UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

EARTHQUAKE DATA ARCHIVING AND RETRIEVAL SYSTEM: REFERENCE MANUAL

G. R. Crane, W. H. K. Lee, and J. T. Newberry

Open-file Report 84-840

October, 1984

CONTENTS

Introduction and System Overview 1

Part 1: Functional Overview 5

- Summary of command types 6
- Command syntax 7
  - Notation 7
  - Command Name 7
  - Command Operands 8
  - Command Options 8
- Brief Command Description 9
  - Directly Executable Commands 9
  - Utility Commands 10
- Functional Steps 11
  - Pre-Archiving Preparation 11
  - The ..SETUP Step 11
  - The ..COMBINE Step 12
  - The ..ARCHIVE Step 13
  - Additional SLAC Archive 13
  - Creating SAMPLE Data Sets 14
  - Cleanup 14
  - Adding SAMPLE Data Sets to SPIRES 14
- Trouble shooting 15
  - Errors in the Archive Job 15
  - Making Corrections to Archived Datasets 15

Part 2: Command Syntax Descriptions 16

- ADD 17
- APPLYUPD 18
- ARCHGET 19
- ARCHIVE 20
- ARCHSAMP 21
- ARCHSET 22
- CLEANUP 23
- CMD.INVENTORY 24
- CMD.INVENTORY.SEARCH 25
- COMBINE 26
- CUMSIZE 27
- DOTAPE 28
- INVENTORY.HELP 29
- LIBDATA 30
- MAKEUPD 31
- PROMPT 32
- REARCHIVE 33
- RPTSAMP 34
- RETRIEVE 35

REVISE 36
RTAPE 37
SETUP 39

Part 3: Database Organization 40

SPIRES Database Files 41
Subfile LIBDATA 41
Purpose 41
Elements 41
Search Terms 41
Specifications 41
Usage Notes 42
Sample Record 42
Subfile CLASS 44
Purpose 44
Elements 44
Search Terms 44
Sample Records 44
Subfile INSTITUTIONS 46
Purpose 46
Elements 46
Search Terms 46
Specifications 46
Usage Notes 46
Sample Record 47
Subfile INVENTORY 48
Purpose 48
Elements 48
Search Terms 50
Specifications 50
Usage Notes 51
Sample Record 51
Subfile TAPES 53
Purpose 53
Elements 53
Search Terms 53
Specifications 53
Usage Notes 53
Subfile QUAKE PROTO 54
Purpose 54
Elements 54
Search Terms 54
Specifications 54
Usage Notes 54

Part 4: Program Listings 55

ADD 56
ADD.INST.CODE 58

APPLYUPD 59
ARCHGET 60
ARCHIVE 62
ARCHSAMP 64
ARCHSET 68
BASE36 73
CLEANUP 74
CMD.INVENTORY.SEARCH 75
CMD.INVENTORY 81
COMBINE 83
CUMSIZE 88
DOTAPE 89
INVENTORY.HELP 91
LIBDATA 94
MAKEUPD 97
PROMPT 99
REARCHIVE 100
REPTSAMP 104
RETRIEVE 107
RTAPE 117
SETUP 130
USGS:INVENTORY 133
USGS:LIBDATA 140
USGS:LIBDATA 141
USGS:INVENTORY.NEWADD 143
USGS:INVENTORY.ADDCARDS 148
USGS:SETUP 153
VALUE 155

Part 5: References 159

INTRODUCTION AND SYSTEM OVERVIEW

This report is designed as a reference manual for the USGS Earthquake Data Archiving and Retrieval System. All the operational procedures in the system are described and the corresponding source codes are listed. It is intended to be used primarily by those who are responsible for the maintenance of the system. It may also be of interest to those who plan to organize a similar system and to those curious users who are wondering how the system works. A companion manual designed primarily for users is now in preparation and will be released soon.

The Earthquake Data Archiving and Retrieval System is designed to organize earthquake-related data in a systematic manner. It is a computer-based system to document, archive, query, and retrieve earthquake-related data sets.

A data set is defined here to be any convenient collection of data. However, in order for a data set to be useful, it must be self explanatory and have indexes for retrieval.

The overall scheme for organizing earthquake-related data is illustrated in Figure 1. We assume that data sets are provided by contributors in the form of magnetic tapes or punched cards with description of contents and formats. These input data sets will first be read onto a disk for editing using the "Edit System". An archivist will then use the "Archive System" to place the edited data sets on archival tapes. In the archiving process, data sets are copied, verified, and indexed in a "Database". Subsequently, the "Backup System" is used to backup the archival tapes, as well as to place the newly archived data sets on a large "Staging Disk". From users' point of view, they may use the "Query System" to search the "Database" for their desired data sets. The result of a search is a list of pointers to the desired data sets on the archival tapes. They are then passed to the "Retrieval System" for retrieval. The "Retrieval System" will first look for the data sets on the "Staging Disk". If the data sets are there, the response will be immediate, and users can display the data sets on their terminals or output them to magnetic tapes or to the printer. Otherwise, data sets will be retrieved from the archival tapes and placed on the "Staging Disk".

Because of the vast amounts of earthquake-related data, it is not economical or necessary to have all the earthquake-related data online. However, we do keep a sample of all the archived data sets online for users to examine before full retrieval. The "Staging Disk" has a capacity of 160 megabytes so that it can hold several hundred data sets temporarily. By scratching out the least used data sets, the "Staging Disk" will quickly contain the most frequently used data sets.

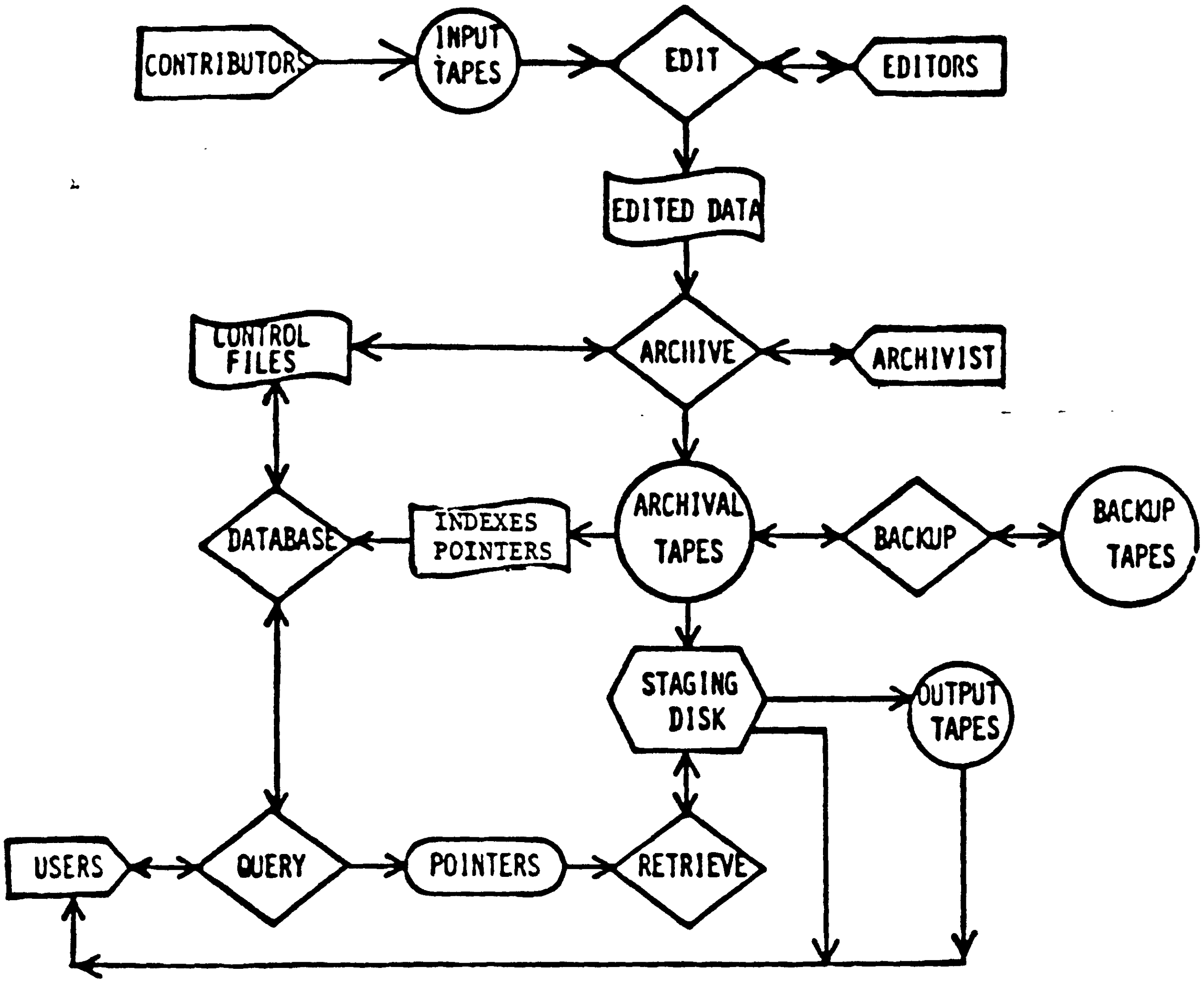

FIGURE 1: Overall scheme for organizing earthquake-related data.

The design of the system has been described in a report by Lee, Scharre, and Crane (1983) and wll not be repeated here. It must be emphasized that the archived data sets are independent of a computer system. They are basically alphanumeric files that can be processed by any general-purpose digital computer. However, the Archiving and Retrieval System must be implemented on a real computer and is thus dependent on the given computer environment.

The USGS Earthquake Data Archiving and Retrieval System has been implemented in the computer system operated by the Stanford Linear Accelerator Center. The system consists of a dual-processor IBM model 3081K main frame computer with 24 magabytes memory storage, tape drives, and high capacity disks. We use the VM/CMS operating system with SPIRES (Stanford Public Information REtrieval System) as the database management system, WYLBUR and XEDIT as the text editors, REXX as the system interpreter exec language and WYLBUR executive language (Please see the references for the software products we used). Our system requires 200 megabytes of on-line disk storage as well as access to several 7-track and 9-track tape drives.

The system is designed to archive any existing earthquake-related data. Because numerous formats exist for these data, it is not practical to force everyone to adopt a uniform format. Therefore, three distinct libraries have been set up for storing, query, and retrieval of earthquake data:

1. A General Library for data sets with arbitrary data structures and formats,

2. A Standardized Library for data sets using a standardized structure and recommended formats,

3. Waveform Libraries to handle the extremely large volume of seismic waveform data.

In the General Library, we archive data sets as received. Unfortunately, most data sets we have received do not have the information necessary to use them. Thus, much time has been spent in gathering the necessary information from various sources and incorporating it into the data sets themselves.

In order to search data sets by their individual records, we must have labels (or indexes) at the record level. The Standardized Library is introduced to store data sets that are coded according to our standardized structure and recommended formats. For the most frequently used data sets, we have written conversion programs to translate data sets in the General Library to the equivalent data sets to be stored in the Standardized Library. The Waveform Libraries consist of data sets that are usually in the standardized formats. They are separated from the Standardized Library because they contain extremely large data sets and are grouped by topics for ease of retrieval.

This report consists of 4 major parts. Part 1 describes the functional components of the system, giving an overview of command syntax and how the system fits together for general maintenance and use. Part 2 gives a detailed description of each command, protocol, or exec which exist to perform maintenance functions needed. Part 3 describes the organization of each of the database files which make up the entire system. And part 4 is a complete listing of all software written in support of this project.

Part 1

FUNCTIONAL OVERVIEW

The following diagram shows the functional flow from receiving the earthquake tape and information to the user's query and archive retrieval of information. Notice that there are basically three main steps, 1) data entry, 2) archival, and 3) query.

```
+-----------------------------------------------+
|                 Pre-Archival                  |
|               Data Preparation                |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                   ..SETUP                     |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                  ..COMBINE                    |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                  ..ARCHIVE                    |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|              ** SLAC Archive **               |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                   ARCHSAMP                    |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                    ..ADD                      |
+-----------------------------------------------+

+-----------------------------------------------+
|                    QUERY                      |
+-----------------------------------------------+
                       |
+-----------------------------------------------+
|                  RETRIEVAL                    |
+-----------------------------------------------+
```

SUMMARY OF COMMAND TYPES

There are three basic types of commands described in this document. The three types are:

1. **SPIRES PROTOCOL COMMANDS**

   Commands which execute a specific SPIRES program. These commands can be identified by the prefix of two periods (..) which must be included in the command. In addition, one must first be in the SPIRES environment. For example:

   ```
   ..SETUP
   ..ARCHIVE
   ```

2. **WYLBUR EXECS**

   Execs which may be identified by the need to issue "EXECUTE FROM" command before the command name. This class of commands are only meaningful if issued from the WYLBUR editor environment. For example:

   ```
   EXECUTE FROM CUMSIZE
   EXECUTE FROM CLEANUP
   ```

3. **CMS REXX EXECS**

   CMS execs which may be executed from any environment, that is, SPIRES and WYLBUR need not be active. However, many of the execs described in this document are called indirectly from SPIRES protocol command progams. For example:

   ```
   ARCHSAMP SL G B
   ```

COMMAND SYNTAX

The USGS Archive commands consist of a command name, usually followed by one or more positional operands and, in some cases, by an option list. All commands are of the form:

```
+---------------+-----------------------------------------------+
|               |                                               |
| command name  | [ operands...] [ (option list...[)]]          |
|               |                                               |
+---------------+-----------------------------------------------+
```

You must use one or more blanks to separate each entry in the command line unless otherwise indicated.

Notation

The following symbols are used to define the command format and should never be typed when the actual command is entered.

underscore	_
braces	{ }
brackets	[ ]
elipsis	...

Uppercase letters and words, and the following symbols, should be entered as specified in the format box.

asterisk	*
comma	,
hyphen	-
equal sign	=
parentheses	( )
period	.
colon	:

Command Name

When a command name is prefixed with two periods (..), then the command is a SPIRES protocol and the SPIRES environment must be entered before the command is issued.

When a command name is prefixed with "EXEC FROM", then the command is a WYLBUR exec program and the WYLBUR environment must be active.

All other commands are direct REXX exec language programs.

Command Operands

The command operands are keywords and/or positional operands of varying length. The operands specify the information on which the system operates when it performs the command function.

You must write the operands in the order in which they appear in the command format unless otherwise specified.

Command Options

The command options are keywords used to control the execution of the command. The options list must be preceded by a left parenthesis.

BRIEF COMMAND DESCRIPTION

The following is a brief description of the commands and support programs created for system maintenance and query. These command descriptions have been divided into two catagories, (1) directly executable commands and (2) commands which could be considered "Utilities".

Directly Executable Commands

ADD	Adds records to the main INVENTORY subfile using the SAMPLE datasets already created (see exec ARCHSAMP).
ARCHIVE	Used to create physical files on the staging disk to be moved to tape.
ARCHSAMP	Used to create sample data sets from ARCHIVE file.
COMBINE	Collects the INDEX, COMMENT, and datafile cards together to create a single archive file. Assigns archive file to tape file.
CUMSIZE	Adds up the cumulative size of multiple tape files using the library archive dataset.
CLEANUP	Prompts for number of lines to display. After display will prompt for line range to delete and verify.
LIBDATA	Used to modify or alter existing LIBDATA entries. Or to display current values for existing LIBDATA entries.
REVISE	Used to aid in modification of front end dataset material. User edits front end and program glues it back together.
REARCHIVE	Re-assigns a dataset to another tape file and re-calculates header info based on new size of dataset. (Analogous to COMBINE during initial archiving).
RETRIEVE	Sets up and submits a batch monitor request to retrieve specific datasets from the USGS archive tapes to the staging disk.
RTAPE	Retrieves datasets from non-archive system tapes.

RPTSAMP	Used to create a SCRIPT input report to summarize sample files.
SETUP	Used to collect INDEX and COMMENT card information, format and create two files.

Utility Commands

APPLYUPD	Applys updates made with the MAKEUPD exec.
ARCHGET	Executed by batch monitor to retrieve files from tape and put them directly onto the WHLW6 staging disk (WHLW6 198). Used by RETRIEVE.
ARCHSET	Creates batch monitor input exec to move archive files to tape. Called from protocol ..ARCHIVE.
DOTAPE	Aids batch monitor jobs in moving data files to tape.
CMD.INVENTORY	Main INVENTORY query protocol. Used for searching and query of the archive index.
CMD.INVENTORY.SRC	Main INVENTORY search protocol. Called from ..cmd.inventory. Prompts for information needed and returns with result of search.
INVENTORY.HELP	Displays a pre-coded help message associated with the single argument passed.
MAKEUPD	Creates update files for datasets stored on tape. Called from protocol ARCHIVE.
MENU	Driver menu to aid in function selection.
PROMPT	Utility protocol called from within other protocols to perform prompting and some input verification.

FUNCTIONAL STEPS

The following overview gives a description of each major step involved in the preparation and archiving of data. It is intended to be a guide to the setup procedures and commands needed in all phases of the data archival system. It does not include information on the search and query functions which are covered in the document "USGS Earthquake Archive Users Reference Manual".

Pre-Archiving Preparation

It is assumed here that the "data" have been moved from tape to disk. The steps or procedures needed to move or copy the data from tape are not covered in this document.

Once the data is somewhere on disk, you should follow these steps.

1. Copy or move the bare data set to the "H" disk (WHLW6 198) and assign it the file type "DATAFILE".

2. Prepare the FORMAT section of the comment file using WYLBUR or identify a previous file to use. This can be on any disk you like with any CMS filename: just remember what you called it.

3. Find the time span and latitude and longitude limits pertaining to the data in the data set.

The ..SETUP Step

This procedure builds two files: an index file and a comment file.

1. INDEX FILE

   This information pertains to the class of the data set, the limits in space and time of the data set, the persons responsible for submitting the data set, and the date that it was submitted.

2. COMMENT FILE

   This information pertains to the title, author, abstract, and the structure (format) of the data set.

The ..SETUP procedure is simply a facility that makes it easy to build these two files, put them in the correct form, and store them in the right place. You could duplicate the results by starting from scratch with the WYLBUR editor. Give the new files the same name that the data set you want to archive has. The files created have the file types 'INDEX' and 'COMMENT' and are stored by ..SETUP on your 'B' disk.

The ..COMBINE Step

This routine establishes the relationship within SPIRES between the index, comment, and datafile and combines them. At this point, the destination (assignment of a position in the data base) of the data set is calculated. The data set is assigned a position in a file on a specific archival tape, although it isn't placed on the tape yet. It's sort of like when you go to a travel agent who makes all the arrangements as to how you are going to get to Morocco, where you are going to stay when you get there, and how to find you while you are there in case someone needs to contact you or get you out of there, but the agent doesn't actually send you there yet.

The ..COMBINE session results in one file on disk called 'ss000nnn ARCHIVE H' where ss is 'GL', 'SL', or 'WL', and nnn is the data set number. The file consists of the DSN line which contains the information about the size of the data set and its destination and archival date and things like that, the lines from the INDEX file, the lines from the COMMENT file, the DATAFILE itself, and a finishing line at the end. The combine session also updates the SL, GL, or WL subfile record in SPIRES database LIBDATA (which you see when you say 'display SL' or whatever), and updates the SL, GL, or WL ARCHLIST. The "ARCHLIST" files contain detailed information on each of the archived (or ready to be archived) data sets. Information such as tape, file, size, data set name and date are stored as one line per data set. The ARCHLIST plays an important role in many of the functions described in this document. The DSN lines that show up on the ARCHLIST at this point begin with the characters 'C#', which tells us that they have been combined, although not yet archived.

If trouble occurs at this point, it still isn't any big deal or problem to back out of the archiving of the current group of data sets. You would have to change the SL, GL, or WL record back to its condition before you started that ..COMBINE session (using something called 'MERGE'), and you would have to delete the appropriate lines from the appropriate ARCHLIST. Its rather as if you decided not to go to Morocco after all and you needed the travel agent to cancel your reservations. The only thing you have lost is time.

The ..ARCHIVE Step

This procedure does the actual storing of the files on tape. It groups single data sets (that you have combined and that exist on the 'H' disk) into 'files' which can each contain several of the data sets. The ..COMBINE session already figured out how to put these data sets in to these files, so this routine does the grouping. You can think of the data sets as reprints or something, each consisting of a different number of stapled pages, and you can think of the bigger 'files' as folders that can contain several of the reprints, up to a certain finite capacity. You can then think of the archival tape as the file drawer you put the folders in.

The large files created (possibly containing several data sets, remember) at this stage are stored on the 'H' disk and look something like this: 'SM9302 FILE002 H'. The first part is the name of the tape where the file will soon be stored and the second part reflects the number of the file on that tape.

The ..ARCHIVE process then creates an 'exec' file which contains the instructions to submit the files just created to the archival tapes. This is called 'ARCHBAT EXEC A' and contains all of the commands that the BATCH job processor and the operator of the tape room at SLAC need to mount the right tape, set up the recording density, the block size, and all that.

When you respond to the 'Ok to submit?' prompt, the 'exec' file is executed and the job is submitted to the batch processor so the files will be written to the tapes. The batch processor sends you job status information at the terminal. It also sends detailed information which it punches to your reader, which you can look at to see in gory detail what happened. Look for a RC=0 (return code). If the RC is something other than 0, something goofed.

The batch processor does a comparison trick by writing the file from disk to tape, rewriting from tape to disk, and comparing the disk files. The results of this are shown in one of the files punched to your reader. Again, look for RC=0 to insure that all is well.

This completes the archiving into the SPIRES data base.

Additional SLAC Archive

This is a separate archival system, used by us as sort of a back up.

Since you have these big files built to be submitted to tapes (like 'SM9302 FILE003 H'), you can just archive them through SLAC. You don't do this through SPIRES.

The command 'ARCHIVE STORE filename filetype filemode' submits the big files to the SLAC archives and punches information to your reader.

Creating SAMPLE Data Sets

Sample data sets may be created at this point, or later as time permits. A SAMPLE dataset will consists of all of the 'header' type of information plus a few sample data records. Samples are created using the ARCHSAMP exec (ARCHSAMP dsn datamode samplmode) where 'dsn' is a dataset name or prefix portion of a group of dataset names, 'datamode' is the mode which contains the datasets and 'samplmode' is the mode letter which will contain the created samples. See the syntax explanation for ARCHSAMP rexx exec for a more detailed explanation.

Cleanup

Now you should go in and clean up by erasing or moving the files you won't need anymore, including the files of type DATAFILE, the INDEX and COMMENT files,and the big tape-structured files that you just submitted to the batch processor and SLAC archives.

Adding SAMPLE Data Sets to SPIRES

Once sample datasets are created they can be batch added to the INVENTORY SPIRES database. The SPIRES protocol ..ADD is used to collect, prepare and add each sample to the database. This same protocol would be used to update database entries when the sample datasets are changed.

The protocol will prompt for LIBRARY, range of datasets and disk which contains the sample datasets.

Samples added using the ..ADD protocol will not be searchable until the following day because it requires overnight processing of the SPIRES database.

TROUBLE SHOOTING

It's hoped that troubles in the system will be few and far between, however, they probably will occur. There are many places in a system of this size where problems requiring human intervention could pop up, i.e. a bad tape, a full disk, or perhaps a computer-wide system failure during a critical step. The most critial point to be careful in the earthquake archival system is while actually archiving or restoring data sets from tape.

Errors in the Archive Job

If errors occur in the archive job itself, it will be noted in the console files returned from the batch monitor. Examine the output to determine the cause. If re-running the job is neccessary, then editing of the "ARCHLIST" file will be needed to change "##" to "C#" in the approporiate line. Once that is done, the job may be re-run with the ..ARCHIVE command.

Making Corrections to Archived Datasets

If errors are discovered on previously archived data sets, then you may make corrections using ..REVISE and then assign the data set to a new tape file using the ..REARCHIVE program. The data set will be placed on tape when the next archive job is run.

Part 2

COMMAND SYNTAX DESCRIPTIONS

The following pages are detailed descriptions of each command available either directly to the user or from an internal subroutine call.

Notation convention - (1). "Utility" is a program command normally executed from within another program. (2) "Protocol" is a SPIRES protocol. (3) "Rexx" is a CMS/executive language interpreter. (4) "WYLBUR" is a command written in the SLAC WYLBUR exec language.

ADD

Adds records to the main QUERY system database using the SAMPLE data sets already created. (Protocol).

```
+------------------------------------------------------------+
|                                                            |
|   ..ADD                                                    |
|                                                            |
+------------------------------------------------------------+
```

Usage Notes:

1. Will prompt for beginning and ending sequence number in addition to the library name (GL, WL etc). Then it will attempt to find and add all SAMPLE datasets within the given range.

APPLYUPD

Used to apply updates previously created by the MAKEUPD exec. (WYLBUR Exec).

```
+------------------------------------------------------------------+
|                                                                  |
| EXEC FROM APPLYUPD WYLBUR                                        |
|                                                                  |
+------------------------------------------------------------------+
```

Usage Notes:

1. Command must be issued from within the WYBLUR environment.

ARCHGET

Executed by batch monitor to retrieve files from tape and put directly onto the WHLW6 staging disk (whlw6 198). (Rexx, Utility).

```
+------------+------------------------------------------------+
|            |                                                |
|  ARCHGET   |    password userid                             |
|            |                                                |
+------------+------------------------------------------------+
```

Where:

password	is the write password to the WHLW6 198 staging disk.
userid	is the userid of the submitting user which is to receive messages from the batch monitor retrieval program.

Usage Notes:

1. Attempts to gain write access to the WHLW6 198 staging disk.

2. Would only normally be executed via a batch monitor program submitted by using the RETRIEVE exec.

3. List of tape files to retrieve is expected to be found in a file called RETRIEVE TAPELIST.

4. Creates and returns to user a file called RETRIEVE LOG which indicates files moved and status.

<u>ARCHIVE</u>

Used to create physical files on the staging disk to be moved to tape. (Protocol).

```
+------------------------------------------------------------------+
|                                                                  |
|   ..ARCHIVE                                                      |
|                                                                  |
+------------------------------------------------------------------+
```

Usage Notes:

1. Prompts for library and modes of archive information and staging disk. Executes the REXX exec "ARCHSET" which will create physical file on staging disk to be moved to tape. Also created is an exec to be used by the BATCH MONITOR system for tape transfer and verification.

2. Upon completion of the exec it will update the library information in the LIBDATA and set it for the next file.

3. Three files will be returned to your reader,

   CON FILE
   GOOD ARCHIVE
   ARCHIVE LOG

4. In addition, if submitting user remains logged onto the system at the time the batch job is running, several informational messages may be received indicating success or failure of specific job steps.

ARCHSAMP

Used to create sample data sets from the ARCHIVE files. (Rexx).

```
+-----------+------------------------------------------------+
|           |                                                |
|  ARCHSAMP |  dsn datamode sampmode [ ( REP ]               |
|           |                                                |
+-----------+------------------------------------------------+
```

Where:

dsn	is the dataset name or dataset name prefix to create samples from.
datamode	disk mode which contains the data (archive) file.
sampmode	disk mode which is to contain the created samples.

ARCHSET

Creates batch monitor input exec to move archive files to tape. Called from protocol ..ARCHIVE (Protocol, Utility).

```
+--------------+---------------------------------------------+
|              |                                             |
|   ..ARCHSET  |  library dmode hmode                        |
|              |                                             |
+--------------+---------------------------------------------+
```

Where:

library	is the library name for datasets to be archived, (GL, WL etc).
dmode	is the current mode of the disk which contains the archive data sets created from ..COMBINE.
hmode	is the current mode of the staging disk.

Usage Notes:

1. Will combine multiple archive files into one or more physical tape files.

2. Generally called from the SPIRES protocol ..ARCHIVE.

CLEANUP

Prompts for number of lines to display. After display will prompt for line range to delete and verify. (WYLBUR).

```
+-------------------------------------------------------------+
|                                                             |
| EXEC FROM CLEANUP WYLBUR                                    |
|                                                             |
+-------------------------------------------------------------+
```

Usage Notes:

1. Command must be issued from within the WYLBUR environment.

CMD.INVENTORY

This is the main INVENTORY query protocol. Will prompt for a command and depending on the response will execute various other protocols to search etc. (Protocol, Utility).

```
+------------------------------------------------------------------+
|                                                                  |
|   ..CMD.INVENTORY                                                |
|                                                                  |
+------------------------------------------------------------------+
```

Usage notes:

1. This protocol would not normally be called directly as a command. In a production environment it will be executed from within another SPIRES protocol when the QUERY subfile is selected.

CMD.INVENTORY.SEARCH

This is the main INVENTORY search protocol, called from CMD.INVENTORY. It will prompt for various information needed to perform a complete search and return to calling program with search results. (Protocol, Utility).

```
+--------------------------------------------------------------+
|                                                              |
|  ..CMD.INVENTORY.SEARCH                                      |
|                                                              |
+--------------------------------------------------------------+
```

Usage notes:

1. This protocol would not normally be called directly as a user tool. In a production environment it will be executed from within the main inventory protocol (CMD.INVENOTRY).

COMBINE

Collects the INDEX, COMMENT and datafile cards together to create a single archive file. (Protocol).

```
+-----------------------------------------------------------------+
|                                                                 |
|   ..COMBINE                                                     |
|                                                                 |
+-----------------------------------------------------------------+
```

Usage Notes:

1. Adds the header and footer cards and updates LIBDATA to reflect current tape information.

CUMSIZE

Add up the cumulative size of multiple tape files using the library archive dataset. (WYLBUR exec).

```
+-----------------------------------------------------------+
|                                                           |
| EXEC FROM CUMSIZE WYLBUR                                  |
|                                                           |
+-----------------------------------------------------------+
```

Usage Notes:

1. Command must be issued from within the WYBLUR environment.

DOTAPE

Utility to aid batch monitor jobs in moving data files to tape. (Rexx, Utility).

```
+--------------+------------------------------------------------+
|              |                                                |
|   DOTAPE     |  dsn tape fileno xxx userid                    |
|              |                                                |
+--------------+------------------------------------------------+
```

Where:

dsn	is the current (or last) dsn in the currently archived tape file.
tape	is the name of the tape.
fileno	is the current tape file number.
xxx	is either '6250' or 'LEAVE' depending on whether this is a first or subsequent file being processed in this session.
userid	is the userid of the user to receive messages from the batch monitor as it moves and compares files.

Usage Notes:

1. This exec is only used by a batch monitor while moving files to archive tape. It should be considered a utility and not a user interface exec.

INVENTORY.HELP

Display a pre-coded help message associated with the single argument passed. (Protocol, Utility).

```
+-------------------+-----------------------------------------+
|                   |                                         |
|  ..INVENTORY.HELP | helpkey                                 |
|                   |                                         |
+-------------------+-----------------------------------------+
```

Where:

helpkey   is the value of a help label within the protocol.

Usage notes:

1. Normally this protocol would be called from within the prompting protocol (..PROMPT).

2. To add or change a help message the protocol must be modified. Each help message is identified by a beginning label (helpkey) and ending RETURN command. The following example should be followed when modifying this protocol (or other protocols in the QUAKE PROTO subfile).

```
SELECT QUAKE PROTO
TRANSFER INVENTORY.HELP CLEAR

-- make WYLBUR or xedit changes --

UPDATE
```

LIBDATA

Used to modify or alter existing LIBDATA entries. Or to display current values for existing LIBDATA entries. (Protocol).

```
+------------------------------------------------------------------+
|                                                                  |
|    ..LIBDATA                                                     |
|                                                                  |
+------------------------------------------------------------------+
```

Usage Notes:

1. Will prompt for desired library name (GL,SL, etc). Then it will display current values and begin prompting for concurence of existing values or new values. Once all new values are entered you will be given a chance to redo, update or exit with no update.

MAKEUPD

Used to make update files for stored datasets. (WYLBUR exec).

```
+------------------------------------------------------------+
|                                                            |
| EXEC FROM MAKEUPD WYLBUR                                   |
|                                                            |
+------------------------------------------------------------+
```

Usage Notes:

1. Command must be issued from within the WYLBUR environment.

PROMPT

This is a utility protocol called from within other protocols to perform prompting and some input verifications. (Protocol, Utility).

```
+-----------+---------------------------------------------+
|           |                                             |
|  ..PROMPT |  helpkey 'promptv'                          |
|           |                                             |
+-----------+---------------------------------------------+
```

Where:

helpkey	is a key identifying a help message associated with this prompt. If the user enters a question mark (?) or the string HELP then protocol PROMPT will pass control to another protocol which will display a help message and return control to PROMPT (which will re-prompt).
promptv	is a string which will be used as the prompt string. col

Usage notes:

1. This protocol would not normally be called directly as a user tool. PROMPT requires two arguments as shown in the syntax.

2. Help text should be included in another SPIRES protocol named INVENTORY.HELP as an executable proc within that protocol. If the help is not found, then a message to that effect will be displayed.

REARCHIVE

Sets up modified data set to be re-archived on different tape file than its original archive. Takes care of modification of header line plus updating of LIBDATA file. (Protocol).

```
+--------------+----------------------------------------------+
|              |                                              |
|   ..REARCHIVE|                                              |
|              |                                              |
+--------------+----------------------------------------------+
```

Usage Notes:

1. This protocol first assumes that the file has already been archived and modifications are being made.

2. The ARCHLIST file will be updated to include an additional line.

RPTSAMP

Used to create a SCRIPT input report to summarize sample files. (Rexx).

```
+-----------+------------------------------------------------+
|           |                                                |
|  RPTSAMP  |  dsn sampmode report-name [ ( REP ]            |
|           |                                                |
+-----------+------------------------------------------------+
```

Where:

dsn	is the dataset name or dataset name prefix to create samples from.
sampmode	disk mode which contains the sample datasets.
report-name	is the filename to be used for the created SCRIPT input file.
REP	is an optional parameter that indicates the report-name file should be replaced if it exists.

RETRIEVE

Set up and submit a batch monitor request to retrieve specific datasets from the USGS archive tapes to the staging disk. (Rexx).

```
+------------+---------------------------------------------+
|            |                                             |
|            |     |                  |                    |
|  RETRIEVE  |     | fn ft [ fm ]     |                    |
|            |     | SUBMIT pswd      |                    |
|            |     |                  |                    |
|            |                                             |
|            |                                             |
+------------+---------------------------------------------+
```

Where:

fn ft fm	is the filename of a file that contains records of the form found in a 'library ARCHLIST' file. If optional parameter is missing then prompting will aid in creating the list.
SUBMIT	indicates that a previously created retrieve list is to be submitted to the batch monitor system.
pswd	is the write password of the WHLW6 198 staging disk.

Usage Notes:

1. Optional SUBMIT is used to submit the batch job exec created from a previous RETRIEVE fn ft... request.

2. Exec will test the R/W status of the staging disk and require all current users to detach the disk so that the batch monitor will be able to get write status.

REVISE

Used to aid in the modification of a data set's front end material after it had previously been archived to tape. Will pull the front end material into WYLBUR, allow modification and then glue it back together, replacing the original. (Protocol).

```
+-----------------------------+-----------------------------------+
|                             |                                   |
|    ..REVISE                 |                                   |
|        or                   |                                   |
|    EXEC FROM REVISE WYLBUR  |                                   |
|                             |                                   |
|                             |                                   |
+-----------------------------+-----------------------------------+
```

Usage Notes:

1. Calls a WYLBUR exec REVISE, therefore WYLBUR must be active.

RTAPE

RTAPE reads files from a tape based on a control file. Files may be placed on a specific disk if requested or punched to the user. (Rexx, Utility).

```
+---------+--------------------------------------------------+
|         |                                                  |
|         |                                                  |
|         |                                                  |
|         |  READ   listname ( optiona                       |
|         |                                                  |
|  RTAPE  |  SUBMIT listname ( optiona                       |
|         |                                                  |
|         |  REVIEW [ listname ]                             |
|         |                                                  |
|         |  PROMPT [ listname ]                             |
|         |                                                  |
|         |                                                  |
|         |    optiona:     [LABEL label         ]           |
|         |                 [LRECL lrecl         ]           |
|         |                 [DEN density         ]           |
|         |                 [RECFM recfm         ]           |
|         |                 [BLKSIZE blksize     ]           |
|         |                 [SETUP YES|NO        ]           |
|         |                 [TAPE tapename       ]           |
|         |                 [OWNER disk-owner    ]           |
|         |                 [MODE disk-mode      ]           |
|         |                 [VADDR disk-vaddr    ]           |
|         |                 [PASSWORD disk-pswd  ] ]         |
|         |                                                  |
|         |                                                  |
+---------+--------------------------------------------------+
```

Where:

listname	is the cms filename of a file containing tape file information. The filetype of this file will always be TAPCNTRL.
SETUP	indicates if a tape setup is needed. Allowable values are YES or NO, default is YES.
TAPE	the name given to the tape when submitted to the SLAC library.
OWNER	is the userid owning the disk where data files are to be put. Only needed if files are not being returned via reader files.

VADDR	is the virtual address of the disk where data files are to be put. This is a required value if OWNER is specified.
PASSWORD	is the multi-write password for the disk owned by OWNER and at the virtual address of VADDR.
LABEL	is the tape label type. Values are BLP, SL, NL, AL where NL is the default.
DEN	is the tape density of 7TRK, 800, 1600 or 6250 bpi where 1600 is the default.
LRECL	is the logical record length, default = 80.
RECFM	is the record format, values are F, FB, V, VB, U, FA, FBA where default is FB.
BLKSIZE	is the record blocksize to be used if not found in the tape control file. Default = 1600.
FILES	is a single integer value or range of values (nnn-yyy) of specific files to restore.
TRTCH	is the tape recording technique for 7 track tapes. Values are O, OC, OT, E, ET, where default is OC.

Usage Notes:

1. This exec only operates on files listed in the cms file whose filetype is TAPCNTRL. You may use this exec to create the tape control file with the PROMPT command.

2. Most prompts can be terminated by the response of QUIT.

3. The normal sequence of commands to recover files from a user's tape would be RTAPE PROMPT, which creates a tape control file, and then RTAPE SUBMIT.

SETUP

Use SETUP to collect INDEX and COMMENT card information, format and create two files, one for index information and another for comment information related to a particular file name. (Protocol).

```
+------------+------------------------------------------------+
|            |                                                |
|  ..SETUP   |   [ INDEX | COMMENT | BOTH ]                   |
|            |                                                |
+------------+------------------------------------------------+
```

Where:

INDEX	is the optional parameter which indicates that only INDEX information should be prompted for. This information pertains to the class of the data set, the limits in space and time of the data set, the person responsible for submitting the data set, and the date that it was submitted.
COMMENT	is the optional parameter which indicates that only COMMENT information should be prompted for. This information pertains to the contents of the data set, the author of the data set itself, and the structure (format) of the data set.
BOTH	is the default parameter which indicates that both COMMENT and INDEX information will be prompted for.

Usage Notes:

1. If optional parameters are not given, then BOTH will be assumed as the default and prompting will begin with INDEX.

Part 3

DATABASE ORGANIZATION

The following is a detailed deSCRIPTion of each database subfile. A subfile is defined as a database of related information similar to a file drawer in a filing cabinet. Each subfile will consist of its own unique elements and searchable indexes. A SPIRES file definition is generally used to describe the entire "filing cabinet", which may contain one or more subfiles.

SPIRES DATABASE FILES

Subfile LIBDATA

Purpose: This subfile serves a very specific and narrow purpose. When full grown it will only contain a few (about 5) records. Each record maintains information on the current archive status of a partiular library (GL, SL, WL, etc). Information such as the current archive tape, file number, and total records written thus far are stored in this subfile. SPIRES protocols access this data while creating archive files and archive batch execs.

Elements: The following is a listing of the elements and element alias names used.

```
+------------------------------------------------------------------+
|                                                                  |
|                   Elements of Subfile LIBDATA                    |
|                                                                  |
|      ** Required **                                              |
|      LIBRARY  (Key)                                              |
|      ** Optional **                                              |
|      TAPEVOL, TAPENAME, TAPE                                     |
|      DSN                                                         |
|      FILENO, FILE                                                |
|      START, RECNO, REC                                           |
|      SIZE                                                        |
|      CUMSIZE, NCARDS, NCARD                                      |
|      UPDACCT                                                     |
|      UPDDATE                                                     |
|      UPDTIME                                                     |
|                                                                  |
+------------------------------------------------------------------+
```

Search Terms: Because of the simple structure and limited number of records expected to be saved in this database, no search terms are provided.

Specifications:

```
+------------------------------------------------------------+
|                                                            |
|                 Detail of Subfile LIBDATA                  |
|                                                            |
|     Owning account:  USGS                                  |
|     Disk          :  USGS 200                              |
|     Filedef       :  USGS: LIBDATA                         |
|     Format(s)     :  USGS: LIBDATA                         |
|                   :  $PROMPT                               |
|     Processing    :  Overnight by System                   |
|     Accessed-by   :  ..SETUP                               |
|                   :  ..ARCHIVE                             |
|     Space used    :  Approx 66 bytes/rec                   |
|     Overhead      :  37%                                   |
|                                                            |
+------------------------------------------------------------+
```

Usage Notes: All record maintenance should be performed by using the system provided $PROMPT format. For detailed deSCRIPTion of the usage of $prompt see the SPIRES documentation. Format LIBDATA is used internally by accessing protocols and should not be used for general subfile maintenance.

Before any record maintenance or query is performed you must select the subfile and set the $PROMPT format as follows:

```
+------------------------------------------------------------+
|                                                            |
|     SELECT LIBDATA                                         |
|     SET FORMAT $PROMPT                                     |
|                                                            |
+------------------------------------------------------------+
```

At this point you may use one of three common commands to manipulate any record in the database.

1. ADD -- to add a new library name to the database

2. REMOVE xx -- to remove a specific library name where "xx" would be replaced by the approporiate name (GL, WL, SL etc).

3. MERGE xx -- to alter or modify some of the information in a specific library where "xx" would be replaced by the approporiate name (GL, WL etc).

Sample Record:

```
+-------------------------------------------------------------------+
|                                                                   |
|                Sample Record in Subfile LIBDATA                   |
|                                                                   |
|     LIBRARY        GL                                             |
|     TAPEVOL        SM9309                                         |
|     DSN            47                                             |
|     FILENO         36                                             |
|     START          0                                              |
|     SIZE           0                                              |
|     CUMSIZE        557377                                         |
|     UPDACCT        USGS                                           |
|     UPDDATE        09/15/83                                       |
|     UPDTIME        09:04:36                                       |
|                                                                   |
+-------------------------------------------------------------------+
```

Subfile CLASS

Purpose: The CLASS subfile contains the current list of class and subclass cross reference information. Each class/subclass combination is maintained as two or more words separated by blanks. The first word being the "CLASS", the second and subsequent words are considerd part of the subclass definition. Each class/subclass is marked with a code (1A, 4C etc) which is not used at the current time. This subfile is only used as a lookup table for searching and data-entry verification purposes.

Elements:

```
+----------------------------------------------------------+
|                                                          |
|                Elements of Subfile CLASS                 |
|                                                          |
|     ** Required **                                       |
|     CLASS  (Key)                                         |
|     ** Optional **                                       |
|     CODE                                                 |
|                                                          |
+----------------------------------------------------------+
```

Search Terms: The CODE element is indexed in this subfile which serves an additional lookup purpose for adding and searching.

```
+----------------------------------------------------------+
|                                                          |
|              Search Terms for Subfile CLASS              |
|                                                          |
|     Goal Records:  RECORD                                |
|     Simple Index:  CODE                                  |
|                                                          |
+----------------------------------------------------------+
```

Sample Records:

```
+----------------------------------------------------------+
|                                                          |
|              Sample Records in Subfile CLASS             |
|                                                          |
|     CLASS        EARTHQUAKE PHASE                        |
|     CODE         1A                                      |
|                                                          |
|     CLASS        EARTHQUAKE STATION                      |
|     CODE         1B                                      |
|                                                          |
|     CLASS        EARTHQUAKE SUMMARY                      |
|     CODE         1C                                      |
|                                                          |
+----------------------------------------------------------+
```

Subfile INSTITUTIONS

Purpose: The INSTITUTIONS subfile is generally used as a lookup table by input processing rules and SPIRES protocols. In addition to the institution name, address, and other specific information, it contains a unique institution code which is to be used in searching and data input.

Elements:

```
+--------------------------------------------------------------+
|                                                              |
|              Elements of Subfile INSTITUTIONS                |
|                                                              |
|     ** Required **                                           |
|     INST-CODE  (Key)                                         |
|     ** Optional **                                           |
|     INST-NAME                                                |
|     INST-ADDR                                                |
|     CONTACT                                                  |
|     GEO-CODE, GC                                             |
|     COUNTRY-CODE, CC                                         |
|     STATE-CODE, SC                                           |
|     CITY-SORT, CS                                            |
|                                                              |
+--------------------------------------------------------------+
```

Search Terms: Goal Records: RECORD Simple Index: INST-NAME

Specifications:

```
+--------------------------------------------------------------+
|                                                              |
|               Detail of Subfile INSTITUTIONS                 |
|                                                              |
|     Owning account:  USGS                                    |
|     Disk          :  USGS 200                                |
|     Filedef       :  USGS: INVENTORY                         |
|     Format(s)     :  USGS:                                   |
|                   :  $PROMPT                                 |
|     Processing    :  Overnight by System                     |
|     Accessed-by   :  ..SETUP                                 |
|     Space used    :  Approx 94 bytes/rec                     |
|     Overhead      :  20%                                     |
|                                                              |
+--------------------------------------------------------------+
```

Usage Notes: Maintenance of records to this subfile should be performed using the system $PROMPT format. The institution codes (INST-CODE) element must be a unique value.

Before any record maintenance or query is performed you must select the subfile and set the $PROMPT format as follows:

```
+------------------------------------------------------------+
|                                                            |
|     SELECT INSTITUTIONS                                    |
|     SET FORMAT $PROMPT                                     |
|                                                            |
+------------------------------------------------------------+
```

At this point you may use one of three common commands to manipulate any record in the database.

1. ADD -- to add a new institution code and name to the database

2. REMOVE xx -- to remove a specific institution, where "xx" would be replaced by the appropriate institution code.

3. MERGE xx -- to alter or modify some of the information in a specific institution, where "xx" would be replaced by the appropriate institution code.

Sample Record: Note that element INST-ADDR in the following sample records is a multiply occuring element and may have more than one occurrence. The norm, however, is only a single occurrence of this element.

```
+------------------------------------------------------------+
|                                                            |
|           Sample Records of Subfile INSTITUTIONS           |
|                                                            |
|     INST-CODE       ABAG                                   |
|     INST-NAME       ASSOCIATION OF BAY AREA GOVERNMENTS    |
|     INST-ADDR(1)    BERKELEY, CA 94705                     |
|                                                            |
|     INST-CODE       BRN                                    |
|     INST-NAME       BROWN UNIVERSITY                       |
|     INST-ADDR(1)    PROVIDENCE, RI 02912                   |
|                                                            |
|     INST-CODE       CDMG                                   |
|     INST-NAME       CALIFORNIA DIVISION OF MINES AND GEOLOGY |
|     INST-ADDR(1)    SACRAMENTO, CA 95816                   |
|                                                            |
+------------------------------------------------------------+
```

Subfile INVENTORY

Purpose: This is the main query and archive system subfile. Data contained here is extracted from sample archive datasets. Searching is performed on this database via protocol driven-prompts. Data input and display will normally be handled via dedicated formats designed for that purpose (rather than using a system provided format like $PROMPT). When INVENTORY is selected as a subfile the main query system protocol will automatically begin execution. Depending on the user response, a variety of protocols, formats, and execs may be executed.

Elements: The following is a list of elements for the subfile. Note that elements listed in the "Virtual" section do not physically exist. There purpose is an aid to searching and system query.

```
+------------------------------------------------------------+
|                                                            |
|                Elements of Subfile INVENTORY               |
|                                                            |
|     ** Required **                                         |
|     DSN  (Key)                                             |
|     ** Optional **                                         |
|     SIZE                                                   |
|     DATE                                                   |
|     ARCH                                                   |
|     TAPE                                                   |
|     FILE                                                   |
|     STRT                                                   |
|     TITLE                                                  |
|     SUBMIT                                                 |
|     RELEASE-MONTHS                                         |
|     RELEASE-DATE                                           |
|     CLASS                                                  |
|     PERSON, AUTHOR                                         |
|     INSTITUTION, INST                                      |
|     INST-CODE, ICODE                                       |
|     KEYWORDS, KEY, K                                       |
|     ALPHA                                                  |
|       ** Optional **                                       |
|       BEGIN-DATE, BD, MINDAT                               |
|       END-DATE, MAXDAT                                     |
|       BOTTOM-LAT, MINLAT                                   |
|       TOP-LAT, MAXLAT                                      |
|       LEFT-LONG, MINLON                                    |
|       RIGHT-LONG, MAXLON                                   |
|       CONTRACT-NUMBER, PROJECT                             |
|       DATA-REF-NO, REF                                     |
|     ** Virtual **                                          |
|     LIBRARY                                                |
|     COLAT                                                  |
|     COLATB                                                 |
|     COLATMIN                                               |
|     COLATMAX                                               |
|     EALON                                                  |
|     EALONB                                                 |
|     EALONMIN                                               |
|     EALONMAX                                               |
|     DAT                                                    |
|     DATB                                                   |
|     DATMIN                                                 |
|     DATMAX                                                 |
|                                                            |
+------------------------------------------------------------+
```

Search Terms: As with the elements of subfile INVENTORY, many of the search terms exist as slaves to the query system protocols. Care should be taken if searching in native SPIRES without the aid of the query system protocol as they have been designed to be included in rather complex search expressions.

Search Terms of Subfile INVENTORY

Goal Records:	RECORD
Simple Index:	DATE
Simple Index:	SUBMIT
Simple Index:	RELEASE-DATE
Simple Index:	CLASS
Simple Index:	AUTHOR, PERSON
Simple Index:	INST, INSTITUTION
Simple Index:	ICODE, INST-CODE
Simple Index:	K, KEY, KEYWORDS, TITLE
Simple Index:	END-DATE, MAXDAT
Simple Index:	LIBRARY
Simple Index:	COLAT, COLATB
Simple Index:	COLATMIN
Simple Index:	COLATMAX
Simple Index:	EALON, EALONB
Simple Index:	EALONMIN
Simple Index:	EALONMAX
Simple Index:	DAT, DATB
Simple Index:	DATMIN
Simple Index:	DATMAX

Specifications: Note that the estimate of space used is very rough at this point. A larger sample of data will be needed to get a more realistic value.

```
+--------------------------------------------------------------+
|                                                              |
|                 Detail of Subfile INVENTORY                  |
|                                                              |
|      Owning account: USGS                                    |
|      Disk          :  USGS 200                               |
|      Filedef       :  USGS: INVENTORY                        |
|      Format(s)     :  USGS: ADDCARDS                         |
|                    :  $PROMPT                                |
|      Processing    :  Overnight by System                    |
|      Accessed-by   :  ..SETUP                                |
|                    :  ..ARCHIVE                              |
|                    :  ..CMD.INVENTORY                        |
|                    :  ..ADD                                  |
|      Space used    :  Approx 400 bytes/rec                   |
|      Overhead      :  30%                                    |
|                                                              |
+--------------------------------------------------------------+
```

Usage Notes: Subfile INVENTORY should be dealt with ONLY via protocol control unless the user understands the internal design. Minor changes in data values can be accomplished using format $PROMPT and the MERGE command. Data for each record is collected from the "sample" datasets and added with the aid of protocol ..ADD.

Sample Record:

```
+------------------------------------------------------------------+
|                                                                  |
|              Sample Record of Subfile INVENTORY                  |
|                                                                  |
|       DSN = GL000021;                                            |
|       SIZE = 12314;                                              |
|       DATE = 830810;                                             |
|       ARCH = WL;                                                 |
|       TAPE = SM9309;                                             |
|       FILE = 16;                                                 |
|       STRT = 1;                                                  |
|       TITLE = HYPOCENTER DATA FILE OF THE                        |
|      INTERNATIONAL SEISMOLOGICAL CENTRE (ISC) FOR THE            |
|      YEAR 1973;                                                  |
|       SUBMIT = 01/01/78;                                         |
|       RELEASE-MONTHS = 99;                                       |
|       RELEASE-DATE = 04/01/86;                                   |
|       CLASS = EARTHQUAKE SUMMARY;                                |
|       PERSON = ";";                                              |
|       INSTITUTION = INTERNATIONAL SEISMOLOGICAL                  |
|      CENTRE, NEWBURY RG13 1LX, BERKSHIRE, UNITED                 |
|      KINGDOM;                                                    |
|       ALPHA;                                                     |
|         BEGIN-DATE = 730101;                                     |
|         END-DATE = 731231;                                       |
|         BOTTOM-LAT = -90;                                        |
|         TOP-LAT = 90;                                            |
|         LEFT-LONG = -180;                                        |
|         RIGHT-LONG = 180;                                        |
|         DATA-REF-NO = A002;                                      |
|                                                                  |
+------------------------------------------------------------------+
```

Subfile TAPES

Purpose: The general purpose of this subfile is to keep track of both USGS-owned and temporary user tapes. Any tape which is the responsibility of the USGS would be cataloged in this file.

Elements: Element selected was designed to follow the tape log form for ease of data entry and file maintenance.

Search Terms:

Specifications:

```
+------------------------------------------------------------------+
|                                                                  |
|                     Detail of Subfile TAPES                      |
|                                                                  |
|      Owning account:  WHLW6                                      |
|      Disk          :  WHLW6 191                                  |
|      Filedef       :  WHLW6: TAPES                               |
|      Format(s)     :  $PROMPT                                    |
|      Processing    :  User responsiblity                         |
|      Accessed-by   :                                             |
|      Space used    :  Approx 66 bytes/rec                        |
|      Overhead      :  37%                                        |
|                                                                  |
+------------------------------------------------------------------+
```

Usage Notes: File has not yet been installed as a contributory part of the query system.

Subfile QUAKE PROTO

Purpose: This is a standard SPIRES protocol file. All SPIRES protocols associated with the query system are housed in this database. Execution of these protocols is performed by identification of the protocol name prefixed with two periods (..).

Elements:

```
+------------------------------------------------------------------+
|                                                                  |
|                 Elements of Subfile QUAKE PROTO                  |
|                                                                  |
|     ** Required **                                               |
|     NAME(Key)                                                    |
|     ** Optional **                                               |
|     COMMANDS, COMMAND                                            |
|     DATE-UPDATED, DA, DATE, DU, DATE-UPDATE                      |
|     TIME-UPDATED, TI                                             |
|     USER, AC                                                     |
|                                                                  |
+------------------------------------------------------------------+
```

Search Terms: This subfile does not have any searchable terms.

Specifications: Note that space used may very considerably from protocol to protocol: however, this database will not grow very large given the nature of its use.

```
+------------------------------------------------------------------+
|                                                                  |
|                  Detail of Subfile QUAKE PROTO                   |
|                                                                  |
|     Owning account:  USGS                                        |
|     Disk          :  USGS 200                                    |
|     Filedef       :  USGS: QUAKE                                 |
|     Format(s)     :  $PROTOCOL                                   |
|     Processing    :  Overnight by system                         |
|     Accessed-by   :                                              |
|     Space used    :  Approx 500 bytes/rec                        |
|     Overhead      :  20%                                         |
|                                                                  |
+------------------------------------------------------------------+
```

Usage Notes:

Part 4

PROGRAM LISTINGS

The following pages are listings of all of the program code created for the operation of the Earthquake Archiving and Retrieval System.

The programs are given in alphabetical order. Line numbers are added in the left margin for ease of reading and are not contained in the original programs.

Truncation of lines may occur here for lines which exceed the width of this report.

ADD

```
* ADD  (10/07/84, 22:51:50, USGS    )
! set noecho
CLR DYNVAR
-
- Prompts for disk modes of sample data sets to be includta base
- file.  Prompts for low and high end of library samples r update
-
IF $LSTR($TERMINAL,1) = 'G' : BLANK CRT EXTERNAL
SHOW EVAL ' '
SHOW EVAL 'This protocol adds records in the INVENTORY daile'
SHOW EVAL 'using header cards from sample datasets.  You prompted'
SHOW EVAL 'a library prefix, low and high bound dataset r
SHOW EVAL ' '
LET IDATA = $IDATA
IF $SELECT ¬= 'INVENTORY.MAINT' THEN SELECT INVENTORY.MAI
JUMP LIBRARY
++NOCONT
*
* ARCHIVE session aborted.
RETURN
-
++LIBRARY
SHO EVAL ' '
..PROMPT ARCHIVE.DMODE 'LIBRARY (SL,WL,GL CR=GL) ? '
IF #V0 = '' THEN LET V0 = 'GL'
LET LIBRARY = #V0
IF #V0 = '' THEN JUMP BADLIBRARY
IF $SIZE(#V0) ¬= 2 THEN JUMP BADLIBRARY
IF $MATCH(#V0,SL,GL,WL) > 0 THEN JUMP LOW
++BADLIBRARY
* Must provide a library name of SL, Wl or GL
JUMP LIBRARY
-
++LOW
SHO EVAL ' '
..PROMPT ARCHIVE.LOW 'Enter low range value (Cr=1) ? '
IF #V0 = '' THEN LET V0 = '1'
LET LOW = #V0
IF #V0 = '' THEN JUMP BADLOW
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * must be integer val
THEN JUMP LOW
JUMP HIGH
++BADLOW
* Must be integer value
JUMP LOW
-
++HIGH
SHO EVAL ' '
..PROMPT ARCHIVE.HIGH 'Enter high range value (Cr=999) ?
IF #V0 = '' THEN LET V0 = '999'
LET HIGH = #V0
IF #V0 = '' THEN JUMP BADHIGH
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * must be integer val
THEN JUMP HIGH
JUMP FIND.FM2
++BADHIGH
* Must be integer value
JUMP HIGH
-
++FIND.FM2
..PROMPT SETUP.FM2 'Disk mode which contains samples (CR=
IF #V0 = '' THEN LET V0 = 'H'
LET FM2 = #V0
IF $SIZE(#V0) ¬= 1 THEN * Must be 1 character
THEN JUMP FIND.FM2
IF #V0 >= 'A' THEN IF #V0 <= 'Z' THEN JUMP DOADDS
* Must be A-Z
JUMP FIND.FM2
-
++DOADDS
SET FOR ADDCARDS
++ADDLOOP
IF #LOW > #HIGH THEN JUMP ADDDONE
LET TEMP = #LIBRARY||$INSETR(#LOW,'0',6)
IF $RECTEST(#TEMP) > 0 THEN LET MODE = 'ADDUPD'
ELSE LET MODE = 'ADD'
/SET ACTIVE #TEMP SAMPLE #FM2
SET WDST LAST
IF $WDST < 0 THEN /* #TEMP NOT FOUND
THEN JUMP NEXTADD
/#MODE
IF $NO THEN /* ERROR IN #MODE OF $IDATA
/ELSE * $KEY #MODE COMPLETED
++NEXTADD
LET LOW = #LOW + 1
JUMP ADDLOOP
++ADDDONE
* ..ADD Done
/SET ACTIVE #IDATA
RETURN
```

ADD.INST.CODE

```
* ADD.INST.CODE  (09/14/83, 21:56:44, USGS    )
-
- Looks at records in the INVENTORY subfile and resolves es based
- on the institution names.
-
IF $SELECT ¬= 'INVENTORY' THEN SHOW EVAL '-INVENTORY Subfselected'
THEN RETURN
-
THRU CODES SELECT INSTITUTIONS
FOR SUB
++LOOP
REF END='JUMP DONE'
LET INST = $GETCVAL('INST',0,'')
/* #INST
SET PATH CODES
/FIND INST-NAME #INST
TYPE INST-CODE
SET PRIMARY PATH
JUMP LOOP
++DONE
IF $FORTYPE : ENDFOR
RETURN
```

APPLYUPD

```
  1.    SET EXEC NOLOG TER
  2.    SET MODE SHORT NOWARN
  3.    SET ESC &
  4.    ;
  5.    ; APPLYUPD WYLBUR - Apply an update to an archive dataset
  6.    ;
  7.    ; George Crane   - APRIL 1984
  8.    ;
  9.    IF (LAST GT 0) THEN COMM Active file NOT empty.
 10.    IF (LAST GT 0) THEN COMM Clear active and type EXEC FIRST
 11.    THEN EXEC 1000
 12.    COMM
 13.    COMM This exec will apply an update file to an ARCHIVE
 14.    COMM file dataset.  Updated file will be left in WYLBUR.
 15.    COMM
 16.    COMM Enter the filename portion only for the file to be
 17.    COMM corrected.
 18.    COMM
 19.    ;
 20.    READ STRING S1 UPPER PROMPT 'filename ? '
 21.    IF (SUBSTR(S1,1,1) EQ 'Q') THEN EXEC 1000
 22.    IF (SIZE(S1) GT 8) THEN COMM File name must be 8 charactess
 23.    IF (SIZE(s1) GT 8) THEN EXEC 23
 24.    ;
 25.    SMODE &S1 ARCHIVE * (STACK
 26.    IF (RC GT 0) THEN COMM '&S1 ARCHIVE *' file not found.
 27.    THEN EXEC 1000
 28.    READ STRING S2
 29.    SET VAL S2 SUBSTR(S2,1,2)
 30.    ;
 31.    SMODE &S1 UPDATE * (STACK
 32.    IF (RC GT 0) THEN COMM '&S1 UPDATE *' file not found.
 33.    THEN EXEC 1000
 34.    READ STRING S3
 35.    SET VAL S3 SUBSTR(S3,1,2)
 36.    ;
 37.    COMM
 38.    COMM Reading '&S1 ARCHIVE &S2' into WYLBUR active file.
 39.    USE &S1 ARCHIVE &S2 CLEAR
 40.    COMM
 41.    COMM Applying update &S1 UPDATE &S3
 42.    COP FROM &S1 UPDATE &S3 COMB REPLACE NUMBER
 43.    COMM
 44.    DELETE '$DELETE' 1 in all nol
 45.    COMM Dataset '&S1 ARCHIVE' in active file and updates app
 46.    COMM
 47.    COMM Exec done.
 48.    COMM
```

ARCHGET

```
/*-------------------------------------------------------------
/*
/* ARCHGET EXEC - George Crane, October 1983
/*
/* Executed by batch monitor to retrieve files from
/* tape and put directly onto the whlw6 198 disk
/*
/*-------------------------------------------------------------

Arg password user

/*-------------------------------------------------------------
/* Attempt to gain write access to the WHLW6 Disk
/*-------------------------------------------------------------

'CP LINK WHLW6 198 222 M 'password
If RC = 114 Then Do
   'CP MSG 'user' Unable to link to WHLW6 - Password inco
   Exit 4
   End
If RC = 104 Then Do
   'CP MSG 'user' Unable to link to WHLW6 - Other users o
   Exit 4
   End
If RC > 0 Then Do
   'CP MSG 'user' Unable to link to WHLW6 - LINK RC='RC
   Exit 4
   End
'ACCESS 222 B'

'EXECIO * DISKR RETRIEVE TAPELIST A 1 (FINIS'
IF RC > 0 Then Do
   'CP MSG 'user' Error reading RETRIEVE TAPELIST RC='RC
   Exit 4
   End

Do J = 1 to Queued()
   Parse Pull ds.j si.j da.j ta.j fi.j st.j
   End

J = J -1

'SETUP TAPE 181 'ta.1' SL NORING 6250 (END'
Tape = ta.1

Do N = 1 to J
   If tape ¬= ta.1 Then Do
      'SETUP CLEAR'
      'SETUP TAPE 181 'ta.n' SL NORING 6250 (END'
      tape = ta.n
      End

```

```
'ERASE TEMP ARCHIVE A'
'FI INMOVE TAP1 SL 'fi.n' ( DEN 6250 LRECL 80 RECFM FB BL000'
'FI OUTMOVE DISK TEMP ARCHIVE A (LRECL 80 RECFM FB BLKSIZ
'MOVEFILE'
If RC > 0 Then Do
   'CP MSG 'user' Move error on DSN='ds.n' TAPE='ta.n' FI
   Exit 4
   End

'COPYFILE TEMP ARCHIVE A 'DS.N' ARCHIVE B (FROM 'ST.N' FO REP'
message = ' Moved DSN='ds.n' From tape 'ta.n' file 'fi.n'lete'
'EXECIO 1 DISKW RETRIEVE LOG A (FINIS STRING 'message
'CP MSG 'user message
End

'SETUP CLEAR'
'CP MSG 'user' All done'
Exit 0
```

ARCHIVE

```
* ARCHIVE  (09/15/83, 09:03:44, MEOW6    )
! set noecho
CLR DYNVAR
SELECT LIBDATA
SET FOR LIBDATA
-
- Prompts for disk modes of dsnames to be archived and di
- of the staging disk (whlw6 198), then executes the rex
- called ARCHSET which creates batch exec's and submits t
-
JUMP LIBRARY
++NOCONT
*
* ARCHIVE session aborted.
RETURN
-
++LIBRARY
SHO EVAL ' '
..PROMPT ARCHIVE.DMODE 'LIBRARY (SL,WL,GL CR=GL) ? '
IF #V0 = '' THEN LET V0 = 'GL'
LET LIBRARY = #V0
IF #V0 = '' THEN JUMP BADLIBRARY
IF $SIZE(#V0) ¬= 2 THEN JUMP BADLIBRARY
IF $MATCH(#V0,SL,GL,WL) > 0 THEN JUMP DMODE
++BADLIBRARY
* Must provide a library name of SL, Wl or GL
JUMP LIBRARY
-
++DMODE
SHO EVAL ' '
..PROMPT ARCHIVE.DMODE 'Disk mode data sets currently on  '
IF #V0 = '' THEN LET V0 = 'H'
LET DMODE = #V0
IF #V0 = '' THEN JUMP BADDMODE
IF $SIZE(#V0) > 1 THEN JUMP BADDMODE
IF $SPAN(#V0,ABCDEFGHIJKLMNOPQRSTUVWXYZ) = #V0 THEN JUMP
++BADDMODE
* Must be single character mode from A-Z
JUMP DMODE
-
++GMODE
LET GMODE = 'H'
-
++SUBMIT
/ARCHSET #LIBRARY #DMODE #GMODE (
IF $RC > 0 THEN JUMP DONEERR
SHOW EVAL ' '
..PROMPT ARCHIVE.OK 'Ok to submit? (YES/NO Cr=YES) ? '
IF #V0 = '' THEN LET V0 = 'YES'
IF $PMATCH(#V0,'Y?ES',OK?) THEN JUMP SUBMITOK
IF $PMATCH(#V0,'N?O') THEN JUMP DONE
SHOW EVAL 'You must enter YES or NO or return'
```

```
JUMP SUBMIT
++SUBMITOK
/USING GETDATA DIS #LIBRARY
LET SIZE = 0
LET START = 0
LET FILENO = #FILENO + 1
/USING PUTDATA MER #LIBRARY
CMS BATCH SUBMIT (NOPROF TIME 3 SEND ARCHBAT) ARCHBAT
++DONE
*
* ARCHSET Done.
Return
++DONEERR
SHO EVAL ' '
SHO EVAL 'Error detected in ARCHSET exec'
SHO EVAL '..ARCHIVE stopped'
SHO EVAL ' '
RETURN
```

ARCHSAMP

```
/*-----------------------------------------------------------
/*
/* ARCHSAMP - Make sample datasets - George Crane 8/30/83
/*
/*-----------------------------------------------------------

Arg Dsn Datamode Sampmode '(' options

If dsn = '?' | dsn = 'HELP' Then Signal Tell

Replace = ''
library:
library = Readprom('Library; (CR=GL) ==> ')
If Library == '' Then library = 'GL'
Library = Translate(library)
If Length(library) ¬= 2 then Do
   Say 'Library name must be exactly 2 characters'
   Signal library
   End

bnum = Readprom('Library 'library' Beginning number; (CR=)
If bnum == '' Then bnum = 1

Enum = Readprom('Library 'library' Ending number; (CR=LAS)
Enum = Translate(enum)
If enum == '' Then enum = 'LAST'

Sampmode = Readprom('Mode to put samples on; (CR=H) ==> '
sampmode = translate(sampmode)
If sampmode == '' Then sampmode = 'H'

Datamode = Readprom('Mode to find datasets; (CR=H) ==> ')
Datamode = translate(datamode)
If datamode == '' Then datamode = 'H'
temp = Readprom('Ok to replace any existing samples; (CR= ')
temp = translate(temp)
If Abbrev('YES',temp,1) = 1 Then replace = temp
If Abbrev('OK',temp,1) = 1 Then replace = temp
Say 'Creating Samples on 'sampmode' from datasets on 'dat
'LISTX 'library'* ARCHIVE 'datamode '(EXEC PRE 1'
If RC > 0 Then Say 'Error..'
IF RC > 0 Then Exit RC
'EXEC CMS &STACK'
mark = 0
Do J = 1 to Queued()
   Parse pull line.j
   Parse var line.j linx.j .
   temp.j  = Substr(linx.j,3)
   If mark = 0 & temp.j >= bnum Then mark = j
End
max = j - 1
'ERASE CMS EXEC A'
```

```
k = 0
last = temp.max
If enum ¬= 'LAST' Then last = enum
num = bnum
Do j = num to last
   Do while temp.mark > j
      If j > last then leave
      Temp = library||right(j,6,'0')
      Say 'Missing 'temp
      j = j + 1
      End
   If j > last then leave
   temp = library||right(j,6,'0')
   k = k + 1
   samp.k = temp
   'LISTX 'samp.k' ARCHIVE 'datamode' (D SORT TYPE APPEND
   mark = mark + 1
   End

'CMS &STACK'
Do j = 1 to Queued()
   Parse Pull line.j
   End

max = j - 1

Do j = 1 to max
  Say line.j
  End

Do J = 1 to max
   Parse Var line.j fn ft fm recfm lrecl recs blocks date
   'EXECIO 1 DISKR 'fn ft fm recs ' (FINIS'
   Parse Pull fcard
   'EXECIO * DISKR 'fn ft fm '1 (FINIS FIND /C*DATE/ ZONE
   If Queued() = 0 Then 'EXECIO * DISKR 'fn ft fm '1 (FIN/C$DATE/
   If Queued() = 0 Then Say 'Unable to locate C$DATE cardit fm
   Else Do
      Parse Pull junk
      Parse Pull . ':'date';'months';'
      if months == '' Then months = 0
      date = right(date,8,'0')
      yy = substr(date,3,2)
      mm = substr(date,5,2)
      dd = substr(date,7,2)
      tt = months + mm
      If tt > 12 Then Do
         add = trunc(tt/12)
         mm = tt - (add*12)
         yy = yy + add
         newdate = right(mm,2,0)'/'dd'/'right(yy,2,0)
         temp = '19'right(yy,2,0)||right(mm,2,0)||dd
         If Date(sorted) >= temp Then months = 0
      End
   'EXECIO * DISKR 'fn ft fm '1 (FINIS FIND /C*END/ ZONE
   If Queued() = 0 Then Say 'Error finding C*END card on m
   Else Do
      Parse Pull end .
      Parse Pull junk
      sample = fn' SAMPLE 'sampmode
      Call moveit
      End
   End
END
Exit



Moveit:
'SET CMSTYPE HT'
'STATE 'sample
State = RC
'SET CMSTYPE RT'
If State = 0 & Replace == '' Then Do
   Say Sample' - Already Exist'
   Return
   End
If State = 0 & Replace ¬= '' Then 'ERASE 'Sample

message2 =''
last = end + 10
if last > recs then Do
   last = recs
   message = 'No data cards missing from sample'
   End
Else Do
   Missing = recs - last - 1
   Message = '***************** 'missing' data cards not re ******
   message2 = fcard
   End

Header = '*** This is a sample of data set 'fn' ***'
'EXECIO 1 DISKW 'FN' SAMPLE' SAMPMODE' 1 (FINIS STRING 'h

If months > 0 Then Do
   If months = 99 Then Do
      Header1 = '*** This data set can not be retrieved w**'
      Header2 = '***    the written permission of the aut**'
      End
   If months < 99 Then Do
      Header1 = '*** This data set can not be retrieved uwdate' **
      Header2 = '***    unless permission is obtained frothor  ***
      End
   'EXECIO 1 DISKW 'FN' SAMPLE' SAMPMODE' 2 (FINIS STRING1
   'EXECIO 1 DISKW 'FN' SAMPLE' SAMPMODE' 3 (FINIS STRING2
   End


```

```
If RC > 0 Then Do
   Say 'Error writing first record to sample 'fn' SAMPLE e
   Exit 4
   End
'COPYFILE 'fn ft fm fn' SAMPLE 'sampmode' (APPEND FROM 1 t
If RC > 0 Then Do
   Say 'Error copying to 'fn' SAMPLE 'sampmode
   Exit 4
   End
'EXECIO 1 DISKW 'fn' SAMPLE' Sampmode' (FINIS STRING 'mes
If message2 ¬= '' Then 'EXECIO 1 DISKW 'fn' SAMPLE' SampmNIS STRIN(
If RC > 0 Then Do
   Say 'Error writing end card to 'fn 'SAMPLE' Sampmode
   Exit 4
   End
If State = 0 Then Say Sample' - Replaced'
Else Say Sample' - Created'
Return


Tell:
Say ' '
Say 'Create or replace sample data sets'
Say ' '
Say 'Syntax is ARCHSAMP library datamode samplemode [ ( R
Say ' '
Say 'Where library    - is GL, WL or SL'
Say '      datamode   - is the mode which contains the da
Say '      samplemode - is the mode which samples should ed on'
Say '      REPlace    - may be specified to replace existles'
Say ' '
Exit 0
```

ARCHSET

```
/*-----------------------------------------------------------*
/*                                                           *
/* USGS - ARCHSET - Setup batch job to copy archive          *
/*                  data sets to tape                        *
/*                                                           *
/* Creates batch job using "library ARCHLIST", a file        *
/* containing all of the DSN# cards for data sets ready      *
/* to be archived.  The file is created by ..COMBINE         *
/* in SPIRES.  This exec would normally be called from       *
/* a SPIRES protocol ..ARCHIVE with three arguments as       *
/* follows:                                                  *
/*                                                           *
/*      ARCHSET Dmode Gmode [(options]                       *
/*                                                           *
/* Where:  Lib   - is the archive library name of SL,        *
/*                 WL or GL                                  *
/*                                                           *
/*         Dmode - the current mode of the disk which        *
/*                 contains the archive data sets            *
/*                 created from ..COMBINE.  In addition,*
/*                 this disk must contain the                *
/*                 "library ARCHLIST" file which list        *
/*                 the files to archive.                     *
/*                                                           *
/*         Gmode - This is the disk mode of the WHLW6        *
/*                 198 (the large staging disk) where        *
/*                 archive files are combined into tape *
/*                 files for move to tape.                   *
/*                                                           *
/* Note :  A file called "DSN ARCHIVED" will be appended*
/*                 to on the Gmode disk and will contain*
/*                 all of the data set names which were *
/*                 archived to tape and verified.            *
/*                                                           *
/*-----------------------------------------------------------*


Arg lib dmode gmode '(' options

b = 0
If lib   == '' Then Do
   Say 'Library name missing, must supply SL, GL or WL'
   Exit 4
   End
If lib ¬= 'SL' & lib¬= 'WL' & lib ¬= 'GL' Then Do
   Say 'Library must be SL, GL or NL'
   Exit 4
   End

If Dmode == '' Then Dmode = 'D'
If Gmode == '' Then Gmode = 'G'

```

```
Archlist = lib' ARCHLIST 'Dmode

'STATE 'Archlist
If RC > 0 Then Do
   Say Archlist' Not found'
   Exit RC
   End

Say 'Using file 'Archlist
Say 'Using data sets on mode 'Dmode
Say 'Final files to dump will be on mode 'Gmode
'EXECIO * DISKR 'Archlist' 1 (FINIS'
J = 0
ta.1 = ''
Do I = 1 to Queued()
   Parse Pull string
   LINE.I = STRING
   Parse Var string 'C#DSN='ds';SIZE='si';DATE='da';ARCH=E='ta';FI
   If substr(string,1,2) = 'C#' Then Do
     J = J + 1
     ds.j = ds
     si.j = si
     da.j = da
     ta.j = ta
     fi.j = fi
     st.j = st
   End
End

/* setup base values */

Size = si.1
dsn  = ds.1
file = fi.1
strt = st.1
tape = ta.1
max = j
last = 0

i = 1
If ta.1 == '' then Do
   Say ' '
   Say 'No datasets in 'archlist' need to be archived.'
   Say ' '
   Exit 0
   End
Say 'Using tape 'Ta.I
Call Copy I REP
Do J = 2 to max
   If right(ds.j,6) ¬= (right(ds.i,6)+1) Then Do
      Call error j 'Data set not 1 greater than last dataer'
      End
   If ta.j ¬= ta.i Then Do   /* if tapes not same */
      /* following code put in to prevent multiple tape s
```

```
          j = max /* finish */
          Say ' '
          Say 'Multiple tapes needed...'
          Say '  Issue ..SUBMIT'
          Say '  and then ..ARCHIVE again for next tape'
          Say ' '
          Leave
          Call setup
          Say 'Next tape 'ta.j
          Call Copy j REP
          If fi.j ¬= 1 Then Do
             Call error j 'File does not start at 1 for new t
             End
          If st.j ¬= 1 Then Do
             Call error j 'Start record number must be 1 for '
             End
          End
       Else Do
          If fi.j = fi.i Then Do    /* if files same */
          Call Copy J APPEND
          If st.j ¬= (size+strt) Then Do
             Call error J 'File start value not correct'
             End
          End
          If fi.j ¬= fi.i Then Do   /* if files different*/
             If st.j ¬= 1 Then Do
               Call error j 'Start record number must be 1 fole'
               End
             If fi.j ¬= (fi.i + 1) Then Do
               Call error j 'File increment not 1'
               End
             Call Setup
             Call Copy J REP
          End
       End
    I = i + 1
    Size = si.I
    dsn  = ds.I
    file = fi.I
    strt = st.I
    tape = ta.I
    End
    Call setup
    Call Batch

    /* mark lines in archlist file as submitted. */

    QUEUE 'COMMAND SET CMSTYPE HT'
    Queue 'COMMAND LOCATE /DSN='ds.1'/'
    QUEUE 'COMMAND SET LINEND OFF'
    QUEUE 'C/C#/##/'i
    Queue 'FILE'
    'XEDIT 'archlist' (NOPROF'
    'SET CMSTYPE RT'
```

```
Exit 0

/*-----------------------------------------------------*/
/*                                                     */
/* SETUP routine adds current tape and file number to  */
/*        an array, later to be used for building the  */
/*        batch exec job                               */
/*                                                     */
/*-----------------------------------------------------*/

Setup:

b = b + 1
batchds.b = ds.i
batchta.b = ta.i
batchfi.b = fi.i
last = i
Return

/*-------------------------------------------------------*
/*                                                       *
/* BATCH routine creates the needed batch exec lines to *
/*        'MOVEFILE' the data set from the staging disk  *
/*        to tape.  Then move it back and compare to the *
/*        original.  If they compare then the file       *
/*        'DSN ARCHIVED' on the Gmode disk is appended   *
/*        with the current dsn data.                     *
/*                                                       *
/*-------------------------------------------------------*

Batch:
Queue '/* --------- ARCHBAT EXEC ------------ */'
Queue "'SETUP TAPE 181 "batchta.1" SL RING 6250 (END'"
QUEUE "'GIME WHLW6 198 H'"
Den = 'DEN 6250'
Leave = ''
DL = '6250'
tape = batchta.1
k = 1
Do J = 1 to b
   If tape ¬= batchta.j Then Do
      QUEUE "'SET IMPEX OFF'"
      Queue "'TAPE WTM 2'"
      Queue "'TAPE REW'"
      Queue "'SET IMPEX ON'"
      Call Compare k j
      k = j
      tape = batchta.j
      Leave = ''
      Den = 'DEN 6250'
      DL = '6250'
      Queue "'SETUP CLEAR'"
      Queue "'SETUP TAPE 181 "batchta.j" SL RING 6250 (EN
      end
      Queue "'EXEC DOTAPE MOVE "batchds.j batchta.j batchfi.rid()"'"
      Queue "If RC > 0 Then Exit RC"
      Den = ''
      Leave = 'LEAVE'
      DL = 'LEAVE'
      End
   Queue "'SET IMPEX OFF'"
   Queue "'TAPE WTM 2'"
   Queue "'TAPE REW'"
   Queue "'SET IMPEX ON'"
   Call compare k b
   Queue "Exit 0"
   Queue ''
   'EXECIO * DISKW ARCHBAT EXEC A 1 (FINIS'
   Say 'ARCHBAT EXEC A has been created'
   Return

   Compare:
   arg aa bb
   Den = 'DEN 6250'
   Leave = ''
   DL = '6250'
   Do c = aa to bb
      Queue "'EXEC DOTAPE COMP "batchds.c batchta.c batchfi.rid()"'"
      Queue "If RC > 0 Then Exit RC"
      Den = ''
      Leave = 'LEAVE'
      DL = 'LEAVE'
      End
   Return


   Copy:
   Arg string
   Parse var string number type .
   inmove = ds.number' ARCHIVE 'dmode
   outmove = ta.number 'FILE'fi.number Gmode
   'COPYFILE 'inmove outmove '( 'type' TYPE'
   If RC = 0 Then Return 0
   Say ' '
   Say 'Error - Copy of 'inmove' --> 'outmove
   Say ' '
   Exit RC

   Error:
   Parse Arg String
   Parse var string number message
   Say ' '
   Say 'Error - 'message
   Say 'On record number 'j 'dataset='ds.j
   Say ' '
   Exit 4
```

BASE36

```
* BASE36  (03/06/84, 22:35:48, USGS      )
LET BASE = 36
LET DIGITS = 3
LET P = ''
LET ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
LET I1 = 1
LET DIGITS = #DIGITS - 2
LET N::0 = #BASE
++LOOP
LET I = #I1 -1
LET N::#I1 = #N::#I * #BASE
LET I1 = #I1 + 1
IF #I1 > #DIGITS THEN JUMP CALC
JUMP LOOP
++CALC
LET I1 = #DIGITS
LET NUMBER = $INT($ASK)
++LOOP2
LET I3 = #N::#I1
IF #NUMBER < #I3 THEN JUMP ZERO
LET X = $TRUNC(#NUMBER/#I3)
LET NUMBER = #NUMBER - (#X * #I3)
IF $INT(#X) > $INT(9) THEN LET X = $SUBSTR(#ALPHA,#X-10,1
LET P = #P || #X
JUMP NEXT2
++zero
LET P = #P || '0'
++NEXT2
LET I1 = #I1 - 1
IF #I1 >= 0 THEN JUMP LOOP2
LET X = #NUMBER
IF $INT(#X) > $INT(9) THEN LET X = $SUBSTR(#ALPHA,#X-10,1
LET P = #P || #X
RETURN
```

CLEANUP

```
 1.     SET EXEC NOLOG TER
 2.     SET MODE SHORT NOWARN
 3.     SET ESC &
 4.     ;
 5.     ; CLEANUP WYLBUR - Display screen of data and
 6.     ;                  prompt for line deletion.
 7.     ;
 8.     ; George Crane   - 3/22/84
 9.     ;
10.     IF (LAST EQ 0) THEN COMM Active file empty.
11.     THEN EXEC 1000
12.     ;
13.     READ STRING S1 UPPER PROMPT 'Number of lines to display ( '
14.     IF (SUBSTR(S1,1,1) EQ 'Q') THEN EXEC 1000
15.     ;
16.     SET VAL W0 FIRST
17.     ;
18.     READ STRING S3 PROMPT 'Hit return to continue'
19.     POINT &W0(&S1)
20.     SET VAL W0 * + .001
21.     ;
22.     READ STRING S2 UPPER PROMPT 'Delete lines (CR=no delete)
23.     IF (SUBSTR(S2,1,1) EQ 'Q') THEN EXEC 1000
24.     IF (SUBSTR(S2,1,1) EQ 'N') THEN EXEC 17
25.     IF (S2 EQ '') THEN EXEC 17
26.     LIST &S2
27.     READ STRING S3 UPPER PROMPT 'Ok to delete (Y/N; CR=NO) ?
28.     IF (SUBSTR(S3,1,1) EQ 'Q') THEN EXEC 1000
29.     IF (SUBSTR(S3,1,1) EQ 'Y') THEN DELETE &S2
30.     THEN COMM Lines &S2 deleted.
31.     ELSE COMM No lines deleted.
32.     ;ELSE EXEC 17
33.     ;
34.     SET VAL W3 *
35.     READ STRING S3 UPPER PROMPT 'Insert blank line (CR=NO) ?
36.     IF (SUBSTR(S3,1,1) EQ 'Q') THEN EXEC 1000
37.     IF (S3 EQ '') THEN SET VAL S3 'NO'
38.     IF (SUBSTR(S3,1,1) EQ 'Y') THEN &W3 $BLANK
39.     THEN CHANGE '$BLANK' TO '' IN &W3 NOL
40.     THEN COMM Blank line inserted as line number &W3
41.     ELSE COMM No blank inserted.
42.     EXEC 17
```

<u>CMD.INVENTORY.SEARCH</u>

```
* CMD.INVENTORY.SEARCH  (10/07/84, 23:12:54, USGS     )
---------------------------------------------------------
LET FIND = 'FIND'
SET NOSTOP
SET MES 2
- If $report already set then skip setting again.
LET TEMP = 'DSN(1,8) BD(,8) ED(,8) TITLE'
IF $SETFORMAT ¬= '$REPORT' : JUMP SETFORMAT
IF #TEMP = #SAVEINPUT : JUMP SEARCH.LIBRARY
++SETFORMAT
/SET FOR $$REPORT #TEMP
JUMP SEARCH.LIBRARY
--------------------------------------------------------
- Return here if prompt ended in attention             -
--------------------------------------------------------
++NOCONT
*
* SEARCH session aborted.
RETURN
--------------------------------------------------------
---------------------------------------------------------
++SEARCH.LIBRARY
---------------------------------------------------------
LET LABEL = 'SEARCH.LIBRARY'
..PROMPT SEARCH.LIBRARY 'LIBRARY? '
IF #NULL THEN JUMP SEARCH.DAT
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST LIBRARY
THEN JUMP SEARCH.LIBRARY
LET LIBRARY = #V0
IF $MATCH(#LIBRARY,'GL','SL','WL') = 0 THEN JUMP BADLIB
/#FIND LIBRARY #LIBRARY
IF $ZRESULT : JUMP ZRESULT
LET FIND = 'AND'
JUMP SEARCH.DAT
-
++BADLIB
SHOW EVAL '-Library must be GL SL or WL'
JUMP SEARCH.LIBRARY
-
---------------------------------------------------------
++SEARCH.DAT
---------------------------------------------------------
LET LABEL = 'SEARCH.DAT'
..PROMPT SEARCH.DAT 'DATE (BEGIN,END)? '
IF #NULL THEN JUMP SEARCH.AUTHOR
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST BD
THEN JUMP SEARCH.DAT
LET BEGIN = $BREAK(#V0,', ')
LET END = $STRIP($RSUB(#V0,#BEGIN),' ,')
IF #BEGIN = '' THEN LET BEGIN = '010101'
IF #BEGIN = '*' THEN LET BEGIN = '010101'
IF #END = '*' THEN LET END = ''
IF #END = '' THEN LET SEARCH = 'BD >= '#BEGIN
ELSE LET SEARCH = 'BD >= '#BEGIN' AND ED <= '#END
/#FIND #SEARCH
IF $ZRESULT THEN JUMP ZRESULT
LET FIND = 'AND'
-
-
----------------------------------------------------
++SEARCH.AUTHOR
----------------------------------------------------
LET LABEL = 'SEARCH.AUTHOR'
..PROMPT SEARCH.AUTHOR 'AUTHOR(S)? '
IF #NULL THEN JUMP SEARCH.INST
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST AUTHOR
THEN JUMP SEARCH.AUTHOR
LET SEARCH = $CHANGE(#V0,',',' OR ')
/#FIND AUTHOR #SEARCH
IF $ZRESULT THEN JUMP ZRESULT
LET FIND = 'AND'
-
-
----------------------------------------------------
++SEARCH.INST
----------------------------------------------------
LET LABEL = 'SEARCH.INST'
..PROMPT SEARCH.INST 'INSTITUTION(S)? '
IF #NULL THEN JUMP SEARCH.KEYWORDS
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST INST
THEN JUMP SEARCH.INST
LET SEARCH = $CHANGE(#V0,',',' OR ')
/#FIND INST #SEARCH
IF $ZRESULT THEN JUMP ZRESULT
LET FIND = 'AND'
-
----------------------------------------------------
++SEARCH.KEYWORDS
----------------------------------------------------
LET LABEL = 'SEARCH.KEYWORDS'
..PROMPT SEARCH.KEYWORDS 'KEYWORD(S)? '
IF #NULL THEN JUMP SEARCH.LAT
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST KEYWORDS
THEN JUMP SEARCH.KEYWORDS
LET SEARCH = $CHANGE(#V0,',',' OR ')
/#FIND KEYWORDS #SEARCH
IF $ZRESULT THEN JUMP ZRESULT
LET FIND = 'AND'
-
----------------------------------------------------
++SEARCH.LAT
----------------------------------------------------
LET LABEL = 'SEARCH.LAT'
LET LLFLAG = $FALSE
..PROMPT SEARCH.LAT 'LATITUDE (BOTTOM, TOP)? '
IF #NULL THEN JUMP SEARCH.LON
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST COLATMIN
THEN JUMP SEARCH.LAT
-
XEQ PROC GETVALS
IF #VAL3 : *
THEN /* Unrecognized '#VAL3', re-enter
THEN JUMP SEARCH.LAT
-
IF #VAL2 = '' : LET VAL2 = #VAL1
LET LAT1 = $SPAN(#VAL1,.0123456789)
LET VAL1 = $RSUB(#VAL1,#LAT1)
LET LAT2 = $SPAN(#VAL2,.0123456789)
LET VAL2 = $RSUB(#VAL2,#LAT2)
-
- CHECK FOR VALID VALUES
-
IF $MATCH(#VAL1,N,S) = 0 THEN JUMP NONS
IF $MATCH(#VAL2,N,S) = 0 THEN JUMP NONS
IF $REAL(#LAT1) < $REAL(-90) THEN JUMP LATRANGE
IF $REAL(#LAT1) > $REAL(90) THEN JUMP LATRANGE
IF $REAL(#LAT2) < $REAL(-90) THEN JUMP LATRANGE
IF $REAL(#LAT2) > $REAL(90) THEN JUMP LATRANGE
IF #VAL1 = S THEN LET LAT1 = #LAT1 * (-1)
IF #VAL2 = S THEN LET LAT2 = #LAT2 * (-1)
IF $REAL(#LAT1) = 90 THEN IF $REAL(#LAT2) = -90 THEN JUMP
-
- LEGAL VALUES, FIND MIN AND MAX COLATITUTE
-
LET LLFLAG = $TRUE
IF $REAL(#LAT1) = -90 THEN IF $REAL(#LAT2) = 90 THEN JUMPD
LET LAT1 = 90 - #LAT1
LET LAT2 = 90 - #LAT2
IF $REAL(#LAT1) > $REAL(#LAT2) THEN LET TEMP = #LAT1
THEN LET LAT1 = #LAT2
THEN LET LAT2 = #TEMP
JUMP END.SEARCH.LAT
-
++LATWORLD
-
LET LAT1 = 0
LET LAT2 = 180
JUMP END.SEARCH.LAT
-
++NONS
*
* Latitude must have N or S designation
JUMP SEARCH.LAT
-
++NULLNS
*
* Latitude produces null set
JUMP SEARCH.LAT
-
++LATRANGE
```

```
*
* Latitude value out of range, must be 0-90 north or sout
JUMP SEARCH.LAT
-
++DEFLAT
-
LET LAT1 = 0
LET LAT2 = 180
*   Default 90s 90n used.
JUMP END.SEARCH.LAT
-
++END.SEARCH.LAT
-
- LAT1 = COLAT MIN
- LAT2 = COLAT MAX
LET SLAT = '(COLAT>='#LAT1' AND <='#LAT2')'
LET SLAT = #SLAT' OR (COLATMIN<'#LAT1' AND COLATMAX >'#LA
/#FIND (#SLAT)
IF $ZRESULT : JUMP ZRESULT
LET FIND = 'AND'
-
JUMP SEARCH.LON
-
--------------------------------------------------------
++SEARCH.LON
--------------------------------------------------------
LET LABEL = 'SEARCH.LON'
..PROMPT SEARCH.LON 'LONGITUDE (LEFTSIDE, RIGHTSIDE)? '
IF #NULL THEN JUMP SEARCH.DONE
IF $PMATCH(#V0,BRO?WSE) THEN BROWSE FIRST EALONMIN
THEN JUMP SEARCH.LON
-
XEQ PROC GETVALS
IF #VAL3 THEN * TOO MANY VALUES
THEN JUMP SEARCH.LON
-
IF #VAL2 = '' : LET VAL2 = #VAL1
LET LON1 = $SPAN(#VAL1,.0123456789)
LET VAL1 = $RSUB(#VAL1,#LON1)
LET LON2 = $SPAN(#VAL2,.0123456789)
LET VAL2 = $RSUB(#VAL2,#LON2)
-
- CHECK FOR VALID VALUES
-
IF $MATCH(#VAL1,E,W) = 0 THEN JUMP NOEW
IF $MATCH(#VAL2,E,W) = 0 THEN JUMP NOEW
IF #VAL1 = W THEN LET LON1 = #LON1 * (-1)
IF #VAL2 = W THEN LET LON2 = #LON2 * (-1)
IF $REAL(#LON1) < $REAL(-180) THEN JUMP LONRANGE
IF $REAL(#LON1) > $REAL(180) THEN JUMP LONRANGE
IF $REAL(#LON2) < $REAL(-180) THEN JUMP LONRANGE
IF $REAL(#LON2) > $REAL(180) THEN JUMP LONRANGE
IF $REAL(#LON1) = 180 THEN IF $REAL(#LON2) = -180 THEN JUW
-
```

```
- ALL VALUES LEGAL
-
IF $REAL(#LON1) = -180 THEN IF $REAL(#LON2) = 180 THEN JURLD
IF $REAL(#LON1) < 0 THEN LET LON1 = 360 + #LON1
IF $REAL(#LON2) < 0 THEN LET LON2 = 360 + #LON2
IF $REAL(#LON1) > $REAL(#LON2) THEN LET TEMP = #LON1
THEN LET LON1 = #LON2
THEN LET LON2 = #TEMP
JUMP NOSPLIT
-
++NULLEW
*
* Longitude values produce null set
JUMP SEARCH.LON
-
++NOEW
*
* Must specify E or W
JUMP SEARCH.LON
-
++LONWORLD
LET LON1 = 0
LET LON2 = 360
JUMP NOSPLIT
-
++LONRANGE
*
* Longitude value out of range, must be 0-180 east or wes
JUMP SEARCH.LON
-
++DEFLON
-
LET LON1 = 0
LET LON2 = 360
*   Default 180e 180w used.
-
++NOSPLIT
-
LET SLON = '(EALON>='#LON1' AND <='#LON2') OR '
LET SLON = #SLON '(EALONMIN<'#LON1' AND EALONMAX>'#LON2')
JUMP END.SEARCH.LON
-
++END.SEARCH.LON
/#FIND (#SLON)
IF $ZRESULT : JUMP ZRESULT
LET FIND = 'AND'
-
++SEARCH.DONE
SET MES 0
SEQ
IF $STACK > 0 THEN SHOW EVAL $STACK' records found'
ELSE SHOW EVAL 'No records found'
THEN IN ACT CLN CLR TYPE
If $EDITOR = 'WYLBUR' Then SHOW EVAL 'Records in WYLBUR ale'
```

```
Else SHOW EVAL 'Records in '$ODATA
If $LIST = $FALSE THEN SHOW EVAL 'Type LIST to display'
RETURN
-
++CONT
*
ASK UPP PROMPT 'Ok to continue (YES/NO) ? ' ATTN='RETURN'
IF $PMATCH($ASK,'Y?ES',OK) : RETURN
ELSE IF $PMATCH($ASK,'N?OP') : /RETURN JUMP #CONTLEV
*
PLEASE ANSWER YES OR NO
XEQ PROC CONT
-
++HELP
IF $ASK ¬= '?' : IF $PMATCH($ASK,'H?ELP') = 0 : RETURN
/..INVENTORY.HELP #HELP
/RETURN JUMP #LABEL
++DONE
RETURN
-
------------------------------------------------------------
++ZRESULT
------------------------------------------------------------
SHOW EVAL ' '
-* Search produced zero result
/JUMP #LABEL
-
-
++GETVALS
LET TEMP = ' '$ASK' '
LET TEMP = $CHANGE(#TEMP,',',' ')
LET TEMP = $CHANGE(#TEMP,' AND ',' ')
LET TEMP = $CHANGE(#TEMP,' BETWEEN ',' ')
LET TEMP = $CHANGE(#TEMP,' TO ',' ')
LET TEMP = $CHANGE(#TEMP,' FROM ',' ')
++SQUEEZE
IF $MATCH(#TEMP,'?  ?') = 0 : JUMP SQUDONE
LET TEMP = $CHANGE(#TEMP,'  ',' ')
JUMP SQUEEZE
++SQUDONE
LET TEMP = $CHANGE(#TEMP,' N','N')
LET TEMP = $CHANGE(#TEMP,' S','S')
LET TEMP = $CHANGE(#TEMP,' W','W')
LET TEMP = $CHANGE(#TEMP,' E','E')
LET TEMP = $STRIP(#TEMP,' ')
LET VAL1 = $BREAK(#TEMP,' ')
LET TEMP = $STRIP($RSUB(#TEMP,' '),' ')
LET VAL2 = $BREAK(#TEMP,' ')
LET TEMP = $STRIP($RSUB(#TEMP,' '),' ')
LET VAL3 =$BREAK(#TEMP,' ')
RETURN
```

CMD.INVENTORY

```
* CMD.INVENTORY  (10/02/84, 22:42:48, USGS    )
! SET NOECHO
SET NOSTOP
SET MES 0
SET FOR DISP1
LET SHORT = $FALSE
LET CONTLEV = 'ASK.LOOP'
IF $LSTR($TERMINAL,1) = 'G' THEN BLANK CRT EXTERNAL
SET ACTIVE
-
SHOW EVAL 'INVENTORY program -- if in trouble type HELP'
-
IF $UPPER = 0 : *  Upper case set
IF $UPPER = 0 : SET UPPER
++ASK.LOOP
IF #SHORT = $FALSE : ..INV.MENU1
IF $UPPER : LET CASE = '?'
IF $ELSE : LET CASE = '>'
IF $FORTYPE > 0 : LET PROMPT = '+'#CASE' '
IF $ELSE : LET PROMPT = '-'#CASE' '
-
SET MES 0
/ASK PROMPT '#PROMPT' ATTN = 'JUMP WHAT'
LET TEMP = $CAP($BREAK($ASK,' '))
IF $ASK = '?' THEN JUMP HELPIT
IF $PMATCH(#TEMP,'H?ELP') : JUMP HELPIT
IF $PMATCH($CAP($ASK),'SHO?RT') : JUMP SETSHORT
IF $PMATCH($CAP($ASK),'LON?G') : JUMP SETLONG
IF $PMATCH($CAP($ASK),'SEA?RCH') : JUMP SEARCH
IF $PMATCH($CAP($ASK),'LOC?ATE') : JUMP LOCATE
IF $PMATCH($CAP($ASK),'SEN?D') : JUMP SEND
++PASS
SET MES 2
/$ASK
SET MES 0
JUMP ASK.LOOP
-
++SETSHORT
LET SHORT = $TRUE
JUMP ASK.LOOP
-
++SETLONG
LET SHORT = $FALSE
JUMP ASK.LOOP
-
-
++HELPIT
..INV.MENU1.HELP
JUMP ASK.LOOP
-
++SEARCH
..CMD.INVENTORY.SEARCH
```

```
JUMP ASK.LOOP
-
++LOCATE
++DISPLAY
++SEND
SHOW EVAL ' '
SHOW EVAL 'command unavailable - not yet installed.'
SHOW EVAL ' '
JUMP ASK.LOOP
++RETURN
CLEAR SELECT
SET MES 0
SET STOP
SHOW EVAL ' '
SHOW EVAL 'Exiting INVENTORY program.'
SHOW EVAL ' '
RETURN
-
++CONT
SHOW EVAL ' '
ASK UPP PROMPT 'Ok to continue (YES/NO) ? ' ATTN='RETURN'
IF $PMATCH($ASK,'Y?ES',OK) : RETURN
ELSE IF $PMATCH($ASK,'N?OP') : /RETURN JUMP #CONTLEV
SHOW EVAL ' '
PLEASE ANSWER YES OR NO
XEQ PROC CONT
-
++HELP
IF $ASK ¬= '?' : IF $PMATCH($ASK,'H?ELP') = 0 : RETURN
/XEQ PROC #HELP
/RETURN JUMP #LABEL
++DONE
RETURN
```

COMBINE

```
* COMBINE  (03/20/84, 08:35:56, USGS    )
!SET NOECHO
-
- Combine files
- Creates C# card and collects index, comment and datafil
-
- George Crane 7/4/83
-
IF $EDITOR ¬= 'WYLBUR' THEN * Must be in WYLBUR - Command
THEN RETURN
SET MODE SHORT NOWARN
SET LENGTH 80
,SET LENGTH 80
SELECT LIBDATA
SET FORMAT LIBDATA
LET ARCHIVIST = 'WL'
JUMP PICK.ARCH
------------------------------------------------------
- Return here if prompt ended in attention           -
------------------------------------------------------
++NOCONT
*
* COMBINE session aborted.
CLR XEQS
RETURN
-----------------------------------------------------
++PICK.ARCH
/..PROMPT COMBINE.ARCHIVIST 'Archivist (CR=#ARCHIVIST)'
IF #V0 = '' THEN LET V0 = #ARCHIVIST
LET ARCHIVIST = #V0
IF $SIZE(#V0) ¬= 2 THEN * Archivist must be exactly 2 cha
THEN JUMP PICK.ARCH
-
++PICK.LIBRARY
SHO EVAL ' '
..PROMPT COMBINE.LIBRARY 'Library (GL/SL/WL,CR=GL) '
IF #V0 = '' THEN LET V0 = 'GL'
LET LIBRARY = #V0
IF $MATCH(#LIBRARY,GL,SL,WL) > 0 THEN JUMP FIND.FM0
/* '#LIBRARY' is an invalid library name.
JUMP PICK.LIBRARY
-
++FIND.FM0
SHOW EVAL ' '
..PROMPT SETUP.FM0 'Input disk mode for index and comment '
IF #V0 = '' THEN LET V0 = 'B'
LET FM0 = #V0
IF $SIZE(#V0) ¬= 1 THEN * Must be 1 character
THEN JUMP FIND.FM0
IF #V0 >= 'A' THEN IF #V0 <= 'Z' THEN JUMP FIND.FM
* Must be A-Z
JUMP FIND.FM0
```

```
-
-
++FIND.FM
SHOW EVAL ' '
..PROMPT SETUP.FM 'Input disk mode for datafile? (CR=H) '
IF #V0 = '' THEN LET V0 = 'H'
LET FM = #V0
IF $SIZE(#V0) ¬= 1 THEN * Must be 1 character
THEN JUMP FIND.FM
IF #V0 >= 'A' THEN IF #V0 <= 'Z' THEN JUMP FIND.FM2
* Must be A-Z
JUMP FIND.FM
-
++FIND.FM2
SHOW EVAL ' '
..PROMPT SETUP.FM2 'Output disk mode? (CR=H) '
IF #V0 = '' THEN LET V0 = 'H'
LET FM2 = #V0
IF $SIZE(#V0) ¬= 1 THEN * Must be 1 character
THEN JUMP FIND.FM2
IF #V0 >= 'A' THEN IF #V0 <= 'Z' THEN JUMP FIND.FN
* Must be A-Z
JUMP FIND.FM2
-
-----------------------------------------------------
- Find out the default file name                    -
-----------------------------------------------------
++FIND.FN
SHOW EVAL ' '
..PROMPT SETUP.FN          'File name ?          '
IF #V0 = '' : * You must supply a value
THEN JUMP FIND.FN
IF $SIZE(#V0) > 8 THEN /* '#V0' greater than 8 characters
THEN JUMP FIND.FN
LET FN = #V0
++CHECKFN
LET TEMP = 'COMMENT '#FM0
XEQ PROC CHECKFILE
LET TEMP = 'INDEX '#FM0
XEQ PROC CHECKFILE
LET TEMP = 'DATAFILE '#FM
XEQ PROC CHECKFILE
++CHECK.ARCHIVE
/CMS SET CMSTYPE HT
/STATE #FN ARCHIVE #FM
LET RC = $RC
/CMS SET CMSTYPE RT
IF #RC > 0 THEN JUMP GETDATA
/* '#FN ARCHIVE #FM' Already exist
..PROMPT COMBINE.ARCHIVE 'Now What? (ERASE/Redo/Pause,CR=
IF #V0 = '' THEN JUMP FIND.FN
IF $PMATCH(#V0,'R?EDO') THEN JUMP FIND.FN
IF #V0 = 'ERASE' THEN /ERASE #FN ARCHIVE #FM
THEN /* '#FN ARCHIVE #FM' Erased
THEN JUMP GETDATA
IF $PMATCH(#V0,'P?AUSE') THEN BREAK XEQ
THEN JUMP FIND.FN
/* '#V0' Invalid response
JUMP CHECK.ARCHIVE
-
++CHECKFILE
/CMS SET CMSTYPE HT
/STATE #FN #TEMP
LET RC = $RC
/CMS SET CMSTYPE RT
IF #RC = 0 THEN RETURN
/* '#FN #TEMP' not found.
SHOW EVAL ' '
..PROMPT COMBINE.MISSING 'Now What? (Redo/Pause,CR=Redo)
IF #V0 = '' THEN JUMP FIND.FN
IF $PMATCH(#V0,'R?EDO') THEN RETURN FIND.FN
IF $PMATCH(#V0,'P?AUSE') THEN BREAK XEQ
THEN RETURN JUMP CHECKFN
/* '#V0' invalid response
JUMP CHECKFILE
-
-
++GETDATA
SET NOSTOP
/USING GETDATA DIS #LIBRARY
IF $NO THEN /* problem fetching #library data...  program
THEN RETURN
SET STOP
-
++GETITEMS
/CMS LISTFILE #FN DATAFILE #FM (D EXEC
/CMS EXECIO 1 DISKR CMS EXEC A 1 (MARGINS 45 52
ASK
LET DATALEN = $ASK
/USE #FN INDEX #FM0 CLEAR
/COPY FROM #FN COMMENT #FM0
-/COPY FROM #FN DATAFILE #FM
SET LENGTH 80
,SET LENGTH 80
NUM
SET WDSR LAST
WDSR
IF $LSTR($ASK,5) = 'C*END' THEN JUMP ENDOK
SHOW EVAL 'C*END Card missing -- It will be added'
WDSE C*END----------------------------------------------------------
NUM
++ENDOK
SET WDST LAST
LET ITEM = $WDST
LET ITEM = #ITEM + #DATALEN
LET ITEM = #ITEM + 2
LET DSN = #DSN + 1
LET START = #START + #SIZE
LET SIZE = #ITEM
IF #START = 0 THEN LET START = 1
LET CUMSIZE = #CUMSIZE + #ITEM
-
SHO EVAL ' '
IF #CUMSIZE < 1875000 THEN JUMP CHECKREC
-NEW TAPE
LET FILENO = 1
LET START = 1
LET CUMSIZE = #ITEM
SHOW EVAL ' '
/* Tape #TAPEVOL full
++COMBINE.TAPEVOL
..PROMPT COMBINE.TAPEVOL 'Next tape name '
IF #V0 = '' THEN * You must supply a tape name
THEN JUMP COMBINE.TAPEVOL
IF $SIZE(#V0) ¬= 6 THEN * Tapename must be 6 characters
THEN JUMP COMBINE.TAPEVOL
LET TAPEVOL = #V0
JUMP MAKE.HEADER
-
++CHECKREC
LET I2 = #START + #ITEM
IF #I2 < 12000 THEN JUMP MAKE.HEADER
IF #START = 1 THEN JUMP MAKE.HEADER
LET FILENO = #FILENO + 1
LET START = 1
++MAKE.HEADER
IF #FILENO = 0 THEN LET FILENO = 1
LET DSNAME = #LIBRARY||$INSETR(#DSN,'0',6)
LET TEMP = 'C#DSN='#DSNAME';SIZE='
LET TEMP = #TEMP||$INSETR(#ITEM,'0',6)';DATE='$CHANGE($DA')
LET TEMP = #TEMP||';ARCH='#ARCHIVIST';TAPE='#TAPEVOL';FIL
LET TEMP = #TEMP||$INSETR(#FILENO,'0',3)';STRT='$INSETR(#',6)';'
SET MODE NOWARN SHORT
/.5 #TEMP
LET ENDTEMP = 'C#FINIS DSN='#LIBRARY||$INSETR(#DSN,'0',6)
-/END #ENDTEMP
SHOW EVAL ' '
LIST .5 UNN
++COMBINE.FINAL
SHOW EVAL ' '
..PROMPT COMBINE.FINAL 'Ok to save? (Pause/No/Save,CR=Sav
IF #V0 = '' THEN LET V0 = 'SAVE'
IF $PMATCH(#V0,'S?AVE') THEN JUMP OK.TO.SAVE
IF $PMATCH(#V0,'P?AUSE') THEN BREAK XEQ
IF $PMATCH(#V0,'N?O') THEN * SAVE ABORTED.. PROGRAM STOPP
THEN RETURN
IF $PMATCH(#V0,'O?K','Y?ES') THEN JUMP OK.TO.SAVE
/* '#V0' INVALID RESPONSE
JUMP COMBINE.FINAL
++OK.TO.SAVE
SHOW EVAL ' '
/SAVE #DSNAME ARCHIVE #FM2
```

```
IF $RC > 0 THEN * SAVE ERROR
THEN BREAK XEQ
IF #LIBRARY ¬= 'SL' THEN JUMP MAKE.HEADER2
/..BASE36 #DSN
LET P = 'S'||#P
/* BASE36 DSN VALUE IS = #P
++ASK.BASE36
SHOW EVAL ' '
..PROMPT COMBINE.BASE36 'Ok to insert base36? (Yes/No,CR=
IF #V0 = '' THEN LET V0 = 'YES'
IF $PMATCH(#V0,'Y?ES') THEN JUMP OK.TO.BASE36
IF $PMATCH(#V0,'N?O') THEN JUMP MAKE.HEADER2
/* '#V0' INVALID RESPONSE
JUMP ASK.BASE36
++OK.TO.BASE36
/USE #FN DATAFILE #FM CLEAR
SET LEN 130
/CHA 1/4 TO '#P' IN F/L NOL
SAVE * REPLACE
++MAKE.HEADER2
/CMS COPYFILE #FN DATAFILE #FM #DSNAME ARCHIVE #FM2 (APPE
IF $RC > 0 THEN * COPY ERROR
THEN BREAK XEQ
/CMS EXECIO 1 DISKW #DSNAME ARCHIVE #FM2 0 (FINIS STRING
IF $RC > 0 THEN * ERROR INSERTING LAST LINE
THEN BREAK XEQ
/USING PUTDATA MER #LIBRARY
SHOW EVAL ' '
SHOW EVAL 'COMBINE DONE'
/* #DSN #FILENO #START #SIZE #CUMSIZE
/SET ACTIVE #LIBRARY ARCHLIST #FM2
/WDSE #TEMP #FN
SET ACTIVE WYLBUR
JUMP FIND.FN
```

CUMSIZE

```
 1.    SET ESC &
 2.    SET EXEC NOLOG TER
 3.    SET VAL NO 0
 4.    SET VAL WO 0
 5.    SET VAL WO WO + 1
 6.    IF (WO GT LAST) EXEC 10
 7.    REA STR S1 USING &WO COL 21/26
 8.    SET VAL NO S1 + NO
 9.    EXEC 5
10.    COMM
11.    COMM CUMSIZE = &NO
12.    COMM
```

DOTAPE

```
/*-------------------------------------*/
/* dotape exec - George Crane 8/22/83 */
/*-------------------------------------*/

Arg String
Den = ''
Leave = ''
Parse var string command ds ta fi temp user .
If Temp = '6250' Then Den = 'DEN 6250'
If Temp = 'LEAVE' Then Leave = 'LEAVE'


If command = 'MOVE' then Signal Move
If command = 'COMP' Then Signal Compare
Say 'Invalid command 'command
Exit 100

Move:
'FI INMOVE DISK 'ta 'FILE'fi' H'
'FI OUTMOVE TAP1 SL 'fi' ( 'den leave' LRECL 80 RECFM FB 32000'
'MOVEFILE'
If RC > 0 Then Signal badmove
message = '-Move dsn='ds' tape='ta' file='fi' complete..
'EXECIO 1 DISKW ARCHIVE LOG A (FINIS STRING 'message
Exit 0

Compare:
'FI INMOVE TAP1 SL 'fi' ( 'den leave' LRECL 80 RECFM FB B2000'
'FI OUTMOVE DISK 'fi' TEMP A ( LRECL 80 BLKSIZE 32000 REC
'MOVEFILE'
If RC > 0 Then Signal badmove2
'DIFFER 'fi' TEMP A 'ta 'FILE'fi' H'
If RC > 0 Then Signal badcompare
message = '-Comp dsn='ds' tape='ta' file='fi' complete..
'CP MSG 'user message
'EXECIO 1 DISKW ARCHIVE LOG A (FINIS STRING 'message
'ERASE 'fi' TEMP A'
message = 'Dsn 'ds' on tape 'ta' file 'fi' moved and comp
'EXECIO 1 DISKW GOOD ARCHIVE A (FINIS STRING 'message
Exit 0

Badmove:
message = '-Error in move->tape dsn='ds' tape='ta' file='
'EXECIO 1 DISKW ARCHIVE LOG A (FINIS STRING 'message
Exit 1

Badmove2:
message = '-Error in move->disk dsn='ds' tape='ta' file='
'EXECIO 1 DISKW ARCHIVE LOG A (FINIS STRING 'message
Exit 2

Badcompare:
```

```
message = '-Error in compare dsn='ds' tape='ta' file='fi
'EXECIO 1 DISKW ARCHIVE LOG A (FINIS STRING 'message
Exit 3
```

INVENTORY.HELP

```
* INVENTORY.HELP  (10/02/84, 14:11:22, USGS    )
-
SET NOSTOP
XEQ PROC HELP
SET STOP
RETURN
-
++HELP
SET XSTOP
/XEQ PROC $ASK
RETURN
-
++SEARCH.DAT
-
- (1) "DATE (BEGIN,END)?"
*     This prompt allows you to find data sets with time at
* falls within a time period specified by you.  Enter theng
* and the ending date of your desired time period.  The t
* should be separated by a ",", and in the order of year,
* and day.  For example, March 21, 1983 should be entered
* "830321", or "19830321".  A "*" may be used to default r
* the earliest or the latest date; e.g., "*, 830321" willta
* sets with time index that fall on or before March 21, 1
RETURN
-
++SEARCH.LAT
-
- (2) "LATITUDE (BOTTOM, TOP)?"
*     This prompt allow you to find data sets that fall w
* specified area bounded by a latitude interval.  Enter tm
* (or the southernmost) latitude and the top (or northern
* latitude of your desired area.  The two latitude limitsbe
* separated by a ",", and each latitude must have either
* northern hemisphere) or S (for southern hemisphere) folhe
* latitude value in units of degrees.  For example, if yo
* interested in all data sets that may contain informatio
* Brazil, you may enter "34.0S, 4.0N".
RETURN
-
++SEARCH.LON
-
*(3) "LONGITUDE (LEFTSIDE, RIGHTSIDE)?"
*    This prompt allows you to find data sets that fall w
*specified area bounded by a longitude interval.  Enter t
*leftside longitude and the rightside longitude of your d
*area.  The two longitude limits should be separated by ad
*each longitude must have either E (for eastern hemispher
*(for western hemisphere) following the longitude value i
*of degrees.  For example, if you are interested in all d
*that may contain information about Brazil, you may enter
*35.5W".
RETURN
```

```
-
++SEARCH.FOC
-
- (4) "FOCAL DEPTH (MINIMUM, MAXIMUM)?"
*     This prompt allows you to find data sets with focal
* index that fall within a specified focal depth interval
* the minimum and the maximum focal depth of interest.  T
* focal depth limits should be separated by a ",", and arin
* units of integer kilometers.  For example, if you are id
* in data sets that may contain information about of eart
* with focal depth between 5 and 15 km, you enter "5, 15"s
* prompt.
RETURN
-
++SEARCH:MAG
-
- (5) "MAGNITUDE (MINIMUM, MAXIMUM)?"
*     This prompt allows you to find data sets with magniex
* that fall within a specified magnitude interval.  Enter
* minimum and the maximum magnitude of interest.  The twode
* limits should be separated by a ",".  For example, if y
* interested in data sets that may contain information ab
* earthquakes with magnitude between 3.5 and 9.0, you may
* "3.5, 9.0" for this prompt.
RETURN
-
++SEARCH.TYP
-
- (6) "DATA TYPE?"
*     This prompt allows you to search for different typea.
* your answer may be one of the following:
*
*         phase   -- if you desire to retrieve phase data,
*                    arrival times, amplitudes, signal dura
*                    etc. of earthquakes.
*         program -- if you desire to retrieve computer pro
*         station -- if you desire to retrieve station data
*                    coordinates of stations, etc.
*         summary -- if you desire to retrieve summay data,
*                    origin time, hypocenter coordinates, m,
*                    etc. of earthquakes.
RETURN
-
++SEARCH.NAM
-
- (7) "DATA SET NAME?"
*     This prompt allows you to find a data set with a gi
* set name.  Unless you know the data set name given by tr,
* it is best to skip this prompt by hitting the Return ker
* terminal.
RETURN
-
++SEARCH.AUTHOR
-
```

```
- (8) "AUTHOR NAME?"
*     This prompt allows you to find data sets by a given
* For example, if you enter "j.p. eaton", then all data s
* authored by J. P. Eaton will be found.  Please note thahe
* first author of the data set is indexed.
RETURN
-
++SEARCH.INST
-
- (9) "INSTITUTION NAME?"
*     This prompt allows you to find data sets by a given
* institution.  For simplicity in indexing, we have used
* following abbreviation:
*
*        CIRES-- CIRES, University of Colorado, Boulder.
*        CIT  -- California Institute of Technology, Pasad
*        ERI  -- Earthquake Research Institute, Univ. of Tkyo.
*        IPE  -- Institute of Physics of the Earth, Moscow
*        ISC  -- International Seismological Center.
*        LAMONT--Lamont Geological Observatory, Palisades.
*        MIT  -- Massachusetts Institute of Technology, Ca
*        NEIS -- National Earthquake Information Service,
*        NOAA -- National Oceanic and Atmospheric Admin.,
*        UCB  -- University of California, Berkeley.
*        UCLA -- University of California, Los Angeles.
*        UCSB -- University of California, Santa Barbara.
*        UCSC -- University of California, Santa Cruz.
*        UCSD -- University of California, San Diego.
*        UNR  -- University of Nevada, Reno.
*        USC  -- University of Southern California, Los An
*        USMP -- U.S. Geological Survey, Menlo Park.
*        UU   -- University of Utah, Salt Lake City.
*        UW   -- University of Washington, Seattle.
RETURN
-
++SEARCH.KEYWORDS
-
- (10) "KEYWORD?"
*     This prompt allows you to search data sets which ha
* coded with keywords.  Typical keywords are geographic r
* such as "California", and locality name, such as "Parki
* etc.
RETURN
-
++SEARCH.LIBRARY
-
- "LIBRARY?"
*     This prompt allows you to enter one of the inventor
* archive librarys such as GL (General Library), SL (Stan
* library), or WL (Waveform library).
RETURN
```

LIBDATA

```
* LIBDATA  (01/03/84, 23:37:08, USGS    )
!SET NOECHO
-
- Subfile LIBDATA maintenance protocol.
- Used to update or change fields in the libdata subfile
- as file number, tape etc.
-
- George Crane 1/1/84
-
SELECT LIBDATA
SET FORMAT LIBDATA
JUMP PICK.LIBRARY
-------------------------------------------------------
- Return here if prompt ended in attention            -
-------------------------------------------------------
++NOCONT
*
* LIBDATA session aborted.
CLR XEQS
RETURN
------------------------------------------------------
++PICK.LIBRARY
-
SHO EVAL ' '
* This protocol is used to maintain or alter data in a sp
* LIBDATA subfile entry.
*
SHO EVAL ' '
..PROMPT COMBINE.LIBRARY 'Library (GL/SL/WL,CR=GL) '
IF #V0 = '' THEN LET V0 = 'GL'
LET LIBRARY = #V0
IF $SIZE(#LIBRARY) = 2 THEN JUMP SHOW.DATA
/* '#LIBRARY' is an invalid library name, must be 2 chara
JUMP PICK.LIBRARY
++SHOW.DATA
SET NOSTOP
IF $RECTEST(#LIBRARY) > 0 THEN GOTO DATA.MERGE
SHOW EVAL ' '
SHOW EVAL 'Library '#LIBRARY' will be added as a new entr
SHOW EVAL ' '
LET TAPEVOL = 'unknown'
LET FILENO = 0
LET START = 0
LET SIZE = 0
LET CUMSIZE = 0
LET DSN = 0
JUMP SHOWDATA
-
++DATA.MERGE
SHOW EVAL ' '
/USING GETDATA DIS #LIBRARY
IF $NO THEN /* Problem fetching #library data... program
```

```
THEN RETURN
SET STOP
++SHOWDATA
XEQ PROC SHOWVAL
++TAPENAME
SHOW EVAL ' '
/..PROMPT LIBDATA.TAPE 'Enter new tape name (CR=#tapevol)
IF #V0 = '' THEN LET V0 = #TAPEVOL
LET TAPEVOL = #V0
-
++FILENO
SHOW EVAL ' '
/..PROMPT LIBDATA.FILE 'Enter new file number (CR=#fileno
IF #V0 = '' THEN LET V0 = #FILENO
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * Error, must be intee.
THEN JUMP FILENO
LET FILENO = #V0
IF FILENO = 0 THEN LET START = 0
THEN LET SIZE = 0
-
++START
/..PROMPT LIBDATA.START 'Enter file start number (CR=#sta
IF #V0 = '' THEN LET V0 = #START
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * Error, must be intee.
THEN JUMP START
-
++SIZE
/..PROMPT LIBDATA.SIZE 'Enter file size (CR=#size) '
IF #V0 = '' THEN LET V0 = #SIZE
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * Error, must be intee.
THEN JUMP START
LET SIZE = #V0
-
++CUMSIZE
SHOW EVAL ' '
/..PROMPT LIBDATA.CUMSIZE 'Enter tape cum size (CR=#cumsi
IF #V0 = '' THEN LET V0 = #CUMSIZE
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * Error, must be intee.
THEN JUMP CUMSIZE
LET CUMSIZE = #V0
-
++DSN
SHOW EVAL ' '
/..PROMPT LIBDATA.DSN 'Enter dataset number (CR=#dsn) '
IF #V0 = '' THEN LET V0 = #DSN
IF $TYPETEST(#V0,INT) ¬= 'INT' THEN * Error, must be intee.
THEN JUMP CUMSIZE
LET DSN = #V0
-
XEQ PROC SHOWVAL
++ASKOK
SHOW EVAL ' '
..PROMPT LIBDATA.FINAL 'Ok to update? (No/Yes/Redo,CR=NO)
IF #V0 = '' THEN LET V0 = 'NO'
IF $PMATCH(#V0,'Y?ES','OK?') THEN JUMP ASKYES
IF $PMATCH(#V0,'N?O','Q?UIT',QQ?UIT,S?TOP) THEN * Exiting
THEN RETURN
IF $PMATCH(#V0,'R?EDO') THEN JUMP PICK.LIBRARY
/* #v0 is an invalid input.. try again
JUMP ASKOK
-
++ASKYES
IF $RECTEST(#LIBRARY) < 1 THEN JUMP ASKYES.ADD
/USING PUTDATA MERGE #LIBRARY
IF $NO THEN /* Problem updating libdata info... program s
THEN RETURN
SHOW EVAL 'Libdata done .. value updated.'
Return
-
++ASKYES.ADD
/USING ADDDATA ADD
IF $NO THEN /* Problem adding libdata info... program sto
THEN RETURN
SHOW EVAL 'Libdata done .. value added.'
Return
-
RETURN
-
++SHOWVAL
LET TT0 =          'Tape name.... '#TAPEVOL
LET TT1 = $INSETL('File number.. '#FILENO,' ',23)
LET TT2 = $INSETL('Start rec#... '#START,' ',23)
LET TT3 = $INSETL('File size.... '#SIZE,' ',23)
LET TT4 = $INSETL('Cum size..... '#CUMSIZE,' ',23)
LET TT5 = $INSETL('Dataset#..... '#DSN,' ',23)
/* Current data for #library is:
/*    #TT0
/*    #TT1 -- Last file used (0 if new tape)
/*    #TT2 -- Last starting rec # in current file (0 if n
/*    #TT3 -- # 80 byte rec's in current file (0 if new)
/*    #TT4 -- Total # records on current tape (0 if new)
/*    #TT5 -- Total # data sets on tape (0 if new)
RETURN
```

MAKEUPD

```
 1.     SET EXEC NOLOG TER
 2.     SET MODE SHORT NOWARN
 3.     SET ESC &
 4.     ;
 5.     ; MAKEUPD WYLBUR - Make a update for a specific file
 6.     ;
 7.     ; George Crane   - APRIL 1984
 8.     ;
 9.     IF (LAST GT 0) THEN COMM Active file NOT empty.
10.     IF (LAST GT 0) THEN COMM Clear active and type EXEC FIRST
11.     THEN EXEC 1000
12.     COMM
13.     COMM This exec will retrieve specific lines from an ARCHI
14.     COMM file dataset, allow you to modify, and save as an
15.     COMM update file on the G disk.
16.     COMM
17.     COMM Enter the filename portion only for the file to be
18.     COMM corrected.
19.     COMM
20.     ;
21.     READ STRING S1 UPPER PROMPT 'filename ? '
22.     IF (SUBSTR(S1,1,1) EQ 'Q') THEN EXEC 1000
23.     IF (SIZE(S1) GT 8) THEN COMM File name must be 8 charactess
24.     IF (SIZE(s1) GT 8) THEN EXEC 21
25.     ;
26.     SMODE &S1 ARCHIVE * (STACK
27.     IF (RC GT 0) THEN COMM File not found.
28.     THEN EXEC 1000
29.     READ STRING S2
30.     SET VAL S2 SUBSTR(S2,1,2)
31.     ;
32.     COMM
33.     COMM Enter wylbur line range(s) which will make up the up
34.     COMM When you have entered all of the line ranges then en
35.     COMM the word END.
36.     COMM
37.     ;
38.     READ STRING S3 UPPER PROMPT 'Line range ? '
39.     IF (S3 EQ '') THEN EXEC 38
40.     IF (S3 EQ 'END') THEN EXEC 43
41.     COP &S3 FROM &S1 ARCHIVE &S2 COMB REP
42.     EXEC 38
43.     ;
44.     COMM
45.     COMM All lines are now in the active file with original l
46.     COMM numbers.  Make the changes needed and type EXEC to
47.     COMM continue and save the update.  For deleted lines
48.     COMM enter the value $DELETE.
49.     COMM
50.     ;
51.     EXEC PAUSE
52.     ;
```

```
COMM
COMM NOW SAVING &S1 UPDATE G
COMM
SAVE &S1 UPDATE G REP DEC LRECL 90
COMM
COMM EXEC DONE.
COMM
```

PROMPT

```
* PROMPT  (08/11/84, 18:30:38, USGS    )
-
- Prompt and return values in V0, V1 and V2.
- Entry with two operands :
-
-     ..prompt helpproto 'prompt value '
-
- where 'helpproto' is the name of a protocol which conta
-       the help message for this prompt.
-
-       'prompt value' is the prompt message to be displa
-
- George Crane  3/28/83
-
LET NULL = $FALSE
LET V0 = ''
LET V1 = ''
LET V2 = ''
LET HELP = $BREAK($ASK,' ')
LET PROMPT = $EVAL($RSUB($ASK,' '))
-
++PROMPT
/ASK UPP PRO '#PROMPT' NULL='JUMP NULLIN' ATTN='XEQ PROC
XEQ PROC HELP
LET V0 = $ASK
LET V1 = $BREAK($ASK,',')
LET V2 = $STRIP($RSUB($ASK,','),' ')
IF $PMATCH($ASK,'QUI?T') THEN RETURN JUMP NOCONT
RETURN
-
-
++HELP
IF $ASK ¬= '?' THEN IF $PMATCH($ASK,'HEL?P') = 0 THEN RET
/..INVENTORY.HELP #HELP
/RETURN JUMP PROMPT
-
++NULLIN
LET NULL = $TRUE
RETURN
-
++CONT
*
ASK UPP PROMPT 'OK TO CONTINUE (YES/NO) ? ' ATTN='RETURN'
IF $PMATCH($ASK,'Y?ES','OK') THEN RETURN
ELSE IF $PMATCH($ASK,'N?OP') THEN /RETURN RETURN JUMP NOC
*
* PLEASE ANSWER YES OR NO
JUMP CONT
-
```

REARCHIVE

```
* REARCHIVE  (08/17/84, 14:16:10, USGS    )
!SET NOECHO
-
- Re-archives datasets
- Alters C# card and assigns new tape and file number to
-
- George Crane 7/31/84
-
LET ODATA = $ODATA
LET IDATA = $IDATA
SELECT LIBDATA
SET FORMAT LIBDATA
LET ARCHIVIST = 'WL'
SHOW EVAL ' '
JUMP PICK.ARCH
--------------------------------------------------------
- Return here if prompt ended in attention             -
--------------------------------------------------------
++NOCONT
*
* REARCHIVE Session aborted.
/SET ACTIVE #ODATA (ODATA
/SET ACTIVE #IDATA (IDATA
CLR XEQS
RETURN
-------------------------------------------------------
++PICK.ARCH
/..PROMPT COMBINE.ARCHIVIST 'Archivist (CR=#ARCHIVIST)'
IF #V0 = '' THEN LET V0 = #ARCHIVIST
IF $SIZE(#V0) ¬= 2 THEN * Archivist must be exactly 2 cha
THEN JUMP PICK.ARCH
LET ARCHIVIST = #V0
-
++FIND.FM2
..PROMPT SETUP.FM2 'Output staging disk mode? (CR=H) '
IF #V0 = '' THEN LET V0 = 'H'
LET FM2 = #V0
IF $SIZE(#V0) ¬= 1 THEN * Must be 1 character
THEN JUMP FIND.FM2
IF #V0 >= 'A' THEN IF #V0 <= 'Z' THEN JUMP FIND.FN
* Must be A-Z
JUMP FIND.FM2
-
-------------------------------------------------------
- Find out the default file name                      -
-------------------------------------------------------
++FIND.FN
..PROMPT SETUP.FN          'File name ?          '
IF #V0 = '' : * You must supply a value
THEN JUMP FIND.FN
IF $PMATCH(#V0,'QUI?T','END?') : JUMP NOCONT
IF $SIZE(#V0) > 8 THEN /* '#V0' greater than 8 characters
```

```
THEN JUMP FIND.FN
LET FN = #V0
-
LET LIBRARY = $LSTR(#FN,2)
IF $MATCH(#LIBRARY,SL,GL,WL) = 0 THEN * File prefix must L or WL
THEN JUMP FIND.FN
-
++GETDATA
CMS SET CMSTYPE HT
/SMODE * * #FM2
LET RC = $RC
CMS SET CMSTYPE RT
IF #RC > 0 THEN /* No files on disk mode #FM2 .. Somthing
THEN JUMP FIND.FM2
ASK
LET TEMP = $BREAK($XSTR($ASK,KEEP,LAST,3),' ')
IF #TEMP ¬= 'RW' THEN /* Disk mode #FM2 is not in write m
THEN JUMP FIND.FM2
CMS SET CMSTYPE HT
/SMODE #FN ARCHIVE #FM2
LET RC = $RC
CMS SET CMSTYPE RT
IF #RC > 0 THEN /* File '#FN ARCHIVE #FM2' not found
THEN JUMP FIND.FN
ASK
SET NOSTOP
/USING GETDATA DIS #LIBRARY
IF $NO THEN /* problem fetching #library data...  program
THEN RETURN
SET STOP
-
++GETITEMS
/SET ACTIVE #FN ARCHIVE #FM2
LET FIRSTLINE = 1
SET WDSR FIRST
WDSR
IF $SUBSTR($ASK,0,6) = 'C#DSN=' Then LET FIRSTLINE = 2
SET WDSR LAST
WDSR
LET LASTLINE = $WDSR
IF $SUBSTR($ASK,0,7) = 'C#FINIS' THEN LET LASTLINE = $WDS
++ENDOK
SET WDST LAST
LET ITEM = #LASTLINE - #FIRSTLINE  + 1
LET ITEM = #ITEM + 2
-LET DSN = #DSN + 1
LET START = #START + #SIZE
LET SIZE = #ITEM
IF #START = 0 THEN LET START = 1
LET CUMSIZE = #CUMSIZE + #ITEM
-
SHO EVAL ' '
IF #CUMSIZE < 1875000 THEN JUMP CHECKREC
-NEW TAPE
```

```
LET FILENO = 1
LET START = 1
LET CUMSIZE = #ITEM
SHOW EVAL ' '
/* Tape #TAPEVOL full
++COMBINE.TAPEVOL
..PROMPT COMBINE.TAPEVOL 'Next tape name '
IF #V0 = '' THEN * You must supply a tape name
THEN JUMP COMBINE.TAPEVOL
IF $SIZE(#V0) -= 6 THEN * Tapename must be 6 characters
THEN JUMP COMBINE.TAPEVOL
LET TAPEVOL = #V0
JUMP MAKE.HEADER
-
++CHECKREC
LET I2 = #START + #ITEM
IF #I2 < 12000 THEN JUMP MAKE.HEADER
IF #START = 1 THEN JUMP MAKE.HEADER
LET FILENO = #FILENO + 1
LET START = 1
++MAKE.HEADER
IF #FILENO = 0 THEN LET FILENO = 1
LET TEMP = 'C#DSN='#FN';SIZE='
LET TEMP = #TEMP||$INSETR(#ITEM,'0',6)';DATE='$CHANGE($DA')
LET TEMP = #TEMP||';ARCH='#ARCHIVIST';TAPE='#TAPEVOL';FIL
LET TEMP = #TEMP||$INSETR(#FILENO,'0',3)';STRT='$INSETR(#',6)';'
CMS SET CMSTYPE HT
/ERASE #FN CMSUT1 #FM2
CMS SET CMSTYPE RT
/SET ACTIVE #FN CMSUT1 #FM2
/WDSW #TEMP
LET ENDTEMP = 'C#FINIS DSN='#FN
LET TEMP2 = #FN' ARCHIVE '#FM2
/CMS COPYFILE #TEMP2 #FN CMSUT1 #FM2 (APPEND FROM #FIRSTL#ITEM
IF $RC > 0 THEN * Error in copyfile... process aborted
THEN JUMP NOCONT
-/WDSE #ENDTEMP
/CMS COPYFILE #FN CMSUT1 #FM2 (LRECL 80 RECFM F
SET LEN 80
SHOW EVAL #TEMP
++COMBINE.FINAL
SHOW EVAL ' '
..PROMPT COMBINE.FINAL 'Ok to save? (Pause/No/Save,CR=Sav
IF #V0 = '' THEN LET V0 = 'SAVE'
IF $PMATCH(#V0,'S?AVE') THEN JUMP OK.TO.SAVE
IF $PMATCH(#V0,'P?AUSE') THEN BREAK XEQ
IF $PMATCH(#V0,'N?O') THEN JUMP NOCONT
IF $PMATCH(#V0,'O?K','Y?ES') THEN JUMP OK.TO.SAVE
/* '#V0' Invalid response
JUMP COMBINE.FINAL
++OK.TO.SAVE
CMS SET CMSTYPE HT
/ERASE TEMPFILE ARCHIVE #FM2
/RENAME #FN ARCHIVE #FM2 TEMPFILE ARCHIVE #FM2
```

```
/RENAME #FN CMSUT1 #FM2 #FN ARCHIVE #FM2
CMS SET CMSTYPE RT
/SHOW EVAL 'Replaced: '#FN' 'ARCHIVE' '#FM2
/USING PUTDATA MER #LIBRARY
/SHOW EVAL 'Updated : SPIRES Libdata'
CMS SET CMSTYPE HT
/EXECIO * DISKR #LIBRARY ARCHLIST #FM2 (FINIS LOCATE /DSN
CMS SENTRIES
LET RC = $RC
CMS SET CMSTYPE RT
IF #RC > 1 THEN ASK
THEN LET LINE1 = $ASK
THEN ASK
THEN LET LINE2 = $ASK
LET LINE1 = $RSUB(#LINE1,' ')
LET ITEM = #LINE1 - 1
LET S1 = $SUBSTR(#LINE2,81,10)
/SET ACTIVE #LIBRARY ARCHLIST #FM2
/WDSE #TEMP #S1
/SHOW EVAL 'Updated : '#LIBRARY' 'ARCHLIST' '#FM2' Update
++ANOTHER
SHOW EVAL ' '
..PROMPT SETUP.FN          'Another one?  (CR=NO) '
IF #V0 = '' : JUMP DONE
IF #V0 = 'NO' : JUMP DONE
IF $PMATCH(#V0,'Y?ES','O?K') : JUMP FIND.FN
SHOW EVAL 'Answer YES or NO'
JUMP ANOTHER
-
++DONE
/SET ACTIVE #ODATA (ODATA
/SET ACTIVE #IDATA (IDATA
SHOW EVAL ' '
SHOW EVAL 'REARCHIVE Finished.'
RETURN
```

REPTSAMP

```
/*-------------------------------------------------------
/*
/* REPTSAMP Generate script input for sample report
/*           George Crane 3/26/84
/*
/*-------------------------------------------------------

/* Identify the dsn prefix to generate report
/*    disk mode where samples can be found
/*    and file name of generate script input report

Arg Dsn

If dsn = '?' | dsn = 'HELP' Then Signal Tell

Replace = ''
library:
library = Readprom('Library; (CR=GL) ==> ')
If Library == '' Then library = 'GL'
Library = Translate(library)
If Length(library) ¬= 2 then Do
   Say 'Library name must be exactly 2 characters'
   Signal library
   End

Bnum = Readprom('Library 'library' Beginning number; (CR=)
If Bnum == '' Then Bnum = 1

Enum = Readprom('Library 'library' Ending number; (CR=LAS)
Enum = Translate(enum)
If enum == '' Then enum = 'LAST'

Sampmode = Readprom('Mode samples are on; (CR=A) ==> ')
sampmode = translate(sampmode)
If sampmode == '' Then sampmode = 'A'

Report = Readprom('File name for script output; (CR=REPTS ')
Report = Translate(report)
If report == '' Then report = 'REPTSAMP'

Datamode = Readprom('File mode to place 'report' SCRIPT; => ')
Datamode = Translate(Datamode)
If Datamode == '' Then Datamode = 'A'


Say 'Creating Report for 'library' from samples on mode '
'LISTX 'library'* SAMPLE 'sampmode '(EXEC PRE 1'
'EXEC CMS &STACK'
mark = 0
Do J = 1 to Queued()
   Parse Pull line.j .
   temp.j  = Substr(line.j,3)
```

```
      If mark = 0 & temp.j >= bnum Then mark = j
    End
    max = j - 1
    k = 0
    last = temp.max
    If enum ¬= 'LAST' Then last = enum
    num = bnum
    Do j = num to last
      Do while temp.mark > j
        Temp = library||right(j,6,'0')
        Say 'Missing 'temp
        j = j + 1
        End
      temp = library||right(j,6,'0')
      k = k + 1
      samp.k = temp
      mark = mark + 1
      End

    Say ' '
    'SET CMSTYPE HT'
    'ERASE REP1$TMP FILE A'
    'ERASE REP2$TMP FILE A'
    'STATE 'report ' SCRIPT A'
    State = RC
    If state = 0 Then Do
      'SET CMSTYPE RT'
      Say report' SCRIPT Already Exist'
      Answer = Readprom('Ok to ERASE; (CR=NO) ==> ')
      If answer == '' Then Answer = 'NO'
      If Find('OK O Y YE YES',answer) = 0 Then 'ERASE 'reporT A'
      Else Exit 4
      End
    'SET CMSTYPE RT'

    Do j = 1 to k
      fn = samp.j
      ft = 'SAMPLE'
      fm = sampmode
      'EXECIO * DISKR 'fn ft fm '1 (FINIS FIND /C#DSN/ ZONE
      first = 1
      If queued() > 0 Then parse pull first .
      If queued() > 0 Then parse Pull .
      'EXECIO * DISKR 'fn ft fm '1 (FINIS FIND /C*FORMAT/ ZO
      If Queued() = 0 Then Say 'Unable to locate C*FORMAT can ft fm
      Else Do
         Say 'Processing dsn='fn' ...'
         Parse Pull forstart .
         Parse Pull .
         'EXECIO * DISKR 'fn ft fm '1 (FINIS FIND /C*END---/ '
         forend = ''
         If Queued() > 0 Then Parse Pull forend .
         If Queued() > 0 Then Parse Pull .
         If forend ¬= '' Then temp = forend - forstart + 1
        'COPY 'fn ft fm 'REP2$TMP FILE A (REP FROM 'forstartemp
        Queue '.pa;.fo no;.ce Table 'fn
        Queue '.sp 2'
        Queue '.bt //- % -/'fn
        Queue ''
        'EXECIO * DISKW 'report' SCRIPT A 0 (FINIS '
        temp2 = forstart - first
        'COPY 'fn ft fm report 'SCRIPT A (FROM 'first' FOR 'PPEND'
        'SET CMSTYPE HT'
        'COMPARE REP1$TMP FILE A REP2$TMP FILE A'
        return = RC
        'SET CMSTYPE RT'
        If Return > 0 Then Do
           'COPY REP2$TMP FILE A REP1$TMP FILE A (REP'
           'COPY REP2$TMP FILE A 'report 'SCRIPT A (APPEND'
           backup = fn
           End
        Else Do
           Queue ' '
           Queue '.ce ***************************'
           Queue '.ce See previous format from dataset 'backdetails'
           Queue '.ce ***************************'
           Queue ' '
           Queue ''
           'EXECIO * DISKW 'report 'SCRIPT A 0 (FINIS'
           End
        End
        temp2 = forend + 1
        'COPY 'fn ft fm report 'SCRIPT A (FROM 'temp2' APPEN
      End
   Exit

   Tell:
   Say ' '
   Say 'Create or replace script input report for samples'
   Say ' '
   Say 'Syntax is REPORT'
   Say ' '
   Say ' '
   Exit 0
```

RETRIEVE

```
/*-------------------------------------------------------------
/*
/* RETRIEVE EXEC - George Crane, September 1983
/*
/* This exec will set up and submit a batch monitor
/* request to retrieve specific datasets from the
/* USGS archive tapes.
/*
/*                         +
/* Syntax:  RETRIEVE | fn ft [ fm ]          |
/*                   | SUBMIT pswd           |
/*                         +
/*
/* Where :  fn ft fm is a file which contains records of
/*          form contained in a 'library ARCHLIST'
/*          file.
/*
/*          Option SUBMIT is used to submit the batch job
/*          exec created from a previous RETRIEVE fn ft..
/*          request.  'pswd' is the write password for th
/*          staging disk.
/*
/* names
/* used  :  fn ft fm   - file name of ARCHLIST file
/*          string     - input parms
/*          data       - temporary
/*          rest       - temporary
/*          junk       - temporary
/*          j,n        - working integers
/*          tsize      - calculated total recs to move
/*
/*          label      - staging disk label name
/*          mode       - staging disk access mode
/*          rw         - staging disk r/w status
/*          blocking   - staging disk bytes/block
/*          files      - staging disk # files
/*          blks       - staging disk # blocks
/*          pc         - staging disk % full
/*          bleft      - staging disk # blocks free
/*          tblks      - staging disk total # of blocks
/*
/*          ds         - array, data set names
/*          si         - array, data set size # rec's
/*          da         - array, data set date
/*          ta         - array, data set tape name
/*          fi         - array, data set tape file
/*          st         - array, data set starting rec#
/*          ex         - array, data set exist=1
/*
/*-------------------------------------------------------------

Arg string '(' options

library = 'GL'
Do z = 1 to Words(options)
   ww = word(options,Z)
   If Abbrev('LIBRARY',ww,3) = 1 Then Do
     library = word(options,Z+1)
     z = z + 1
     End
   End

Parse Upper Var string cmd1 pswd
If Abbrev('SUBMIT',cmd1,3) = 1 Then Signal submit

/*-----------------------------------------------------
/* Make sure the file we want exist, find out which
/* disk mode its on etc.
/*-----------------------------------------------------

Parse Upper Var string fn ft fm .

If fn == '' Then Do
   If setrlist('A 'library) > 0 Then Signal abort
   fn = 'RETRIEVE'
   ft = 'LIST'
   fm = 'A'
   End

If fn == '' Then Do
   Say 'File name missing'
   Exit 0
   End
If ft == '' Then Do
   Say 'File type missing'
   Exit 0
   End
If fm == '' Then fm = '*'
'LISTFILE 'fn ft fm '(EXEC'
If RC > 0 Then Do   /* missing file       */
   Say 'File 'fn ft fm 'not found'
   Exit 0
   End
'MAKEBUF'
'EXEC CMS &STACK'
Parse Upper Pull fn ft fm
'DROPBUF'
fm = substr(fm,1,1)
Say 'Using 'fn ft fm

/*-----------------------------------------------------
/* Find the WHLW6 disk.  Look at all of the currently
/* accessed disks and see if the staging disk label
/* is accessed anywhere.  If not then do a GIME and
/* remember the mode.
/*-----------------------------------------------------
```

```

Collect:

data = ''
'Q DISK (STACK'
Do I = 1 to Queued()
   Parse Upper Pull label rest
   If label = 'WHL198' Then data = rest
End

If data == '' Then Do
   'EXEC GIME WHLW6 198 (STACK'
   Parse Upper Pull mode junk
   'Q DISK 'mode' (STACK'
   Parse Upper Pull junk
   Parse Upper Pull label data
End

Parse Upper var data . mode rw . . blocking files blks'-' tblks

/*---------------------------------------------------------
/* Collect information from each record in the file.
/* and place in arrays so we can use them later.
/*---------------------------------------------------------
'EXECIO * DISKR ' fn ft fm ' 1 (FINIS'
j = 0
Do i = 1 to Queued()
   Parse Upper Pull string
   line.i = string
   Parse Upper Var string .'DSN=' ds ';SIZE=' si ';DATE='
               'ARCH=' ar ';TAPE=' ta ';FILE=' fi ';STRT=
   If left(string,1) ¬= "*" Then Do
   J = J + 1
   ds.j = ds
   si.j = si
   da.j = da
   ta.j = ta
   fi.j = fi
   st.j = st
   ex.j = 0
   'STATE ' ds.j ' ARCHIVE ' mode
   If RC = 0 Then ex.j = 1      /* file exist        */
   End
End

/*---------------------------------------------------------
/* Look at the size of each file to be retrieved,
/* if does not exist on the staging disk (ex=0) then
/* add its size to a cumulative value (Tsize).
/*---------------------------------------------------------
num = 0
tsize = 0
Do n = 1 to j
   If ex.n = 0 Then Tsize = tsize + si.n
```

```
      If ex.n = 0 Then num = num + 1
   End

   Say 'Total Data sets requested     = 'j
   Say 'Total Data sets to retrieve   = 'num
   If num = 0 Then Do
      Say ' '
      Say 'All needed data sets exist on the staging disk'
      Say 'No retrieve from tape needed'
      Say ' '
      Exit 0
      End

   Say 'Total number of 80 byte recs  = 'tsize
   blocks = (80 * tsize)/ blocking
   blocks = Trunc(blocks) + 100
   Say 'Total number of blocks needed = 'blocks
   Say 'Total number of blocks free   = 'bleft

   /*-----------------------------------------------------
   /* If the number of blocks needed on the staging disk is
   /* greater than the number we have then call upon RETERAS
   /* to solve the problem.
   /*-----------------------------------------------------

   If blocks > bleft Then Do
      If reterase(mode blocks bleft) > 0 Then signal abort
      Signal Collect
      End

   /*-----------------------------------------------------
   /* Create a file called RETRIEVE TAPELIST A which contain
   /* a list of files and datasets to be retrieved.  The for
   /* of the file will be:
   /*
   /*   dsn size date tape fileno startno
   /*-----------------------------------------------------
   Do n = 1 to j
      If ex.n = 0 Then Do              /* if retrieve requir
         Queue ds.n si.n da.n ta.n fi.n st.n
         End
      End
   Queue ''
   'ERASE RETRIEVE TAPELIST A'
   'EXECIO * DISKW RETRIEVE TAPELIST A 0 (FINIS'

   Queue 'ARCHGET EXEC'
   Queue 'RETRIEVE TAPELIST'
   Queue ''
   'EXECIO * DISKW RETRIEVE BATCH A 1 (FINIS'
   Say ' '
   Say 'Ready for batch submit, issue the following command:
   Say ' '
   Say '  RETRIEVE SUBMIT pswd'
```

```
Say ' '
Say '  Where pswd = write password for WHLW6 198'
Say '  A batch monitor job will then be sumitted based ons'
Say '  execution of RETRIEVE which created the batch exec
Say ' '
Say 'All done'
Exit 0

Submit:
If pswd == '' Then Do
   Say 'Error, password missing.. syntax is RETRIEVE SUBMord'
   Exit
   End

If chklink('WHLW6 198') > 0 Then Do
   Say ' '
   Say 'Retrieve will not work until all WRITE links are .'
   Say 'The following list of user(s) have links to the sisk'
   Say linklist
   Say 'Retrieve submit stopped.'
   Exit 0
   End

'CP .LINK WHLW6 198 199 W ' pswd
If RC > 0 Then Do
   Say ' '
   Say 'Error - Cannot link WHLW6 198 with the given pass
   Say 'Retrieve submit stopped.'
   Exit 0
   End
'CP .DET 199'

Say ' '
Say 'Submitting:'
Say '  BATCH SUBMIT (NOPROF TIME 3 SEND RETRIEVE) ARCHGETserid()
Say ' '
'BATCH SUBMIT (NOPROF TIME 3 SEND RETRIEVE) ARCHGET 'pswd)
Say ' '
Exit RC

/*-----------------------------------------------------------
/*
/* CHKLINK ( userid vaddr )
/*
/* Returns 0=no links to vaddr owned by userid
/*         >0=link still exist ( n/3 = number of links )
/*
/*-----------------------------------------------------------

Chklink: Procedure Expose linklist
Arg id addr

/* link mini-disk to a known address to allow a query */

```

```
'CP .LINK ' id addr ' 199 RR'
'CPSTACK Q LINKS 199'

/* Collect multiple line response in a single rex variabl

string2 = ''
Do J = 1 to Queued()
   Parse Upper Pull line
   string2 = string2||line' '
   End
k = Words(string2)
count = k
linklist = ''

/* Release and detach any links we have */

Do j = 1 to k by 3
   string3 = Subword(string2,j,3)
   Parse Upper Var string3 id vaddr rw
   If id = userid() Then Do
      'CP .REL ' vaddr ' (DET'
      If RC > 0 Then 'CP .DET 'vaddr
      If RC = 0 Then Count = Count - 3
      End
  Else if rw ¬= 'R/W' Then count = count - 3
   Else linklist = linklist||string3' '
   End

linklist = Space(linklist)
Return count

Abort:
Say ' '
Say 'Processing aborted'
Say ' '
Exit 4

/*----------------------------------------------------
/*
/* SETRLIST - Create a new RETRIEVE LIST for retrieving f
/*
/* Arguments: mode    = Mode of the staging disk
/*
/*----------------------------------------------------

setrlist: Procedure

Parse Upper Arg mode library .

/* Find the ARCHLIST for the current library working on

'SMODE ' library ' ARCHLIST *'
If RC > 0 Then Do
   Say "File '" library " ARCHLIST * ' not found"
```

```
      Return 4
   End
   Parse Pull archmode .

   /* If the retrieve list already exist on the A disk
   /* then get rid of it and tell the users whats happening.

   'SMODE RETRIEVE LIST A'
   If RC = 0 Then Do
      Say "Erasing 'RETRIEVE LIST A'"
      Say ' '
      Parse Pull .
      'ERASE RETRIEVE LIST A'
   End

   /* Prompt for low and high bound list numbers and format
   /* array called retrdsn

   max = 0

   input:

   Do Forever
     Say 'Dsn1 Dsn2 ? '
     Parse Upper Pull dsn1 dsn2 .
     If Abbrev('QUIT',dsn1,1) = 1 Then Signal process
     If dsn2 == '' then dsn2 = dsn1
     If length(dsn1) < 3 | length(dsn2) < 3 Then Do
       Say 'dsn names must be at least three characters'
       Say 'two character library name + integer'
       Signal input
     End
     prefix = Substr(dsn1,1,2)
     If Substr(dsn2,1,2) ¬= prefix Then Do
       Say 'Prefix value for dsn1 does not match dsn2'
       Signal input
     End
     no1 = Substr(dsn1,3,6)
     no2 = Substr(dsn2,3,6)
     Do I = no1 to no2
       max = max + 1
       num = right(i,6,'0')
       retrdsn.max = prefix || num
       index.max = 0
     End i
   End

   Process:

   /* Read in the ARCHLIST data set and compare with the lis
   /* needed files collected above.  If a needed file is in
   /* archlist then does not exist on a backup tape.

   'EXECIO * DISKR ' library ' ARCHLIST ' archmode ' 1 (FINI
If RC > 0 Then Do
  Say 'Error reading ARCHLIST file'
  Return 4
End

setflag = 0
Do k = 1 to Queued()
  Parse Upper Pull line
  archdsn = Substr(line,7,8)
  Do i = 1 to max
    If archdsn = retrdsn.i Then Do
      'EXECIO 1 DISKW RETRIEVE LIST A 0 (FINIS STRING 'li
      index.i = 1
      setflag = 1
      i = max
    End
  End i
End k

If setflag = 1 Then Say 'RETRIEVE LIST A created.'
Else Do
  Say 'All requested data set names not found in archlist
  Say 'RETRIEVE LIST A has not been created.'
  Return 4
End

/* Report on files needed but not found in archlist

setflag = 0
Do i = 1 to max
  If index.i = 0 Then Say 'Entry for ' retrdsn.i ' does n ' ,
                         'in 'library ' ARCHLIST 'archmo
  Then setflag = 1
End i

setok:
If setflag = 1 Then Do
  Say 'Ok to continue ? '
  Parse Upper Pull answer .
  If Abbrev('YES',answer,1) = 1 Then Return 0
  If Abbrev('OK',answer,1)  = 1 Then Return 0
  If Abbrev('NO',answer,1)  = 1 Then Return 4
  Say 'Please answer YES or NO'
  Signal setok
End

Return 0

/*--------------------------------------------------------
/*
/* RETERASE - Erase enough space to allow retrieve of arc
/*            data sets from tape.
/*
/* Arguments: mode    = current mode of staging disk
/*               blkneed = number of free blocks needed
/*               blocks  = number of blocks used on disk
/*
/*----------------------------------------------------------

reterase: Procedure

Parse Upper Arg mode blkneed blocks .

'SMODE RETRIEVE LIST *'
If RC > 0 Then Do
  Say "File 'RETRIEVE LIST *' not found"
  Return 4
End
Parse Upper Pull rmode .
retlist = 'RETRIEVE LIST ' rmode

start:

/* fill array blk and date with current info about all ar
/* files on the staging disk

'LISTX * ARCHIVE' MODE '(SORT D DATE EXEC'
'EXEC CMS &STACK'
Do j = 1 to Queued()
   Parse Pull fname . . . . . blk date .
   fname.j = fname
   blk.j = blk
   date.j = date
End j
jmax = j -1

/* Fill array dsn with the list of data sets to be retrie
/* so we don't end up erasing something we need

'EXECIO * DISKR ' retlist ' 1 (FINIS'
Do k = 1 to Queued()
   Parse Pull .'DSN=' dsn.k ';'
End k
kmax = k-1

/* Now look throught the list of archive files on the sta
/* disk and find the ones that can be erased.  Ask if its
/* to erase and go on to next until enough space has been
/* realized or user can't erase anymore.

Do j = 1 to jmax
   flag = 0
   Do k = 1 to kmax
      If fname.j = dsn.k Then flag = 1
   End k
   If flag = 0 Then Do
      Say 'File' fname.j 'has' blk.j 'blocks, ok to erase
      Parse Upper Pull answer
```

```
             If answer = 'OK' Then Do
                'ERASE' fname.j 'ARCHIVE' mode
                blocks = blocks - blk.j
                If blocks < 0 Then j = jmax
                End
             End
        End j

      /* Having gone through the list of files, now we can chec
      /* the score.  May want to go back to start and not colle
      /* the $200 for passing go.

      If blocks < 0 Then Return 0
      Say 'Not enough erased, still need' blocks 'blocks'
      Say 'Ok to repeat?'
      Parse Upper Pull Answer
      If answer = 'OK' Then Signal start
      Return 4
```

```
/*************************************************************
/*
/*  RTAPE - Read tape data to disk.
/*
/*           George Crane 1/24/84
/*           Modified from $READTAP EXEC
/*           Reads a non-CMS tape.
/*
/*************************************************************

Parse Upper Arg type listname junk '(' options

/*-----------------------------------------------------------
/* Check to see if no arguments or ? were entered.
/*-----------------------------------------------------------

Select;
   When type == '' Then Signal Tell
   When type = '?' Then Signal Tell
   When Abbrev('HELP',type,1) = 1 Then Signal Tell
   When Abbrev('SUBMIT',type,1) = 1 Then Signal Submit
   When Abbrev('PROMPT',type,1) = 1 Then Signal Prompt
   When Abbrev('REVIEW',type,1) = 1 Then Signal Review
   When Abbrev('READ',type,1) = 1 Then Signal Read
   Otherwise Do
      Say 'Unknown parameter 'type
      Say 'Expecting HELP, SUBMIT, PROMPT, REVIEW or READ
      Exit 0
      End
End

/*-----------------------------------------------------------
/* Prompt for info to create TAPCNTRL file
/*-----------------------------------------------------------

Prompt:
Say ' '
Asktape:
Tapename = Readprom('Tape name ==> ')
tapename = translate(tapename)
If tapename = 'QUIT' Then Exit 0
Temp = length(tapename)
If temp < 4 | temp > 6 then Do
   Say 'Tape name must be 4-6 characters'
   Signal asktape
   End
Asklist:
If listname == '' Then listname = tapename
temp = Readprom('Control file name (CR='listname') ==> ')
If temp ¬= '' Then listname = temp
listname = translate(listname)
If listname = 'QUIT' Then Exit 0
```

```
Address command 'STATE 'listname' TAPCNTRL A'
If RC = 0 Then Do
   Say listname' TAPCNTRL A already exist'
   Asklist:
   Answer = Readprom('Ok to erase Yes/No ? CR=NO ')
   Answer = Translate(Answer)
   If Answer == "" Then Answer = "NO"
   If Answer = 'QUIT' Then Exit 0
   If Find('YES NO',answer) = 0 Then Do
      Say 'Must answer YES or NO'
      Signal asklist
      End
   If Abbrev('YES',answer,1) = 1 Then 'ERASE 'listname' TA'
   Else Exit 4
   End
Asklabel:
Label = Readprom('Tape Label (SL,NL,BLP,AL; CR=NL) ==> ')
Label = Translate(label)
If label = 'QUIT' Then Exit 0
If label == '' Then label = 'NL'
If Find('BLP SL NL AL',label) = 0 Then Do
   Say 'Label must be BLP SL NL or AL'
   Signal asklabel
   End
Asktrk:
tracks = Readprom('Tracks 7 or 9 (CR=9) ==> ')
tracks = Translate(tracks)
If tracks = 'QUIT' Then Exit 0
If tracks == "" Then tracks = 9
If Find('7 9',tracks) =  0 Then Do
   Say 'Tracks must be 7 or 9'
   Signal asktrk
   End
denx = '800 1600 6250'
dend = '1600'
If tracks = 7 Then Do
   denx = '200 556 800'
   dend = '556'
   End
Askden:
Den = Readprom('Tape Density ('denx'; CR='dend') ==> ')
den = Translate(den)
If den = 'QUIT' Then Exit 0
If den == "" Then den = dend
If Find(denx,den) = 0 Then Do
   Say 'Density must be 'denx
   Signal askden
   End
Asktrtch:
If tracks = 9 Then signal askfiles
Trtch = Readprom('TRTCH (O,OC,OT,E,ET; Cr=OC) ==> ')
trtch = Translate(trtch)
If trtch = 'QUIT' Then Exit 0
If trtch == "" Then trtch = 'OC'
If Find('O OC OT E ET',trtch) = 0 Then Do
   Say 'TRTCH must be O OC OT E or ET'
   Signal asktrtch
   End
askfiles:
temp = ':OPTIONS TAPE 'tapename' LABEL 'label' DEN 'den
If tracks = 7 Then temp = temp' TRTCH 'trtch' TRACKS 7'
'EXECIO 1 DISKW 'listname' TAPCNTRL A 1 (FINIS STRING 'te
Askfiles:
temp = readprom('Begin and Ending file numbers (n[,n] Cr=> ')
temp = translate(temp)
If temp = 'QUIT' Then Exit 0
Parse var temp first','second .
If first == '' Then Exit 0
If second == '' Then second = first
If Datatype(first) ¬= 'NUM' | Datatype(second) ¬= 'NUM' T
   Say 'File numbers must be integer values'
   Signal Askfiles
   End
If second < first Then Do
   Say 'First file number must be <= second number'
   Signal Askfiles
   End
Do j = first to second
   fname.j = 'FILE'right(j,4,'0')
End
Askrecfm:
temp = Readprom('RECFM (F,FB,V,VB,U,FA,FBA; CR=FB) ==> ')
temp = translate(temp)
If temp == '' Then temp = 'FB'
If temp = 'QUIT' Then Exit 0
If Find('F FB V VB U FA FBA',temp) = 0 Then Do
   Say 'Record format must be F FB V VB U FA or FBA'
   Signal askrecfm
   End
Recfm = temp
Asklrecl:
Temp = Readprom('LRECL (CR=80) ==> ')
Temp = translate(temp)
If Temp == '' Then temp = 80
If temp = 'QUIT' Then Exit 0
If Datatype(temp) ¬= 'NUM' Then Do
   Say 'LRECL must be integer value'
   Signal Asklrecl
   End
lrecl = temp
Askblock:
Temp = Readprom('BLKSIZE (CR=80) ==> ')
Temp = translate(temp)
If Temp == '' Then temp = 80
If temp = 'QUIT' Then Exit 0
If Datatype(temp) ¬= 'NUM' Then Do
   Say 'Blocksize must be integer value'
   Signal Askblock
   End
Block = temp
Do j = first to second
   temp = j' 'listname' 'fname.j' 'recfm' 'lrecl' 'block
   'EXECIO 1 DISKW 'listname' TAPCNTRL A 0 (FINIS STRING
   End
Signal askfiles /* get another set of files */
Exit 0

/*-------------------------------------------------------
/* Review a TAPCNTRL file info
/*-------------------------------------------------------

Review:
If listname = '?' | listname == '' Then Do
   'LISTF * TAPCNTRL A (D'
   Exit 0
   End
Options = options||' MODE A'
Call Getoptions
'DISKIO READ' listname 'TAPCNTRL * (QUIET FOR *'
Say 'Tape 'tapename label den
Say ' '
Say ' filenum  filename filetype  rf  lrecl     blocksize
Say ' -------  -------- --------  --  -----     ---------
Do J = 1 to Queued()
   Parse upper pull xyz
   Parse var xyz fnum fname ftype trf tlrl tblk comment
   If left(fnum,1) ¬= '*' & left(fnum,1) ¬= ':' Then Do
      Temp = right(fnum,7)||'    '||left(fname,9)||left(f
      Temp = temp||left(trf,3)||right(tlrl,10)||right(tbl
      Say Temp
   End
End

Exit 0

/*-------------------------------------------------------
/* Submit a batch job to read tape.
/*-------------------------------------------------------

Submit:
Call Getoptions
Say ' '
Asktime:
temp = Readprom('CPU Run time (CR= 0:30) ==> ')
temp = translate(temp)
If temp == '' Then temp = '0:30'
If temp = 'QUIT' Then Exit 0
cputime = temp

If owner == '' & lmode == '' Then Do
  Answer = Readprom('Return files in reader? ')
  Answer = translate(answer)
  If Answer = 'QUIT' Then Exit 0
  If Find('Y YE YES O OK ',answer) > 0 Then Do
  lmode = 'A1'
  options = options' MODE 'lmode
  End
Else If owner == '' Then Do
   owner = Readprom('Owner id of disk ')
   owner = translate(owner)
   If owner = 'QUIT' Then Exit 0
   Vaddr = Readprom('Virtual Address  ')
   vaddr = translate(vaddr)
   If vaddr = 'QUIT' Then Exit 0
   Options = options' OWNER 'owner' VADDR 'vaddr
   lmode = ''
   End
end

If owner ¬= '' Then Do
   Temp = chklink(owner vaddr)
   If temp = 1 Then Do
      Say 'Error - disk 'owner vaddr' already linked R/W euser
      Say 'Unable to continue'
      Exit 4
      End
   If temp > 1 Then Do
      Say 'Error 'temp' linking 'owner vaddr
      Say 'Unable to continue'
      Exit 4
      End
end
If Abbrev('SUBMIT',type,1) = 1 Then Do
   Say ' '
   Say 'Mount 'tapename label den
   If owner ¬= '' Then Do
      Say 'Files to be moved to the 'owner vaddr' disk'
      If Password == '' Then Do
         Password = readprom('Enter multi-write password
         Password = translate(password)
         If Password = 'QUIT' Then Exit 0
         options = options' PASS 'password
         End
      End
   Else Say 'Files will be returned to your reader'
   Queue 'RTAPE EXEC'
   Queue listname 'TAPCNTRL'
   Queue ''
   'SET CMSTYPE HT'
   'ERASE RTAPE BATCH A'
   'SET CMSTYPE RT'
   'EXECIO * DISKW RTAPE BATCH A 1 (FINIS'
   Say ' '
   Say 'Submitting...'
   temp = listname' ( 'options
   'BATCH SUBMIT (NOPROF TIME 'cputime' SEND RTAPE) RTAPEemp
```

```
      Say ' '
      Exit RC
      End

/*---------------------------------------------------
/* Read files from tape
/*---------------------------------------------------

Read:
Options = options||' MODE A'
Call Getoptions

If owner ¬= '' Then Do
   'CP .LINK 'owner vaddr ' 199 M 'password
   If RC > 0 Then Do
      Say 'Error linking 'owner vaddr 'RC='rc
      Exit rc
      End
   'ACCESS 199 B'
   lmode = 'B1'
   End
/*---------------------------------------------------
/* Read data from control file
/*---------------------------------------------------

'DISKIO READ' listname 'TAPCNTRL * (QUIET FOR *'

/*---------------------------------------------------
/* STACK a bottom of file marker.
/* Set the LEAVE option for standard labeled tapes.
/* Execute the tape setup command.
/*---------------------------------------------------
trace all
Queue '**BOTTOM**'
If tracks = 7 Then trktype = '7TRK'
else trktype = ''
If Setup = 'YES' Then Do
   Say 'Setting up tape 'tapename ' as 'label den
   temp = tapename' 'label' 'den' 'trktype
   'SETUP TAPE 181' temp '( END )'
   If rc ¬= 0 then Signal 'Err9'
End

/*---------------------------------------------------
/* loop through files to be processed.
/*---------------------------------------------------

'TAPE REW'
position = 0
oldfnm = -1
first = 'TRUE'
Next:
    Parse Upper Pull temp
    Parse var temp fnum .
   If fnum = '**BOTTOM**' then Signal 'END1'
   If fnum = ':OPTIONS' Then Signal Next
   Star = left(temp,1)
   If Star = '*' then Parse var temp '*' fnum fname ftypel tblk .
   Else Parse var temp fnum fname ftype trf tlrl tblk .
   If fflag = 'YES' Then Do
      If fnum < fmin | fnum > fmax Then Signal next
      End
   Else If star = '*' then Signal next
   If trf == '' then trf = recfm
   If tblk == '' then tblk = blksize
   If tlrl == '' then tlrl = lrecl

   /*------------------------------------------------------/
   /* Set file number to LABOFF if label is NL and FNUM i/
   /* more than the previous time.                        /
   /*------------------------------------------------------/

   pnum = oldfnm + 1
   oldfnm = fnum
   If label = 'NL' & fnum = pnum then fnum = 'LABOFF'
   lab = label
   If fnum = 'LABOFF' then lab = ''
   If label = 'BLP' Then fnum = ((fnum-1)*3) + 2
   first = 'FALSE'
   temp =  lab fnum '(RECFM' trf 'LRECL' tlrl 'BLKSIZE' t' den
   If tracks = 7 Then temp = temp' 7TRACK TRTCH 'trtch
   'FILEDEF INMOVE TAP1 'temp
   'FILEDEF OUTMOVE DISK' fname ftype lmode '(RECFM' trf tlrl 'BLK
   Say 'Moving 'lab fnum 'RECFM 'trf' LRECL 'tlrl' BLKSIZ DEN 'den
   Say 'To     'fname ftype lmode
   'MOVEFILE'
   /*------------------------------------------------------/
   /* Check MOVEFILE return code.                         /
   /*------------------------------------------------------/

   rc = rc
   If rc ¬= 0 then Signal 'Err10'
   Signal next
END1:

Exit

Getoptions:
/*---------------------------------------------------------
/* Set the defaults for fileid etc
/*---------------------------------------------------------

If listname == '' Then Do
   Say 'Tape control file name missing'
   Exit 4
   End

If chkfile(listname) > 0 Then Do
```

```
   Say 'Error reading 'listname' TAPECNTRL'
   Exit 4
   End

If listname == '' then listname = 'TAPEFILE'
Tapopt:
lmode = ''
label = 'NL'
den = 1600
trtch = 'OC'
lrecl = 80
recfm = 'FB'
blksize = 1600
tapename = space(listname)
owner = ''
vaddr = ''
password = ''
Setup = 'YES'
options = space(options)
If words(options) > 0 Then Do j = 1 to words(options) by
   w1 = word(options,j)
   w2 = word(options,j+1)

   Select;
     When Abbrev('SETUP',w1,3) = 1 Then Do
        temp = Find('YES NO',w2)
        If temp = 0 Then signal err12
        setup = w2
        End
     When Abbrev('TAPE',w1,3) = 1 Then Do
        tapename = w2
        End
     When Abbrev('OWNER',w1,3) = 1 Then Do
        Owner = w2
        End
     When Abbrev('MODE',w1,3) = 1 Then Do
        lmode = w2
        End
     When w1 = 'VADDR' Then Do
        If owner == '' Then owner = userid()
        vaddr = w2
        End
     When Abbrev('PASSWORD',W1,4) = 1 Then Do
        Password = w2
        End
     When w1 = 'LABEL' Then Do
        temp = Find('BLP SL NL AL',w2)
        If temp = 0 Then Do
           Say w2 'is not a valid LABEL option'
           Exit 24
        End
        label = w2
        End
     When w1 = 'DEN' Then Do
```

```
                temp = Find('800 1600 6250',w2)
                If temp = 0 Then Do
                   Say w2 'is not a valid density'
                   Exit 24
                End
                den = w2
                End
             When w1 = 'TRACKS' Then Do
                temp = Find('7 9',w2)
                If temp = 0 Then Do
                   Say w2 'is not a valid track, must be 7 or 9'
                   Exit 24
                End
                tracks = w2
                End
             When w1 = 'LRECL' Then Do
                temp = Datatype(w2)
                If temp ¬= 'NUM' Then Do
                   Say w2 'is not a valid LRECL, must be integer'
                   Exit 24
                End
                recfm = w2
                End
             When w1 = 'RECFM' Then Do
                temp = Find('F FB V VB U FA FBA',w2)
                If temp = 0 Then Do
                   Say w2 'is not a valid RECFM, must be F FB V Vr FBA'
                   Exit 24
                End
                If temp = 0 Then Signal Err7
                recfm = w2
                End
             When w1 = 'BLKSIZE' Then Do
                temp = Datatype(w2)
                If temp ¬= 'NUM' Then Do
                   Say w2 'is not a valid BLOCKSIZE, must be inte
                   Exit 24
                End
                blksize = w2
                End
             When Abbrev('FILES',w1,1) = 1 Then Do
                Parse var w2 fmin','fmax
                If fmax == '' Then fmax = fmin
                fflag = 'YES'
                End
             When w1 = 'TRTCH' Then Do
                temp = Find('O OC OT E ET',w1)
                If temp = 0 Then Do
                   Say 'TRTCH must be O OC OT E ET - value was 'w
                   Exit 4
                End
                trtch = w2
                End
             Otherwise Do
```

```
            Say 'Unknown option 'w2 'ignorned'
            End
         End
      End

      If lmode == '' & owner ¬= '' Then  Do
         If password == '' Then Do
            Say 'Disk PASSWORD missing'
            Exit 4
            End
         If vaddr == '' Then Do
            Say 'Disk VADDR not specified'
            Exit 4
            End
        End

      /*-----------------------------------------------------
      /* Check for legal arguments
      /*-----------------------------------------------------

      temp = Length(tapename)
      If temp > 3 | temp < 7 Then Return
      Say tapename' is not a valid tape name, must be 4-6 chara
      Exit 24

      /*-----------------------------------------------------
      /*
      /* CHKLINK ( userid vaddr )
      /*
      /* Returns 0=no write links to disk
      /*           1=write link exist.. userid is in writeuser
      /*          >1=error in link to disk.  RC returned
      /*
      /*-----------------------------------------------------

      Chklink:
      Arg id addr

      /* link mini-disk to a known address to allow a query */
      'CP .LINK ' id addr ' 199 RR'
      If RC > 0 Then Do
        'CP LINK ' id addr ' 199 RR'
        Return RC
      End

      'CPSTACK Q LINKS 199'
      'CP .DETACH 199'
      /* Collect multiple line response in a single rex variabl

      string2 = ''
      Do J = 1 to Queued()
         Parse Upper Pull line
         string2 = string2||line' '
      End
```

```
writeuser = ''
Do J = 1 to words(string2) by 3
   If left(word(string2,j+2),3) = 'R/W' Then writeuser = ing2,j)
End
If writeuser ¬= '' Then return 1
Return 0


/*------------------------------------------------------------
/* look for TAPCNTRL filetype if filetype not specified
/*------------------------------------------------------------
Chkfile:
options2 = ''
'STATE' listname 'TAPCNTRL *'
If rc = 0 then Do
   'EXECIO * DISKR 'listname' TAPCNTRL * (FINIS FIND /:OP
   If Queued() > 0 Then Parse Pull .
   If Queued() > 0 Then Parse Pull . options2
   Options = options2||' '||options
   Return 0
   End
Return Rc


TELL:

/*------------------------------------------------------------
/* Tell the user what the exec file does and how to use i
/*------------------------------------------------------------

Select;
  When Abbrev('SUBMIT',listname,1) = 1 Then Signal tellsu
  When Abbrev('PROMPT',listname,1) = 1 Then Signal tellpr
  When Abbrev('REVIEW',listname,1) = 1 Then Signal tellre
  When Abbrev('READ',listname,1)   = 1 Then Signal tellre
  Otherwise
End
'CLRSCRN'
Say 'Commands syntax is:  RTAPE command arguments ( optio
Say ' '
Say 'Where commands are:'
Say ' '
Call saysubmit
Call sayprompt
Call sayreview
Call sayread
Say ' For more help on any command issue RTAPE ? command'
Exit 0

Tellsubmit:
Say ' '
call saysubmit
Call wherelist
Call optiona
```

```
Exit 0

Wherelist:
Say ' Where:'
Say ' '
Say '   listname   is the filename of a cms file with fil'
Say '              TAPCNTRL on the users A disk.  Command
Say '              would generally be the way to originale'
Say '              the listname file.'
Return

Optiona:
Say ' '
Say '   optiona    [LRECL nnn] [BLKSIZE nnn] '
Return
Saysubmit:
Say '    RTAPE SUBMIT listname ( options '
Say '         Submits batch monitor job to retrieve the fied'
Say '         in the listname TAPCNTRL file'
Say ' '
Return

Sayprompt:
Say '    RTAPE PROMPT listname'
Say '         Aid for creating a new listname TAPCNTRL fil
Say ' '
Return

Sayreview:
Say '    RTAPE REVIEW [listname]'
Say '         To examine the items in a specific listname  file'
Say '         or review the TAPCNTRL files on you A disk'
Say ' '
Return

Sayread:
Say '    RTAPE READ listname (options'
Say '         To read files specified in listname directlysk'
Say '         Note that you must have tape mounted.'
Say ' '
Return

Tellsubmit:
Say ' '
Say ' RTAPE SUBMIT listname (options'
Say ' '
Say ' Submits batch monitor job to retrieve files listed
Say ' listname TAPCNTRL file.'
Say ' '
Say ' Where:'
Say ' '
Say ' listname  is the filename of a file on the A disk wletype'
Say '           of TAPCNTRL'

```

```

/*-----------------------------------------------------
/* Error messages
/*-----------------------------------------------------

Err9:
Say 'SETUP of' tapename 'failed - is tape in library at s
Say 'density (' den ') and label (' label ')'
Exit 8

Err10:
Say 'Error in MOVEFILE command return code was' rc
Say 'file number =' fnum  'filename =' fname
Say 'filetype =' ftype  'and filemode =' lmode '.'
Exit rc

Err11:
Say 'file' listname 'TAPCNTRL not found, read tape aborte
Exit 12

Err12
Say "Invalid SETUP argument '"w2"' must be YES or NO"
Exit 12
Exit 0  /* end of program */
```

SETUP

```
* SETUP  (07/18/84, 21:06:48, USGS    )
!SET NOECHO
-
- Setup cards
-
- George Crane 6/6/83
-
IF $EDITOR ¬= 'WYLBUR' THEN * Must be in WYLBUR - Command
THEN RETURN
SET GLOBAL FORMAT USGS:SETUP
SELECT INSTITUTIONS
JUMP FIND.TYPE
-----------------------------------------------------
- Return here if prompt ended in attention          -
-----------------------------------------------------
++NOCONT
*
* SETUP session aborted.
RETURN
-----------------------------------------------------
- Find out the SETUP type, index or comment         -
-----------------------------------------------------
++FIND.TYPE
LET TYPE = $ASK
IF #TYPE : JUMP TYPE.GIVEN
ELSE LET TYPE = 'COMMENT'
ELSE JUMP TYPE.GIVEN
SHOW EVAL ' '
++TYPE.PROMPT
..PROMPT SETUP.TYPE        'INDEX, COMMENT or BOTH ? '
IF #V0 = '' : * You must supply a value
THEN JUMP TYPE.PROMPT
LET TYPE = #V0
++TYPE.GIVEN
LET TYPE = $PMATCH(#TYPE,'IND?EX','COM?MENT','BOT?H')
IF #TYPE = 0 : * Illegal type, must be INDEX or COMMENT
THEN JUMP TYPE.PROMPT
-----------------------------------------------------
- Find out the default file name                    -
-----------------------------------------------------
SHOW EVAL ' '
++FIND.FN
..PROMPT SETUP.FN          'File name ?         '
IF #V0 = '' : * You must supply a value
THEN JUMP FIND.FN
IF $SIZE(#V0) > 8 THEN /* '#V0' greater than 8 characters
THEN JUMP FIND.FN
LET FN = #V0
/CMS SET CMSTYPE HT
/CMS LISTF #FN * B
LET RC = $RC
IF #RC > 0 THEN JUMP SETUPTYPE
```

```
CMS SET CMSTYPE RT
SHO EVAL 'The following files already exist with file nam''''
SHO EVAL ' '
/CMS LISTF #FN * B (D
++FIND.WHAT
SHO EVAL ' '
..PROMPT SETUP.DUPLICATE 'Now What? (ERASE/Redo/Pause/Cono) '
IF #VO = '' THEN JUMP FIND.FN
IF $PMATCH(#VO,'R?EDO') THEN JUMP FIND.FN
IF $PMATCH(#VO,'C?ONTINUE') THEN JUMP SETUPTYPE
IF #vO = 'ERASE'  THEN /ERASE #FN * A
THEN /* Above list of files erased
THEN JUMP PICK.CONT
IF $PMATCH(#VO,'P?AUSE') THEN BREAK XEQ
THEN JUMP FIND.FN
/* '#VO' invalid response
JUMP FIND.WHAT
++PICK.CONT
++SETUPTYPE
-----------------------------------------------------
- Go to the correct setup type                      -
-----------------------------------------------------
IF #TYPE = 3 THEN GOTO SETUP.INDEX
IF #TYPE = 1 THEN GOTO SETUP.INDEX
ELSE GOTO SETUP.COMMENT
-
-----------------------------------------------------
-           Setup Comment                           -
-----------------------------------------------------
++SETUP.COMMENT
CMS SET CMSTYPE HT
/STATE #FN COMMENT A
IF $RC = 0 THEN /ERASE #FN COMMENT A
CMS SET CMSTYPE RT
-
..VALUE TITLE TITLE ALIGN C*
-
..VALUE AUTHOR AUTHOR ALIGN C*
-
..VALUE INSTITUTION INSTITUT ALIGN C*
-
..VALUE ABSTRACT ABSTRACT ALIGN C*
-
..VALUE REFERENCE REFERENC ALIGN C*
-
..VALUE FORMAT FORMAT NOALIGN C*
-
CLR ACT
/COP FROM #FN TITLE A
/COP FROM #FN AUTHOR A
/COP FROM #FN INSTITUT
/COP FROM #FN ABSTRACT A
/COP FROM #FN REFERENC A
/COP FROM #FN FORMAT A
```

```
END,C*END--------------------------------------------------------
/SAVE #FN COMMENT B REP
-
++SETUP.INDEX
-
..VALUE DATE DATE SINGLE C*
-
..VALUE CLASS CLASS SINGLE C*
-
..VALUE PERSN PERSN ALIGN C*
-
..VALUE ALPHA ALPHA ALIGN C*
-
..VALUE KEYWD KEYWD ALIGN C*
-
CLR ACT
/COP FROM #FN DATE
/COP FROM #FN CLASS
/COP FROM #FN PERSN
/COP FROM #FN ALPHA
/COP FROM #FN KEYWD
-
/SAVE #FN INDEX B REP
IF #TYPE = 3 THEN GOTO SETUP.COMMENT
```

USGS:INVENTORY

```
FILE USGS:INVENTORY/MASTER WHLW6,MEOW6
SUBFILE INVENTORY.MAINT/ACCOUNTS $W6
SUBFILE EARTHQUAKE INVENTORY/ACCOUNTS $W6/SWITCH 3
EXPLAIN This is the United States Geological Survey
EXPLAIN earthquake archiving and retrieval data base
EXPLAIN system.  It is used for the purpose of
EXPLAIN archival, query and retrieval of earthquake-
EXPLAIN related data sets.
SUBFILE INVENTORY/ACCOUNTS USGS, WHLW6,MEOW6
CMD SET XEQ QUAKE PROTO
CMD ..CMD.INVENTORY
-
KEY DSN/LEN 8/SINGLE/CAP
EXPL This is the 8-character code for an unique
EXPL data set name.
ELEM SIZE/INT/SING
EXPL A 6 digit integer for specifying the size of the
EXPL data set (total number of cards in the data set).
ELEM DATE/DATE/SING/CALL FIXDATE/IND/OUT,CALL DATEOUT
EXPL The date (YYMMDD) on which the data set is archived.
ELEM ARCH/LEN 2/SINGLE/CAP
EXPL A 2 character code denoting the archivist
ELEM TAPE/LEN 6/SING/CAP
EXPL A 6 character code specifying the archive type.
ELEM FILE/INT/SING/RANGE 0,999
EXPL A 3 digit integer specifying the tape file number.
ELEM STRT/INT/SING/RANGE 0,999999
EXPL A 6 digit interger specifying the start position of
EXPL the data set.  This integer is the card sequence
EXPL number in a given tape file at which this data begin
ELEM TITLE/SQU/SING/IND KEYWORDS
EXPL Data set title
ELEM SUBMIT/DATE/SINGLE/IND/CAL FIXDATE
ELEM RELEASE-MONTHS/INT/SINGLE
ELEM RELEASE-DATE/DATE/SINGLE/IND
INPR $DATE/ $SQU/ $CALL(RELEASE,INCLOSE)
SRCP $DATE(TRUNC)
ELEM CLASS/CHANGE ',',' '/SQU/CAP
    /LOOKUP VERIFY,CLASS,,W/IND/IND KEYWORDS
OUTP $CAP(EACH.WORD)
SRCPROC $CAP/ $CHANGE('.','. ')/ $SQU/
  $SUBF.LOOKUP(REPLACE, INSTITUTIONS, 'INST-NAME', D)
EXPL This is the Class and Subclass catagory.  First word
EXPL considered the class and second (and subsequent) wor
EXPL are considered part of the sub-class value.  Commas
EXPL optional.  I.e.  'Electric, Phase', ELECTRIC is the
EXPL class and PHASE is the sub-class.
ELEM PERSON,AUTHOR/BRE ';'/NAME/INDEX/CAP/SQU
EXPL Contains names of author(s), principal inventigator(
EXPL persons submitting the data set. -r Enter in the ord
EXPL first, middle, and last.
ELEM INSTITUTION,INST/SINGLE/IND/CAP/CHANGE '.','. '/SQU/
```

```
EXPL Institution submitting data set.  Value must be 1-4
EXPL character institution code from table.
ELEM INST-CODE,ICODE/SINGLE/IND/CAP/SQU
EXPL Three or for character institution identification co
ELEM KEYWORDS,KEY,K/CAP/IND/WORD
EXPL Keywords provided by author.
ELEM ALPHA/STR
   ELEM BEGIN-DATE,BD,MINDAT/DATE/SINGLE/CALL FIXDATE/OUTTEOUT/IND
   EXPL Beginning date of the data set (YYMMDD).
   ELEM END-DATE,MAXDAT,ED/DATE/SINGLE/CALL FIXDATE/OUT,COUT/IND
   EXPL Ending date of the data set (YYMMDD).
   ELEM BOTTOM-LAT,MINLAT/REAL/SINGLE
   EXPL Southernmost (bottom) latitutude in degrees for t
   EXPL data set.
   ELEM TOP-LAT,MAXLAT/REAL/SINGLE
   EXPL Northernmost (top) latitutude in degrees for the .
   ELEM LEFT-LONG,MINLON/REAL/SINGLE
   EXPL Westernmost (left-side) longitutude in degrees.
   ELEM RIGHT-LONG,MAXLON/REAL/SINGLE
   EXPL Easternmost (right-side) longitude in degrees.
   ELEM CONTRACT-NUMBER,PROJECT/SINGLE
   EXPL Project or contract number
   ELEM DATA-REF-NO,REF/SINGLE/CAP/SQU
   EXPL Data reference number.
   END
VIRTUAL
ELEM LIBRARY/SINGLE/CAP/SQU/IND/OUT,CALL GETLIB
EXPL This is the first two characters of element DSN.
ELEM COLAT/IND/REAL
  INPROC $SQU
  OUTPROC $CALL(GETLATMIN)
ELEM COLATB/IND COLAT/REAL
  INPROC $SQU
  OUTPROC $CALL(GETLATMAX)
ELEM COLATMIN/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETLATMIN)
ELEM COLATMAX/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETLATMAX)
ELEM EALON/IND/REAL
  INPROC $SQU
  OUTPROC $CALL(GETMINEA)
ELEM EALONB/IND EALON/REAL
  INPROC $SQU
  OUTPROC $CALL(GETMAXEA)
ELEM EALONMIN/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETMINLON)
ELEM EALONMAX/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETMAXLON)
ELEM DAT/IND/REAL
  INPROC $SQU
  OUTPROC $CALL(GETDATMIN)
ELEM DATB/IND DAT/REAL
  INPROC $SQU
  OUTPROC $CALL(GETDATMAX)
ELEM DATMIN/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETDATMIN)
ELEM DATMAX/INDEX/REAL
  INPROC $SQU
  OUTPROC $CALL(GETDATMAX)
VARIABLE T1/STRING
VARIABLE T2/STRING
VARIABLE T3/STRING
VARIABLE T4/STRING
VARIABLE F1/REAL
VARIABLE F2/REAL
VARIABLE F3/REAL
VARIABLE F4/REAL
VARIABLE SHORT/INT
VARIABLE S1/REAL
VARIABLE S2/REAL
VARIABLE S3/REAL
VARIABLE S4/REAL
VARIABLE NOPASS/INT
VARIABLE MIN/REAL
VARIABLE MAX/REAL
VARIABLE MIN1/REAL
VARIABLE MAX1/REAL
VARIABLE DATE/STRING
VARIABLE MM/INT
VARIABLE DD/STRING
VARIABLE YY/INT
VARIABLE NEW/INT
VARIABLE ADD/INT
USE GETLIB
  SET VAL $LSTR($GETUVAL(DSN,0,'XX'),2)
  END
USE CHKNOPASS
  LET NOPASS = 1
  LET T1 = $GETUVAL(MINLAT,0,?)
  LET T2 = $GETUVAL(MAXLAT,0,?)
  LET T3 = $GETUVAL(MINLON,0,?)
  LET T4 = $GETUVAL(MAXLON,0,?)
  -
  IF #T1 = '?' THEN RETURN
  IF #T2 = '?' THEN LET T2 = #T1
  IF #T3 = '?' THEN RETURN
  IF #T4 = '?' THEN LET T4 = #T3
  -
  LET S1 = $REAL(#T1)
  LET S2 = $REAL(#T2)
  LET S3 = $REAL(#T3)
  LET S4 = $REAL(#T4)
  -
```

```
     IF #S1 = 90 THEN IF #S2 = -90 THEN RETURN
     IF #S3 = 0  THEN IF #S4 = 0   THEN RETURN
     IF #S3 = 180 THEN IF #S4 = -180 THEN RETURN
     -
     LET NOPASS = 0
     RETURN
     END
   USE GETLATMIN
     XEQ USE CHKNOPASS
     IF #NOPASS THEN SET VAL ''
     THEN RETURN
     -
     XEQ USE GETCO
     SET VAL $RETYPE(#MIN,STRING)
     RETURN
     END
   USE GETLATMAX
     XEQ USE CHKNOPASS
     IF #NOPASS THEN SET VAL ''
     THEN RETURN
     -
     XEQ USE GETCO
     SET VAL $RETYPE(#MAX,STRING)
     RETURN
     END
   USE GETCO
     IF #S1 = -90 THEN IF #S2 = 90 THEN JUMP WORLD
     -
     LET MIN = 90 - #S1
     LET MAX = 90 - #S2
     IF #MIN < #MAX THEN RETURN
     LET S1 = #MIN
     LET MIN = #MAX
     LET MAX = #S1
     RETURN
     -
     ++WORLD
     LET MIN = 0
     LET MAX = 180
     RETURN
     END
   USE GETMINEA
     XEQ USE CHKNOPASS
     IF #NOPASS THEN SET VAL ''
     THEN RETURN
     -
     XEQ USE GETEA
     SET VAL $RETYPE(#MIN1,STRING)
     RETURN
     END
   USE GETMAXEA
     XEQ USE CHKNOPASS
     IF #NOPASS THEN SET VAL ''
     THEN RETURN
```

```
  -
  XEQ USE GETEA
  SET VAL $RETYPE(#MAX1,STRING)
  RETURN
  END
USE GETMINLON
  XEQ USE CHKNOPASS
  IF #NOPASS THEN SET VAL ''
  THEN RETURN
  -
  XEQ USE GETEA
  SET VAL ''
  IF #SHORT THEN RETURN
  SET VAL $RETYPE(#MIN1,STRING)
  RETURN
  END
USE GETMAXLON
  XEQ USE CHKNOPASS
  IF #NOPASS THEN SET VAL ''
  THEN RETURN
  -
  XEQ USE GETEA
  SET VAL ''
  IF #SHORT THEN RETURN
  SET VAL $RETYPE(#MAX1,STRING)
  RETURN
  END
USE GETEA
  IF #S3 = -180 THEN IF #S4 = 180 THEN JUMP WORLD
  -
  LET MIN1 = #S3
  LET MAX1 = #S4
  IF #MIN1 < 0 THEN LET MIN1 = 360 + #MIN1
  IF #MAX1 < 0 THEN LET MAX1 = 360 + #MAX1
  RETURN
  -
  ++WORLD
  LET MIN1 = 0
  LET MAX1 = 360
  RETURN
  END
USERPROC GETDATMIN
   XEQ USE GETDAT
   IF #NOPASS : SET VAL ''
   THEN RETURN
   SET VAL $RETYPE(#F1,STRING)
   RETURN
   END
USERPROC GETDATMAX
   XEQ USE GETDAT
   IF #NOPASS : SET VAL ''
   THEN RETURN
   SET VAL $RETYPE(#F2,STRING)
   RETURN
   END
USERPROC GETDAT
   LET NOPASS = $TRUE
   LET T1 = $GETUVAL(MINDAT,0,?)
   LET T2 = $GETUVAL(MAXDAT,0,?)
   IF #T2 = '?' : LET T2 = #T1;
   IF #T1 = '?' THEN RETURN
   LET F1 = #T1
   LET F2 = #T2
   LET NOPASS = $FALSE
   IF #F1 < #F2 : RETURN
   LET F3 = #F2
   LET F2 = #F1
   LET F1 = #F3
   RETURN
   END
USERPROC FIXDATE
   IF $SPAN($VAL,0123456789) ¬=$VAL THEN RETURN
   IF $SIZE($VAL)=6 THEN JUMP S6
   IF $SIZE($VAL)=8 THEN JUMP S8
   SET ERROR S
   RETURN
   ++S6
   SET VAL $SUBSTR($VAL,2,2)'/'            $SUBSTR($VAL,4,2)'/'
   RETURN
   ++S8
   SET VAL $SUBSTR($VAL,4,2)'/'            $SUBSTR($VAL,6,2)'/'
   RETURN
   END
USERPROC DATEOUT
   SET VAL $SUBSTR($VAL,6,2)          $SUBSTR($VAL,0,2)
   END
USERPROC RELEASE
   SET OCC 1
   SET DELETE
   LET DATE = $DATEOUT($GETUVAL('SUBMIT',0),0)
   LET MM = $SUBSTR(#DATE,0,2)
   LET DD = $SUBSTR(#DATE,3,2)
   LET YY = $SUBSTR(#DATE,6,2)
   LET ADD = $GETUVAL('RELEASE-MONTHS',0)
   LET MM = #MM + #ADD
   IF #MM <= $INT(12) THEN SET VAL #DATE
   THEN RETURN
   LET NEW = $TRUNC(#MM/12)
   LET MM = $MOD(#MM,12)
   LET YY = #YY + #NEW
   SET VAL $INSETR(#MM,'0',2)||'/'||#DD||'/'||#YY
   RETURN
   END
-
GOAL CLASS/PASS KEY
SUBFILE CLASS/ACCOUNTS USGS,WHLW6,MEOW6/FORMAT $PROMPT
EXPL This file contains the CLASS and SUB-CLASS definitio
EXPL and associated code value which is a 4 character fie
EXPL This subfile is used as a lookup table for CLASS.
KEY CLASS/SINGLE/CAP/CHANGE ',',''/SQU
EXPL This is the Class and Subclass value, the first word
EXPL is considered the CLASS and second (and subsequent)
EXPL words area considered to be the SUB-CLASS value.
EXPL Commas are ignored.
ELE CODE/SINGLE/CAP/IND/RECNAME CODE
EXPL This is a code assigned to each unique class/sub-cla
EXPL value.  It is a 4 byte field, the first two bytes be
EXPL the code for the CLASS and the second two the code f
EXPL the associated SUB-CLASS
-
GOAL INST/PASS KEY
SUBFILE INSTITUTIONS/ACCOUNTS USGS,WHLW6,MEOW6/FORMAT $PR
EXPL This subfile is a lookup table for archival institut
EXPL name codes.
KEY INST-CODE/SINGLE/CAP
EXPL This is a 1-4 character code field which is assigned
EXPL to each institution.
ELE INST-NAME/SINGLE/IND/WORD
EXPL The full institution name.
ELE INST-ADDR
EXPL This is the general mailing address for the institit
EXPL it is not required.
ELE CONTACT/SINGLE
ELE GEO-CODE,GC/SINGLE
ELE COUNTRY-CODE,CC/SINGLE
ELE STATE-CODE,SC/SINGLE
ELE CITY-SORT,CS/SINGLE
-
GOAL CODE/PASS KEY
- Used as an index record of element CODE in the CLASS su
- for the purpose of lookup from the INVENTORY subfile.
KEY CODE/SINGLE/CAP
```

USGS:LIBDATA

```
FILE USGS:LIBDATA
SUBFILE LIBDATA/ACC USGS,WHLW6,MEOW6/FOR $PROMPT
KEY LIBRARY/SINGLE/CAP
ELE TAPEVOL,TAPENAME,TAPE/SINGLE/TEXT/CAP
ELE DSN/INT/SINGLE
ELE FILENO,FILE/SINGLE/INT
ELE START,RECNO,REC/SINGLE/INT
ELE SIZE/SINGLE/INT
ELE CUMSIZE,NCARDS,NCARD/SINGLE/INT
ELE UPDACCT/SINGLE
INPROC $GEN.ACCT
ELE UPDDATE/DATE UPD/SINGLE
ELE UPDTIME/TIME UPD/SINGLE
```

USGS:LIBDATA

```
ID = USGS:LIBDATA;
MODDATE = THUR. AUG. 16, 1984;
DEFDATE = TUES. JULY 5, 1983;
FILE = USGS:LIBDATA;
RECORD-NAME = REC01;
FRAME-ID = GETDATA;
  DIRECTION = OUTPUT;
  FRAME-DIM = 0,160;
  USAGE = NAMED, DISPLAY;
  LABEL = LIBRARY;
    GETELEM;
    UPROC = LET LIBRARY = $CVAL;
  LABEL = DSN;
    GETELEM;
    UPROC = LET DSN = $CVAL;
  LABEL = TAPEVOL;
    GETELEM;
    UPROC = LET TAPEVOL = $CVAL;
  LABEL = FILENO;
    GETELEM;
    UPROC = LET FILENO = $CVAL;
  LABEL = START;
    GETELEM;
    UPROC = LET START = $CVAL;
  LABEL = SIZE;
    GETELEM;
    UPROC = LET SIZE = $CVAL;
  LABEL = CUMSIZE;
    GETELEM;
    UPROC = LET CUMSIZE = $CVAL;
FRAME-ID = PUTDATA;
  DIRECTION = INPUT;
  USAGE = NAMED, MERGE;
  LABEL = LIBRARY;
    VALUE = #LIBRARY;
    PUTELEM;
  LABEL = DSN;
    VALUE = $STRING(#DSN);
    PUTELEM;
  LABEL = TAPEVOL;
    VALUE = #TAPEVOL;
    PUTELEM;
  LABEL = FILENO;
    VALUE = $STRING(#FILENO);
    PUTELEM;
  LABEL = START;
    VALUE = $STRING(#START);
    PUTELEM;
  LABEL = SIZE;
    VALUE = $STRING(#SIZE);
    PUTELEM;
  LABEL = CUMSIZE;
```

```
      VALUE = $STRING(#CUMSIZE);
      PUTELEM;
  FRAME-ID = ADDDATA;
    DIRECTION = INPUT;
    USAGE = FULL, NAMED;
    LABEL = LIBRARY;
      VALUE = #LIBRARY;
      PUTELEM;
    LABEL = DSN;
      VALUE = $STRING(#DSN);
      PUTELEM;
    LABEL = TAPEVOL;
      VALUE = #TAPEVOL;
      PUTELEM;
    LABEL = FILENO;
      VALUE = $STRING(#FILENO);
      PUTELEM;
    LABEL = START;
      VALUE = $STRING(#START);
      PUTELEM;
    LABEL = SIZE;
      VALUE = $STRING(#SIZE);
      PUTELEM;
    LABEL = CUMSIZE;
      VALUE = $STRING(#CUMSIZE);
      PUTELEM;
  FORMAT-NAME = LIBDATA;
    ALLOCATE = USGS:SETUP;
    FRAME-NAME = GETDATA;
      FRAME-TYPE = DATA;
    FRAME-NAME = PUTDATA;
      FRAME-TYPE = DATA;
    FRAME-NAME = ADDDATA;
      FRAME-TYPE = DATA;
```

USGS:INVENTORY.NEWADD

```
ID = USGS:INVENTORY.NEWADD;
AUTHOR = George Crane;
MODDATE = MON. NOV. 15, 1982;
DEFDATE = MON. SEPT. 27, 1982;
FILE = USGS:INVENTORY;
RECORD-NAME = REC01;
VGROUP = USGS:LOCAL;
  VARIABLE = TEMP;
    LENGTH = 200;
    TYPE = STRING;
  VARIABLE = I1;
    TYPE = INT;
FRAME-ID = ADD;
  DIRECTION = INPUT;
  FRAME-DIM = 0,80;
  USAGE = FULL;
  LABEL = GETLINE;
    START = X,1;
    LENGTH = 5;
    GETDATA = 4, 5;
    UPROC = IF $CVAL = 'C#DSN' : JUMP CNTRL;
    UPROC = IF $CVAL = 'C$AKE' : JUMP AKEYS;
    UPROC = IF $CVAL = 'C$IKE' : JUMP IKEYS;
    UPROC = IF $CVAL = 'C$RKE' : JUMP RKEYS;
    UPROC = IF $CVAL = 'C*END' : RETURN;
    UPROC = JUMP GETLINE;
  LABEL;
    UPROC = RETURN;
  LABEL = CNTRL;
    START = *,7;
    LENGTH = 8;
    GETDATA = 4, 5;
    PUTELEM = DSN;
  LABEL = SIZE;
    START = *,21;
    LENGTH = 6;
    GETDATA = 4, 5;
    PUTELEM = SIZE;
  LABEL = DATE;
    START = *,33;
    LENGTH = 6;
    GETDATA = 4, 5;
    PUTELEM = DATE;
  LABEL = ARCH;
    START = *,45;
    LENGTH = 2;
    GETDATA = 4, 5;
    PUTELEM = ARCH;
  LABEL = TAPE;
    START = *,53;
    LENGTH = 6;
    GETDATA = 4, 5;
  PUTELEM = TAPE;
LABEL = TAPEFILE;
  START = *,65;
  LENGTH = 3;
  GETDATA = 4, 5;
  PUTELEM = TAPEFILE;
LABEL = STRT;
  START = *,74;
  LENGTH = 6;
  GETDATA = 4, 5;
  PUTELEM = STRT;
LABEL;
  UPROC = JUMP GETLINE;
LABEL = AKEYS;
  PUTELEM = AKEYS-GROUP;
LABEL;
  START = *,10;
  LENGTH = 70;
  GETDATA = 4, 5;
  UPROC = LET TEMP = $CVAL;
LABEL = DATA-TYPE;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = DATASET-NAME;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = AUTHOR;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = SUPPLIER;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = REGION;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = LOCALITY;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
  PUTELEM;
LABEL = AKEY;
  VALUE = "$BREAK(#TEMP,';')";
  UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
  UPROC = IF $UVAL = '' : JUMP GETLINE;
```

```
          PUTELEM;
          LOOP = 1;
        LABEL;
          UPROC = JUMP GETLINE;
        LABEL = IKEYS;
          PUTELEM = IKEYS-GROUP;
        LABEL;
          START = *,10;
          LENGTH = 70;
          GETDATA = 4, 5;
          UPROC = LET TEMP = $CVAL;
        LABEL = START-DATE;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP GETLINE;
          PUTELEM;
        LABEL = END-DATE;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP GETLINE;
          PUTELEM;
        LABEL = REFNUM;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP GETLINE;
          PUTELEM;
        LABEL = CELNUM;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP IKEY;
        LABEL;
          VALUE = $PVAL;
          PUTELEM = CELNUM;
        LABEL = IKEY;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP GETLINE;
          UPROC = LET I1 = 0;
        LABEL;
          VALUE = $PVAL;
          PUTELEM = IKEY::#I1;
        LABEL;
          UPROC = LET I1 = #I1+1;
          UPROC = IF #I1 > 2 : JUMP GETLINE;
        LABEL;
          UPROC = JUMP IKEY;
        LABEL = RKEYS;
          PUTELEM = RKEYS-GROUP;
        LABEL;
          START = *,10;
          LENGTH = 70;
          GETDATA = 4, 5;
          UPROC = LET TEMP = $CVAL;
        LABEL = MINLAT;
```

```
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP MAXLAT;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = MAXLAT;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP MINLON;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = MINLON;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP MAXLON;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = MAXLON;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP MINMAG;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = MINMAG;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP MAXMAG;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = MAXMAG;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP RKEY;
        LABEL;
          VALUE = $PVAL;
          PUTELEM;
        LABEL = RKEY;
          VALUE = "$BREAK(#TEMP,';')";
          UPROC = "LET TEMP = $STRIP($RSUB(#TEMP,';'),' ')";
          UPROC = IF $UVAL = '' : JUMP GETLINE;
          UPROC = LET I1 = 0;
        LABEL;
          VALUE = $PVAL;
          PUTELEM = RKEY::#I1;
        LABEL;
          UPROC = LET I1 = #I1+1;
          UPROC = IF #I1 > 2 : JUMP GETLINE;
        LABEL;
          UPROC = JUMP RKEY;
```

```
FORMAT-NAME = NEWADD;
  ALLOCATE = USGS:LOCAL;
  FRAME-NAME = ADD;
    FRAME-TYPE = DATA;
```

USGS:INVENTORY.ADDCARDS

```
ID = USGS:INVENTORY.ADDCARDS;
AUTHOR = George Crane;
MODDATE = TUES. OCT. 2, 1984;
DEFDATE = MON. SEPT. 27, 1982;
FILE = USGS:INVENTORY;
RECORD-NAME = REC01;
VGROUP = USGS:LOCAL;
  VARIABLE = XVAL;
    OCCURS = 10;
    TYPE = DYNAMIC;
  VARIABLE = LINE;
    LENGTH = 100;
    TYPE = STRING;
  VARIABLE = TEMPD;
    TYPE = STRING;
  VARIABLE = TEMP;
    LENGTH = 200;
    TYPE = STRING;
  VARIABLE = TEMP2;
    LENGTH = 200;
    TYPE = STRING;
  VARIABLE = I1;
    TYPE = INT;
  VARIABLE = PARM;
    OCCURS = 30;
    TYPE = STRING;
  VARIABLE = P;
    TYPE = INT;
FRAME-ID = PARSE;
  DIRECTION = INPUT;
  FRAME-DIM = 0,80;
  LABEL;
    UPROC = LET XVAL::0 = #XVAL::#I1;
    UPROC = LET P = 0;
  LABEL;
    UPROC = LET PARM::$LOOPCT = '';
    LOOP = 20;
  LABEL = LOOP;
    UPROC = "LET TEMP = $BREAK(#XVAL::0,';')";
    UPROC = "LET XVAL::0 = $STRIP($RSUB(#XVAL::0,';'),'
    UPROC = LET P = #P + 1;
    UPROC = LET PARM::#P = $STRIP(#TEMP,' ');
    UPROC = IF #XVAL::0 = '' : RETURN;
    UPROC = JUMP LOOP;
FRAME-ID = ADD;
  DIRECTION = INPUT;
  FRAME-DIM = 0,80;
  USAGE = FULL;
  LABEL;
    UPROC = EVAL $DYNZAP(XVAL,$LOOPCT,'');
    LOOP = 10;
  LABEL = GETLINE;
```

```
      START = X,1;
      LENGTH = 80;
      GETDATA = 5;
      UPROC = LET LINE = $CVAL;
      UPROC = LET TEMP = $BREAK(#LINE,'= :');
      UPROC = LET LINE = $STRIP($RSUB(#LINE,#TEMP),'= :');
      UPROC = IF #TEMP = 'C#DSN' THEN JUMP CNTRL;
      UPROC = IF $MATCH(#TEMP,'C*','C*') THEN JUMP PUTVAL;
      UPROC = IF $MATCH($BREAK(#TEMP,'-'),'C*END') THEN JU
      UPROC = LET I1 = $MATCH(#TEMP,'C*TITLE','C*AUTHOR',
'C*INSTITUTION', 'C*REFERENCE', 'C*CLASS', 'C*PERSN', 'C*
'C*KEYWD','C*DATE');
      UPROC = JUMP PUTVAL;
    LABEL = TITLE;
      VALUE = "$BREAK(#XVAL::1,';')";
      PUTELEM = TITLE;
    LABEL = AUTHOR;
      VALUE = #XVAL::6;
      INPROC = "$BREAK(';')/$SQU/$CAP";
      PUTELEM = AUTHOR;
    LABEL = INSTITUTION;
      VALUE = "$BREAK(#XVAL::3,',;')";
      INPROC = $CAP/ $SQU;
      PUTELEM = INSTITUTION;
    LABEL;
      UPROC = LET I1 = 5;
    LABEL;
      IND-FRAME = PARSE;
    LABEL;
      UPROC = IF #PARM::1 = '' THEN JUMP ENDCLASS;
    LABEL;
      VALUE = #PARM::1', '#PARM::2;
      PUTELEM = CLASS;
    LABEL;
      UPROC = IF #PARM::3 = '' THEN JUMP ENDCLASS;
    LABEL;
      VALUE = #PARM::1', '#PARM::3;
      PUTELEM = CLASS;
    LABEL;
      UPROC = IF #PARM::4 = '' THEN JUMP ENDCLASS;
    LABEL;
      VALUE = #PARM::1', '#PARM::4;
      PUTELEM = CLASS;
    LABEL = ENDCLASS;
      UPROC = LET I1 = 9;
    LABEL;
      IND-FRAME = PARSE;
    LABEL;
      VALUE = #PARM::1;
      UPROC = IF #PARM::2 = '' THEN LET PARM::2 = 0;
      PUTELEM = SUBMIT;
    LABEL;
      VALUE = #PARM::2;
      PUTELEM = RELEASE-MONTHS;
```

```
LABEL;
  VALUE = #XVAL::7;
  INPROC = $CAP/$SQU/$CHANGE(' ','');
  UPROC = LET I1 = 7;
  UPROC = LET XVAL::7 = $CVAL;
LABEL;
  IND-FRAME = PARSE;
LABEL;
  UPROC = IF $STRIP(#PARM::1,' ') = '' THEN JUMP END-D
LABEL;
  VALUE = #PARM::1;
  PUTELEM = BEGIN-DATE;
LABEL = END-DATE;
  UPROC = IF $STRIP(#PARM::2,' ') = '' THEN JUMP LONLA
LABEL;
  VALUE = #PARM::2;
  PUTELEM = END-DATE;
LABEL = LONLAT;
  UPROC = LET I1 = 3;
LABEL;
  UPROC = IF $STRIP(#PARM::#I1,' ') = '' THEN JUMP CON
  UPROC = LET I1 = #I1 + 1;
  LOOP = 3;
LABEL;
  UPROC = LET TEMP = $SPAN(#PARM::3,'1234567890.');
  UPROC = LET TEMP2 = $RSUB(#PARM::3,#TEMP);
  UPROC = IF #TEMP2 = 'S' THEN LET TEMP = '-'#TEMP;
  UPROC = IF $MATCH(#TEMP2,'S','N') : JUMP;
  UPROC = * 'Unrecognized value '''#TEMP2''' for bottote';
  UPROC = ABORT;
LABEL;
  VALUE = #TEMP;
  PUTELEM = BOTTOM-LAT;
LABEL;
  UPROC = LET TEMP = $SPAN(#PARM::4,'1234567890.');
  UPROC = LET TEMP2 = $RSUB(#PARM::4,#TEMP);
  UPROC = IF #TEMP2 = 'S' THEN LET TEMP = '-'#TEMP;
  UPROC = IF $MATCH(#TEMP2,'S','N') : JUMP;
  UPROC = * 'Unrecognized value '''#TEMP2''' for top l;
  UPROC = ABORT;
LABEL;
  VALUE = #TEMP;
  PUTELEM = TOP-LAT;
LABEL;
  UPROC = LET TEMP = $SPAN(#PARM::5,'1234567890.');
  UPROC = LET TEMP2 = $RSUB(#PARM::5,#TEMP);
  UPROC = IF #TEMP2 = 'W' THEN LET TEMP = '-'#TEMP;
  UPROC = IF $MATCH(#TEMP2,'E','W') : JUMP;
  UPROC = * 'Unrecognized value '''#TEMP2''' for left e';
  UPROC = ABORT;
LABEL;
  VALUE = #TEMP;
  PUTELEM = LEFT-LONG;
LABEL;
```

```
          UPROC = LET TEMP = $SPAN(#PARM::6,'1234567890.');
          UPROC = LET TEMP2 = $RSUB(#PARM::6,#TEMP);
          UPROC = IF #TEMP2 = 'W' THEN LET TEMP = '-'#TEMP;
          UPROC = IF $MATCH(#TEMP2,'E','W') : JUMP;
          UPROC = * 'Unrecognized value '''#TEMP2''' for rightte';
          UPROC = ABORT;
        LABEL;
          VALUE = #TEMP;
          PUTELEM = RIGHT-LONG;
        LABEL = CONTRACT;
          VALUE = #PARM::7;
          UPROC = IF #PARM::7 = '' : JUMP;
          PUTELEM = CONTRACT-NUMBER;
        LABEL;
          VALUE = #PARM::8;
          UPROC = IF #PARM::8 = '' : JUMP;
          UPROC = LET I1 = 2;
          PUTELEM = DATA-REF-NO;
        LABEL;
          IND-FRAME = PARSE;
        LABEL;
          UPROC = RETURN;
        LABEL = PUTVAL;
          UPROC = IF #I1 = 0 THEN JUMP GETLINE;
          UPROC = LET XVAL::#I1 = #XVAL::#I1' '#LINE;
          UPROC = JUMP GETLINE;
        LABEL = ERROR;
          UPROC = * 'ERROR ON '#LINE;
          UPROC = ABORT;
        LABEL;
          UPROC = RETURN;
        LABEL = CNTRL;
          START = *,7;
          LENGTH = 8;
          GETDATA = 4, 5;
          PUTELEM = DSN;
        LABEL = SIZE;
          START = *,21;
          LENGTH = 6;
          GETDATA = 4, 5;
          PUTELEM = SIZE;
        LABEL = DATE;
          START = *,33;
          LENGTH = 6;
          GETDATA = 4, 5;
          UPROC = LET TEMPD = $SUBSTR($UVAL,0,2)'/'$SUBSTR($UV/'
    $SUBSTR($UVAL,4,2);
        LABEL;
          VALUE = #TEMPD;
          PUTELEM = DATE;
        LABEL = ARCH;
          START = *,45;
          LENGTH = 2;
          GETDATA = 4, 5;
        PUTELEM = ARCH;
      LABEL = TAPE;
        START = *,53;
        LENGTH = 6;
        GETDATA = 4, 5;
        PUTELEM = TAPE;
      LABEL = FILE;
        START = *,65;
        LENGTH = 3;
        GETDATA = 4, 5;
        PUTELEM = FILE;
      LABEL = STRT;
        START = *,74;
        LENGTH = 6;
        GETDATA = 4, 5;
        PUTELEM = STRT;
      LABEL;
        UPROC = JUMP GETLINE;
    FORMAT-NAME = ADDCARDS;
      ALLOCATE = USGS:LOCAL;
      FRAME-NAME = PARSE;
        FRAME-TYPE = INDIRECT;
      FRAME-NAME = ADD;
        FRAME-TYPE = DATA;
```

```
 ID = USGS:SETUP;
 MODDATE = THUR. MARCH 29, 1984;
 DEFDATE = TUES. JUNE 7, 1983;
 FRAME-ID = VALUEOUT;
   DIRECTION = OUTPUT;
   FRAME-DIM = 80,80;
   LABEL;
     UPROC = LET LOOP = #T1 - 1;
     UPROC = LET I2 = 1;
     UPROC = IF #LOOP < 0 THEN LET LOOP = 0;
     UPROC = LET I3 = $SIZE(#TYPE) + 5;
     UPROC = IF #ALIGN = 'NOALIGN' THEN JUMP NOALIGN;
   LABEL;
     VALUE = #VALUE::#I2;
     MARGINS = #I3,80;
     START = 1,#I3;
     OUTPROC = $SQU;
     BREAK = " ";
     UPROC = LET I2 = #I2 + 1;
     PUTDATA;
     LOOP = #LOOP;
     XSTART = *,*+2;
   LABEL;
     VALUE = "';'";
     MARGINS = #I3,80;
     START = *,*+1;
     UPROC = IF ¬$MATCH(#TYPE,'DATE','CLASS','PERSN','ALPWD') :
JUMP;
     UPROC = -IF #CSYM = 'C*' THEN JUMP;
     UPROC = LET I2 = #I2 - 1;
     UPROC = IF #I2 < 0 THEN LET I2 = 0;
     UPROC = LET VALUE::#I2 = $CHAR(#VALUE::#I2);
     UPROC = IF #VALUE::#I2 = '' THEN LET VALUE::#I2 = '
     UPROC = "IF $RSTR(#VALUE::#I2,$SIZE(#VALUE::#I2)-1) EN
JUMP";
     PUTDATA;
   LABEL;
     UPROC = JUMP FILL;
   LABEL = NOALIGN;
     VALUE = #VALUE::#I2;
     START = 1,#I3;
     BREAK = " ";
     UPROC = LET I2 = #I2 + 1;
     PUTDATA;
     LOOP = #LOOP;
     XSTART = *+1,#I3;
   LABEL = FILL;
     UPROC = LET PRE = #CSYM#TYPE':';
   LABEL;
     VALUE = #PRE;
     START = 1,1;
     UPROC = LET PRE = #CSYM;
```

```
      PUTDATA;
      LOOP = #LOOP;
      XSTART = *+1,1;
  FRAME-ID = READIN;
    DIRECTION = INPUT;
    FRAME-DIM = 80,80;
    LABEL;
      START = 1,1;
      GETDATA = 5;
      UPROC = LET VAL = #CSYM#TYPE':';
      UPROC = IF $BREAK($UVAL,' ') = #VAL : IF #ALIGN = 'NTHEN
 JUMP NOALIGN;
      UPROC = IF $BREAK($UVAL,' ') = #VAL THEN JUMP VALUE;
      LOOP;
    LABEL;
      UPROC = RETURN;
    LABEL = VALUE;
      START = 1,1;
      GETDATA = 5;
      INPROC = $SQU;
      UPROC = LET T1 = $LOOPCT + 1;
      UPROC = LET VALUE::#T1 = $RSUB($CVAL,' ');
      LOOP;
    LABEL;
      UPROC = RETURN;
    LABEL = NOALIGN;
      START = 1,1;
      GETDATA = 5;
      UPROC = LET T1 = $LOOPCT + 1;
      UPROC = LET VALUE::#T1 = $RSUB($CVAL,' ');
      LOOP;
    LABEL;
      UPROC = RETURN;
  FORMAT-NAME = SETUP;
    ALLOCATE = USGS:SETUP;
    FRAME-NAME = VALUEOUT;
      FRAME-TYPE = XEQ;
    FRAME-NAME = READIN;
      FRAME-TYPE = XEQ;
```

VALUE

```
* VALUE  (03/29/84, 09:50:26, USGS     )
-
IF $ASK = '' THEN * ILLEGAL CALL, NO ARGUMENT
THEN RETURN
LET TYPE = $BREAK($ASK,' ')
LET TEMP = $RSUB($ASK,' ')
LET FM = $BREAK(#TEMP,' ')
LET TEMP = $RSUB(#TEMP,' ')
IF #FM = '' THEN LET FM = 'COMMENT'
LET ALIGN = $BREAK(#TEMP,' ')
IF #ALIGN = '' THEN LET ALIGN = 'ALIGN'
IF $MATCH(#ALIGN,'SINGLE','ALIGN','NOALIGN') = 0 : /* #ALGAL ARGUM
THEN RETURN
LET TEMP = $RSUB(#TEMP,' ')
LET CSYM = $BREAK(#TEMP,' ')
IF $MATCH(#CSYM,'C$','C*') = 0 THEN /* '#CSYM' ILLEGAL VA
THEN RETURN
SET MODE SHORT NOWARN
SHOW EVAL ' '
CMS SET CMSTYPE HT
/STATE #FN #TYPE A
IF $RC = 0 THEN /ERASE #TYPE COMMENT A
CMS SET CMSTYPE RT
SET WDSR LAST
IF $WDSR > 0 : CLR ACT
++VALUE
LET T1 = 0
-
IF #TYPE = 'DATE' THEN JUMP DO.DATE
IF #TYPE = 'ALPHA' THEN JUMP DO.ALPHA
/..PROMPT SETUP.COMM.#TYPE  '#CSYM#TYPE:'
IF #V0 = '' THEN JUMP NULL.VALUE
LET V1 = $RSUB(#V0,' ')
IF $LSTR(#V0,1) ¬= '/' THEN JUMP PUT.VALUE
LET V0 = $RSTR(#V0,1)
IF #V1 = '' THEN LET V0 = #V0' '#FM' 'A
CMS SET CMSTYPE HT
/STATE #V0
LET RC = $RC
CMS SET CMSTYPE RT
IF #RC ¬= 0 THEN /* File '#V0' does not exist'
THEN JUMP VALUE
/USE #V0 CLR
IF #ALIGN = 'NOALIGN' THEN JUMP VALUE.WHAT
XEQ GLO FRAME READIN USI F/L
JUMP VALUE.IN
++PUT.VALUE
IF #TYPE = 'CLASS' THEN JUMP CLASSCONT
IF #TYPE ¬= 'INSTITUTION' THEN JUMP PROMPTCONT
IF #V1 ¬= '' THEN JUMP PROMPTCONT
IF $RECTEST(#V0) < 1 THEN /* INST CODE #V0 NOT FOUND
THEN JUMP ADD.INST
```

```
/REF #V0
LET V0 = $GETCVAL('INST-NAME',0,#V0)
LET I1 = 0
++NEXTADDR
LET TEMP = $GETCVAL('INST-ADDR',#I1,'')
IF #TEMP = '' THEN JUMP CLRREF
LET V0 = #V0', '#TEMP
LET I1 = #I1 + 1
JUMP NEXTADDR
++CLRREF
CLR REF
LET T1 = 1
LET VALUE::1 = #V0
JUMP VALUE.IN
++PROMPTCONT
LET T1 = #T1 + 1
LET VALUE::#T1 = #V0
++GET.VALUE
IF #ALIGN = 'SINGLE' THEN JUMP VALUE.IN
LET VAL = #CSYM$INSETL(' ',' ',$SIZE(#TYPE))':'
/..PROMPT SETUP.COMM.#TYPE '#VAL'
IF #V0 = '' THEN JUMP VALUE.IN
JUMP PUT.VALUE
++NULL.VALUE
LET T1 = 1
LET VALUE::1 = ' '
-IF #CSYM = 'C$' THEN LET VALUE::1 = ' ;'
IF $MATCH(#TYPE,'DATE','CLASS','PERSN','ALPHA','KEYWD') :UE::1 = '
++VALUE.IN
- format title
IN ACT CLN CLR XEQ GLO FRAME VALUEOUT
++VALUE.LIST
SHO EVAL ' '
LIST UNN
JUMP VALUE.WHAT
++VALUE.BREAK
SHO EVAL ' '
SHOW EVAL '-Make needed changes.'
BREAK XEQ
XEQ GLO FRAME READIN USI F/L
IN ACT CLN CLR XEQ GLO FRAME VALUEOUT
SHO EVAL ' '
- output to wylbur active
- display and break
- read in new or possibly modified title
++VALUE.WHAT
SHO EVAL ' '
..PROMPT SETUP.COMM.WHAT  'Now What? (List/Next/edit/redo) '
IF #V0 = '' THEN JUMP VALUE.DONE
IF $PMATCH(#V0,L?IST) THEN JUMP VALUE.LISTIT
IF $PMATCH(#V0,N?EXT) THEN JUMP VALUE.DONE
IF $PMATCH(#V0,E?DIT) THEN JUMP VALUE.BREAK
IF $PMATCH(#V0,R?EDO) THEN JUMP VALUE
/* '#V0' unrecognized response
```

```
JUMP VALUE.WHAT
++VALUE.DONE
LET VAL = $LSTR(#TYPE,8)
/SAVE #FN #VAL A REP
-
++NO.VALUE
RETURN
-
++VALUE.LISTIT
SHOW EVAL ' '
LIST UNN
SHOW EVAL ' '
JUMP VALUE.WHAT
-
++NOCONT
RETURN JUMP NOCONT
-
++DO.ALPHA
SHO EVAL ' '
..PROMPT SETUP.INDX.BDATE 'Beginning Date   '
LET VALUE = #V0'; '
..PROMPT SETUP.INDX.EDATE 'Ending date      '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.BLAT  'Bottom Latitude '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.TLAT  'Top Latitude     '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.LLAT  'Left Longitude   '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.RLAT  'Right Longitude '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.CONT  'Contract number '
LET VALUE = #VALUE||#V0'; '
..PROMPT SETUP.INDX.REFN  'Data ref number '
LET VALUE = #VALUE||#V0'; '
LET VALUE::1 = #VALUE
LET T1 = 1
JUMP VALUE.IN
-
++DO.DATE
SHOW EVAL ' '
..PROMPT SETUP.DATE.DATE 'Submitted date    '
LET VALUE = #V0'; '
..PROMPT SETUP.DATE.HOLD '# months holding '
LET VALUE = #VALUE||#V0'; '
LET VALUE = #VALUE||#FN'; '
LET VALUE::1 = #VALUE
LET T1 = 1
JUMP VALUE.IN
-
++CLASSCONT
LET T1 = 1
LET VALUE::1 = #V0
IF $RSTR(#VALUE::1,$SIZE(#VALUE::1)-1) ¬= ';' THEN LET VA #VALUE::
```

```
LET VAL = 'C$SUBCL:'
++CLASSCONT2
/..PROMPT SETUP.COMM.#TYPE '#VAL'
IF #V0 = '' THEN JUMP VALUE.IN
LET VALUE::1 = #VALUE::1' '#V0
IF $RSTR(#VALUE::1,$SIZE(#VALUE::1)-1) ¬= ';' THEN LET VA #VALUE::
JUMP CLASSCONT2
-
++ADD.INST
SHOW EVAL ' '
..PROMPT ADD.INST 'Add inst code? (Yes/No,CR=No) '
SHOW EVAL ' '
IF #V0 = '' THEN JUMP VALUE
IF $PMATCH(#V0,N?O) > 0 THEN JUMP VALUE
IF $PMATCH(#V0,Y?ES,O?K) = 0 THEN * Please respond YES or
THEN JUMP ADD.INST
SET NOSTOP
ADD
IF $NO : THEN SET STOP
THEN JUMP ADD.INST
SET STOP
JUMP VALUE
```

Part 5

REFERENCES

IBM, Virtual Machine/System Product: Introduction, GC19-6200-1, IBM, 1982.

IBM, Virtual Machine/System Product: CMS Command and Macro Reference, SC19-6209, IBM, 1983.

IBM, Virtual Machine/System Product: System Product Interpreter Users Guide, SC24-5238, IBM, 1983.

Lee, W. H. K., Scharre, D. L. and Crane, G. R, A Computer-based System for Organizing Earthquake-related Data, USGS Open-file Report 83-518, 1983.

Stanford University, VM-WYLBUR Editor Mod05, Stanford University, 1980.

Stanford University, A Guide to Data Base Development -- A SPIRES Primer, Stanford University, 1984.