4],"A",1); FillChar(str[5],"R",1)|
360:      4:FillChar(str[3],"A",1); FillChar(str[4],"P",1); FillChar(str[5],"R",1)|
361:      5:FillChar(str[3],"M",1); FillChar(str[4],"A",1); FillChar(str[5],"Y",1)|
362:      6:FillChar(str[3],"J",1); FillChar(str[4],"U",1); FillChar(str[5],"N",1)|
363:      7:FillChar(str[3],"J",1); FillChar(str[4],"U",1); FillChar(str[5],"L",1)|
364:      8:FillChar(str[3],"A",1); FillChar(str[4],"U",1); FillChar(str[5],"G",1)|
365:      9:FillChar(str[3],"S",1); FillChar(str[4],"E",1); FillChar(str[5],"P",1)|
366:     10:FillChar(str[3],"O",1); FillChar(str[4],"C",1); FillChar(str[5],"T",1)|
367:     11:FillChar(str[3],"N",1); FillChar(str[4],"O",1); FillChar(str[5],"V",1)|
368:     12:FillChar(str[3],"D",1); FillChar(str[4],"E",1); FillChar(str[5],"C",1);
369:   ELSE
370:     FillChar (str[3], "-", 3);
371:   END;  (* CASE *)
372:
373: END ConvertMilitaryDate;
374:
```

```
375:
376: PROCEDURE DisplaySageError (message :ARRAY OF CHAR);
377:
378: (**
379:         DisplaySageError - Display provided message & associated SageError value
380:
381:         T. H. Tucker
382:
383:         ENTRY   - message   messsage to be displayed
384: *)
385:
386:
387: VAR
388:   errorNumber :ARRAY [0..6] OF CHAR;
389:   str, string :ARRAY [0..79] OF CHAR;
390:   success     :BOOLEAN;
391:
392: BEGIN
393:
394:   FillChar (str, CHR(0), 80);
395:   FillChar (string, CHR(0), 80);
396:
397:   CardToStr (SageError,errorNumber,5,success);
398:   Concat (message," > ", str, success);
399:   Concat (str, errorNumber, string, success);
400:   DisplayMessage (string, FALSE);
401:   Pause (3000);
402: END DisplaySageError;
403:
404:
405: PROCEDURE FailureGroupMember (VAR failID, failGrp :ARRAY OF CHAR) : BOOLEAN;
406:
407: (**
408:     FailureGroupMember - Returns flag indicating if failure mode in given group.
409:
410:     G. H. Beers - December 1987
411:
412:     Returns BOOLEAN flag indicating if the given failure mode belongs to the
413:     given failure group by locating the mode in relation FailureM and then
414:     determining if the given group matches the group FailGrp in the relation.
415:
416:     CALLING SEQUENCE -
417:
418:       FailureGroupMember (failID, failGrp) : BOOLEAN;
419:
420:     ENTRY -
421:
422:       failID - ARRAY OF CHAR
423:         The given failure Mode which is to be matched to the given group
424:
425:       failGrp - ARRAY OF CHAR
426:         The given failure Group which must match the given failure mode's
427:         group in the relation FailureM if a TRUE result is returned.
428:
429:     EXIT  -
430:
431:       result : BOOLEAN
432:         TRUE  - If the failure group of the given failure mode in FailureM
433:                 matches the given failure group "failGrp"
434:
435:         FALSE - If the failure group in FailureM does not match the given
436:                 failure group.
437:
438:      FILES -
439:
440: *)
441:
442: VAR
443:   result : BOOLEAN;
444:   alreadyOpen :BOOLEAN;
445:
446: BEGIN
447:   IF failID[0] = failGrp[0] THEN
448:     result := TRUE;
449:   ELSE
```

```
450:     result := FALSE;
451:   END;
452:   RETURN result;
453: END FailureGroupMember;
454:
455:
456: PROCEDURE GetCatName (    category : CARDINAL;
457:                        VAR catName  : ARRAY OF CHAR);
458:
459: (**
460:        GetCatName - Return the category name
461:
462:        H.D. Stewart    07/10/87
463:
464:        This procedure returns the category name based on the
465:        given category number.
466:
467:        CALLING SEQUENCE -
468:
469:           GetCatName (category,catName)
470:
471:        ENTRY -
472:
473:           category : CARDINAL
474:                The category number
475:
476:        EXIT -
477:
478:           catName : ARRAY OF CHAR
479:                The name of the category (Mechanical, Electrical, etc.).
480:
481:        FILES -
482: *)
483:
484: BEGIN
485:   CASE category OF
486:     1 : MoveString ("Mechanical",catName,0)|
487:     2 : MoveString ("Electrical",catName,0);
488:   ELSE
489:     FillChar (catName,nul,HIGH(catName)+1);
490:   END;
491: END GetCatName;
492:
493:
494: PROCEDURE GetFailureGroups(VAR comp :ARRAY OF CHAR;
495:                            VAR fgs  :ARRAY OF CHAR;
496:                            VAR fCnt :CARDINAL);
497: (**
498:     GetFailureGroups - assembles the valid failure groups for a component.
499:
500:     G. H. Beers - December 1987
501:
502:     Locates and counts valid failure groups for a given component and inserts
503:     the groups into an array.
504:
505:     CALLING SEQUENCE -
506:
507:       GetFailureGroups (comp, fgs; fCnt)
508:
509:     ENTRY -
510:
511:       comp - ARRAY OF CHAR
512:         Component for which failure groups will be located in the "CmpValid"
513:         relation.
514:
515:     EXIT  -
516:
517:       fgs : ARRAY OF CHAR
518:         Array of 9 characters which returns up to three valid failure groups
519:         of three characters each.
520:
521:       fCnt : CARDINAL
522:         Number of valid failure Groups located for given component.
523:
524:      FILES -
```

```
525:
526: *)
527:
528: VAR
529:   fg          : ARRAY [0..2] OF CHAR;
530:   n0          : ARRAY [0..10] OF CHAR;
531:   alreadyOpen :BOOLEAN;
532:
533: BEGIN
534:   alreadyOpen := MyOpenRelation ("CmpValid", "Comp", FALSE);
535:   RecdRecordA("CmpValid", "Comp", EQ, comp);
536:   FillChar (fgs, " ", 9);
537:   fCnt := 0;
538:   LOOP
539:     GetRepeatName("FailGrp", fCnt+1, n0);
540:     GetFieldA ("CmpValid", n0, fg);
541:     IF BlankData(fg, 3) THEN
542:       EXIT;
543:     ELSE
544:       MoveLeft (fg, fgs[(fCnt)*3], 3);
545:     END;
546:     INC (fCnt);
547:     IF fCnt = 3 THEN EXIT; END;
548:   END;  (* LOOP *)
549:   MyCloseRelation ("CmpValid", alreadyOpen);
550: END GetFailureGroups;
551:
552:
553:
554: PROCEDURE MilitaryDate (VAR str :ARRAY OF CHAR);
555:
556: (**
557:         MilitaryDate - Return the current date in Military format
558:
559:         T. H. Tucker
560:
561:         Returns the current date in Military format: dd mmm yyyy
562:         e.g., 30 OCT 1987.
563:
564:
565:         EXIT  - str   value of current date in Military format.
566:                       str is ARRAY OF CHAR, at least 11-chars long.
567:
568: *)
569:
570:
571: VAR
572:   date    :ARRAY [0..9] OF CHAR;
573:   last    :CARDINAL;
574:   month   :ARRAY [0..3] OF CHAR;
575:   success :BOOLEAN;
576:
577:
578: BEGIN
579:
580:   CurrentDate1 (date);
581:
582:   last := HIGH(str);
583:   IF (last > 10) THEN
584:     FillChar (str[11], CHR(0), last-10);
585:     last := 10;
586:   END;
587:
588:   IF (last < 10) THEN
589:     WriteLn;
590:     WriteString ("PROCEDURE MilitaryDate: ");
591:     WriteString ("argument is too small (11 ch. min.)");
592:     WriteLn;
593:     RETURN;
594:   END;
595:
596:   FillChar (str, CHR(32), 11);
597:   MoveLeft (date[8], str[0], 2);
598:   MoveLeft (date[0], str[7], 4);
599:
```

```
600:   MoveLeft (date[3], month, 2);
601:   FillChar (month[2], CHR(0), 1);
602:   StrToCard (month, last, success);
603:
604:   CASE last OF
605:      1:FillChar(str[3],"J",1); FillChar(str[4],"A",1); FillChar(str[5],"N",1)|
606:      2:FillChar(str[3],"F",1); FillChar(str[4],"E",1); FillChar(str[5],"B",1)|
607:      3:FillChar(str[3],"M",1); FillChar(str[4],"A",1); FillChar(str[5],"R",1)|
608:      4:FillChar(str[3],"A",1); FillChar(str[4],"P",1); FillChar(str[5],"R",1)|
609:      5:FillChar(str[3],"M",1); FillChar(str[4],"A",1); FillChar(str[5],"Y",1)|
610:      6:FillChar(str[3],"J",1); FillChar(str[4],"U",1); FillChar(str[5],"N",1)|
611:      7:FillChar(str[3],"J",1); FillChar(str[4],"U",1); FillChar(str[5],"L",1)|
612:      8:FillChar(str[3],"A",1); FillChar(str[4],"U",1); FillChar(str[5],"G",1)|
613:      9:FillChar(str[3],"S",1); FillChar(str[4],"E",1); FillChar(str[5],"P",1)|
614:     10:FillChar(str[3],"O",1); FillChar(str[4],"C",1); FillChar(str[5],"T",1)|
615:     11:FillChar(str[3],"N",1); FillChar(str[4],"O",1); FillChar(str[5],"V",1)|
616:     12:FillChar(str[3],"D",1); FillChar(str[4],"E",1); FillChar(str[5],"C",1);
617:   ELSE
618:     FillChar (str[3], "-", 3);
619:   END;  (* CASE *)
620:
621: END MilitaryDate;
622:
623:
624: PROCEDURE MyCloseRelation (relation :ARRAY OF CHAR;
625:                            status   :BOOLEAN);
626:
627: (**
628:    MyClos.  ation - Closes relation or data file as necessary
629:
630:    G. H. Beers - December 1987
631:
632:    Checks the initial status of a relation and returns it to its initial state.
633:
634:    CALLING SEQUENCE -
635:
636:      MyCloseRelation (relation,status)
637:
638:    ENTRY -
639:
640:      relation - ARRAY OF CHAR
641:                 Name of relation to be opened or checked.
642:
643:      status   - BOOLEAN
644:                 inital relation status flag
645:                 TRUE  - Relation or index file was initially open
646:                 FALSE - Relation was initially closed
647:
648:    EXIT  -
649:
650:    FILES -
651:
652: *)
653:
654: BEGIN
655:   IF status THEN
656:     CloseRelationFiles (relation);
657:   ELSE
658:     CloseRelation (relation);
659:   END;
660: END MyCloseRelation;
661:
662:
663: PROCEDURE MyOpenRelation (relation, key :ARRAY OF CHAR;
664:                                modify :BOOLEAN) :BOOLEAN;
665:
666: (**
667:    MyOpenRelation - Opens relation if necessary and returns initial status
668:
669:    G. H. Beers - December 1987
670:
671:    Checks the current status of a relation and opens it if necessary.
672:
673:    CALLING SEQUENCE -
674:
```

```
675:       MyOpenRelation (rolation,key)
676:
677:     ENTRY -
678:
679:       relation - ARRAY OF CHAR
680:                  Name of relation to be opened or checked.
681:
682:       key      - ARRAY OF CHAR
683:                  Name of key to use for testing relation current status
684:
685:     EXIT  -
686:
687:       result   - BOOLEAN
688:                  Initial relation status flag
689:                  TRUE - Relation or index file was initially open
690:
691:                  FALSE - Relation was closed
692:
693:      FILES -
694:
695: *)
696:
697: VAR
698:   result :BOOLEAN;
699:
700: BEGIN
701:   result := TRUE;
702: (* IF NOT RelationIsOpen (relation) THEN  *)
703:
704:   FindRecord (relation, key, Next);
705:   IF  (SageError = 106)  THEN
706:     OpenRelation (relation, modify);
707: (*  CloseRelationFiles (relation);  *)
708:     result := FALSE
709:   END;
710:   RETURN result;
711: END MyOpenRelation;
712:
713:
714: PROCEDURE PlacesC (m :CARDINAL) :CARDINAL;
715:
716: (**
717:         PlacesC - Determine character places needed for cardinal
718:
719:         T. H. Tucker
720:
721:         Determines how many character places are required to display the
722:         CARDINAL number, m.  This value is the number of digits  m
723:         takes up.
724:
725:
726:         ENTRY - m: CARDINAL number
727:
728:         EXIT  - (FUNCTION) result: number of character places to display m
729: *)
730:
731:
732: VAR
733:   result :CARDINAL;
734:
735:
736: BEGIN
737:
738:   result := 6;
739:   IF (m < 10000) THEN result := 4; END;
740:   IF (m <  1000) THEN result := 3; END;
741:   IF (m <   100) THEN result := 2; END;
742:   IF (m <    10) THEN result := 1; END;
743:
744:   RETURN result;
745:
746: END PlacesC;
747:
748:
749: PROCEDURE ShowSageError (string :ARRAY OF CHAR);
```

```
750:
751: (**
752:         ShowSageError - Write SageError (& string) below cursor position
753:
754:         T. H. Tucker
755:
756:         Writes the value of SageError & user-supplied string 1-line
757:         below the current cursor position.
758:
759:
760:         ENTRY - string - string User wants to display following SageError value
761:
762: *)
763:
764:
765: VAR
766:   errorNumber :CARDINAL;
767:   str         :ARRAY [0..79] OF CHAR;
768:   str6        :ARRAY [0..5] OF CHAR;
769:   success     :BOOLEAN;
770:
771:
772: BEGIN
773:
774:   errorNumber := SageError;
775:   CardToStr (errorNumber, str6, PlacesC(errorNumber), success);
776:
777:   FillChar (str, CHR(0), 80);
778:   IF (success) THEN
779:     Concat ("SageError ",str6,str,success);
780:     Concat (str, " ", str, success);
781:     Concat (str, string, str, success);
782:   ELSE
783:     str := "?";
784:     Concat (str, string, str, success);
785:   END;
786:
787:   WriteLn;
788:   WriteString (str);
789:   WriteLn;
790:   Pause (2500);
791:
792: END ShowSageError;
793:
794:
795: END General.
```

```
1: MODULE RetrHard; (* Retrieves results for hardware *)
2:
3: (* ------------------------------------------------------------------
4:
5:    Program Title :  Nuclear Computerized Library for Assessing Reactor
6:                     Reliability (NUCLARR)
7:
8:    Developed For :  U. S. Nuclear Regulatory Commission
9:                     Office of Research
10:                    Division of Reactor System Safety
11:
12:                    FIN A6850
13:
14:   Date          :  FY 86
15:
16:   NRC Contact   :  T. G. Ryan    Phone : FTS 443-7619
17:
18:   Code Developer:  EG&G Idaho, Inc.
19:                    Idaho National Engineering Laboratory
20:
21:                    G. H. Beers   Phone : FTS 583-9288
22:                    O. J. Call    Phone : FTS 583-9114
23:                    T. H. Tucker  Phone : FTS 583-9123
24:
25:   This program was prepared for an agency of the United States
26:   Government.  Neither the United States Government nor any agency
27:   thereof, or any of their employees, makes any warranty, expressed or
28:   implied, or assumes any legal liability or responsibility for any
29:   third party's use, or the results of such use, of any portion of this
30:   program or represents that its use by such third party would not
31:   infringe privately owned rights.
32:
33:   ------------------------------------------------------------------ *)
34:
35:
36:               IMPORT (* module     *)
37:                Program
38:               ;
39:
40:
41: FROM ClrHouse  IMPORT (* procedure  *)
42:                Clearinghouse
43:               ;
44:
45: FROM OlayHw    IMPORT (* procedure  *)
46:                CheckForError ,turnCheckOff
47:               ;
48:
49: FROM Sage      IMPORT (* variable   *)
50:                SageError
51:               ,
52:
53:                      (* procedures *)
54:                OpenDataBase ,CloseDataBase  ,ClearRelation
55:               ,DisplayForm  ,DisplayMessage
56:               ,GetFieldA
57:               ;
58:
59: FROM ThorPort  IMPORT (* procedures *)
60:                ClearScreen
61:               ;
62:
63:
64: CONST
65:   MaxForms   = 5;
66:   MaxScreens = 3;
67:   BufferSize = 9000;
68:
69:   RingBell   = TRUE;
70:   NoDefault  = FALSE;
71:
72:
73: VAR
74:   aOption  : CHAR;
```

```
 75:   olStatus : Program.CallResult;
 76:   index    : CARDINAL;
 77:
 78:
 79: BEGIN (* RetrHard *)
 80:
 81:   (* ---------------------- initialization ------------------------- *)
 82:
 83:   OpenDataBase ("Hardware.DFL",MaxForms,MaxScreens,BufferSize);
 84:   IF  (SageError <> 0)  THEN
 85:     DisplayMessage (
 86:       "The schema file, 'Hardware.DFL', is not available",RingBell);
 87:     RETURN;
 88:   END; (* SageError IF *)
 89:
 90:
 91:   (* ------------------- set FilesLeftOpen Flag --------------------- *)
 92:
 93:   turnCheckOff := TRUE;
 94:
 95:   (* ------------------- main menu processing ----------------------- *)
 96:
 97:   LOOP
 98:     ClearRelation ("Utility");
 99:     DisplayForm ("MenuRH1","Utility","UA1",NoDefault);
100:     GetFieldA ("Utility","UA1",aOption);
101:
102:     olStatus := Program.normalReturn;
103:
104:     CASE  aOption  OF
105:       "E" : EXIT|
106:
107:       "?" : Clearinghouse ()|
108:
109:       "1" : DisplayMessage
110:               ("Loading Descriptive Search Programs ... please standby",
111:                     FALSE);
112:             Program.Call ("DHardw.LOD",olStatus);
113:             CheckForError (olStatus)|
114:
115:
116:       "2" : DisplayMessage
117:               ("Loading Ad Hoc Search Programs ... please standby",
118:                     FALSE);
119:             Program.Call ("AHardw.LOD",olStatus);
120:             CheckForError (olStatus);
121:
122:       "3" : Program.Call ("DocHardw.LOD",olStatus);
123:             CheckForError (olStatus)|
124:
125:       "4" : DisplayMessage("** Hardware Glossary is not yet complete **",
126:                            RingBell);
127:
128:     ELSE
129:       DisplayMessage ("Invalid Option",RingBell);
130:     END; (* CASE *)
131:   END; (* LOOP *)
132:
133:   ClearScreen ();
134:   CloseDataBase ();
135: END RetrHard.
```

INDEX

(Excludes Appendices)

NRC FORM 335 (8-87) NRCM 1102, 3201, 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

SEE INSTRUCTIONS ON THE REVERSE

1. REPORT NUMBER (Assigned by NRCM, DPS, add Vol. No., if any)
NUREG/CR-4639
EGG-2458
Volume 2

2. TITLE AND SUBTITLE
Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR)
Programmer's Guide

3. LEAVE BLANK

4. DATE REPORT COMPLETED
MONTH: September YEAR: 1988

5. AUTHOR(S)
Osmond J. Call, Jeffrey A. Jacobson

6. DATE REPORT ISSUED
MONTH: September YEAR: 1988

7. PERFORMING ORGANIZATION NAME AND MAILING ADDRESS (Include Zip Code)
EG&G Idaho, Inc.
Idaho Falls, ID 83415

8. PROJECT/TASK/WORK UNIT NUMBER

9. FIN OR GRANT NUMBER
A6850

10. SPONSORING ORGANIZATION NAME AND MAILING ADDRESS (Include Zip Code)
Division of Systems Research
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555

11a. TYPE OF REPORT
Technical

b. PERIOD COVERED (Inclusive dates)

12. SUPPLEMENTARY NOTES

13. ABSTRACT (200 words or less)

The Nuclear Computerized Library for Assessing Reactor Reliability (NUCLARR) is an automated data base management system for processing and storing human error probability and hardware component failure data. The NUCLARR system software resides on an IBM (or compatible) personal micro-computer and can be used to furnish data inputs for both human and hardware reliability analysis in support of a variety of risk assessment activities.

The NUCLARR system is documented in a five-volume series of reports. Volume II of this series is the Programmer's Guide for maintaining the NUCLARR system software. This Programmer's Guide provides, for the software engineer, an orientation to the software elements involved, discusses maintenance methods, and presents useful aids and examples.

14. DOCUMENT ANALYSIS - a. KEYWORDS/DESCRIPTORS
NUCLARR, computer code
human error probability
programmer's guide
NUCLARR system software

b. IDENTIFIERS/OPEN-ENDED TERMS

15. AVAILABILITY STATEMENT
Unlimited

16. SECURITY CLASSIFICATION
(This page) Unclassified
(This report) Unclassified

17. NUMBER OF PAGES

18. PRICE