

ornl

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0315117 6

ORNL/TM-11555

**Characteristics Data Base:
Programmer's Guide to the
LWR Quantities Data Base**

ORNL LIBRARY LIBRARY
MARTIN MARIETTA LIBRARIES
CHARACTERISTICS
LWR QUANTITIES
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish to make a copy of this report,
please send a request to your local
branch office or library.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

invited to DOE and DOE contractors from the Office of Environmental Management, P.O. Box 62, Oak Ridge, TN 37831, telephone (423) 576-2801, FTS 529-2801.

Awards to the public from the National Technical Information Service, Department of Commerce, 5285 Port Royal Rd., Springfield, Virginia 22161.
NTIS price codes—Printed Copy, A08 Microfiche, A07

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or trade name, trademark, manufacturer, or otherwise, does not constitute an endorsement or imply its endorsement, recommendation, or favoritism by the United States Government or any agency thereof. The views and conclusions expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Chemical Technology Division

CHARACTERISTICS DATA BASE: PROGRAMMER'S GUIDE
TO THE LWR QUANTITIES DATA BASE

K. E. Jones

DataPhile, Inc.
Knoxville, Tennessee

R. S. Moore

Automated Sciences Group, Inc.
Oak Ridge, Tennessee

Date Published - August 1990

NOTICE This document contains information of a preliminary nature.
It is subject to revision or correction and therefore does not represent a
final report.

Report prepared by
DataPhile, Inc., Knoxville, TN
and

Automated Sciences Group, Inc., Oak Ridge, TN
under Contract DE-AC05-86OR21642

for
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6285
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
Office of Civilian Radioactive Waste Management
under contract DE-AC05-84OR21400



3 4456 0315117 6

TABLE OF CONTENTS

1.0 INTRODUCTION	1
2.0 SYSTEM DEVELOPMENT	3
2.1 PROGRAM START-UP	3
2.2 RESOURCE REQUIREMENTS	3
2.3 DEVELOPMENT ENVIRONMENT AND TOOLS	3
2.3.1 Clipper Compiler	4
2.3.2 Microsoft Linker	4
2.3.3 SAYWHAT Screen Generator	5
2.3.4 OverLay()	5
2.3.5 Microsoft C Compiler	5
2.4 MISCELLANEOUS	6
2.4.1 HDISPLAY.EXE	6
2.4.2 DBPRINT.EXE	7
3.0 MENU FLOW CHART	8
4.0 DATA FILE STRUCTURES	10
5.0 PROGRAMS	21
5.1 CLIPPER PROGRAMS	21
5.1.1 QTYMENU Program	22
5.1.4 GENRPT PROGRAM	24
5.1.5 FIXNDX Program	25
5.2 PROGRAMS USED TO INSTALL THE QUANTITIES DATABASE	25
APPENDIX A PROGRAM LISTINGS	A-1

PREFACE

The LWR Quantities Data Base is one of the important elements of the Characteristics Data Base (CDB), which provides the detailed technical characteristics of potential repository wastes.* The User's Guide to the LWR Quantities Data Base has been published (see reference 1 on page 1), and this report provides the corresponding Programmer's Guide. Other PC data bases and guides available through this task program are:

LWR Radiological Data Base

LWR Assemblies Data Base

LWR NFA Hardware Data Base

High-Level Waste Data Base

LWR Serial Numbers Data Base

Karl J. Notz
Task Manager
CDB Program

*"Characteristics of Spent Fuel, High-Level Waste, and Other Radioactive Wastes Which May Require Long-Term Isolation," DOE/RW-0184, Volumes 1-6 (December 1987) and Volumes 7-8 (June 1988).

Abstract

The LWR Quantities Data Base is a menu-driven PC data base developed as part of OCRWM's waste, technical data base on the characteristics of potential repository wastes, which also includes non-LWR spent fuel, high-level and other materials. This programmer's guide completes the documentation for the LWR Quantities Data Base, the user's guide having been published previously. The PC data base itself may be requested from the Oak Ridge National Laboratory, using the order form provided in Volume 1 of publication DOE/RW-0184.

1.0 INTRODUCTION

The LWR Spent Fuel Quantities Database System (Quantities Database) contains detailed data about LWR spent fuel assemblies in the United States. The database includes data on both historically discharged assemblies and on projected assembly usage through the year 2037. The historical data includes the year the assemblies were discharged, the pool where they are currently stored, their assembly class, assembly type, and approximate burnup, weight, and enrichment. The projected data includes the same data fields, excluding the assembly type and the storage pool. The data base is distributed on floppy disks to people in the nuclear industry to assist in planning for the permanent nuclear waste repository.

The Quantities Database is menu-driven and very easy to use. People with little or no computer expertise should find it easy to produce reports on the data of interest. For instructions on using the system, please see "User's Guide to the LWR Quantities Database System."¹ Those knowledgeable in PC databases can also take the data files and perform additional manipulations to obtain other reports and/or graphs.

This document describes the design and development of the Quantities Database. It provides a complete description of the data file structures and an outline of the major code modules. It serves as a reference for a programmer maintaining the system, or for others interested in the technical detail of this database. The data file structures, in particular, will be useful for those interested in personalized applications programming using the data.

The Quantities Database was originally developed in 1987, and contained historical data through Dec. 31, 1985 and projected data from Pacific Northwest Laboratories (PNL) through 2020. The system has been subsequently updated as more current data has become available, and new reporting capabilities have been added as needed. The current version of the Quantities Database contains historical data through Dec. 31, 1988, and projected data through the year 2037. Both historical and projected data are obtained from the Energy Information Administration (EIA). For projected data, the No New Orders Case with extended burnup is used.

EIA obtains the data from the utility companies via the RW-859 survey form. It evaluates and standardizes the data somewhat and distributes the data base on magnetic tape. The

¹User's Guide to the LWR Quantities Database System, DOE/RW-1084, Appendix 2D, Oak Ridge National Laboratory, Oak Ridge, TN, December 1987.

data are downloaded to a PC and formatted into dBase III Plus² data files. Another document will be written in 1990 to describe this process in detail. We expect to update the Quantities Database on an annual basis, as new historical data and revised projections become available.

²dBase III Plus is a product of Ashton-Tate Corporation.

2.0 SYSTEM DEVELOPMENT

2.1 PROGRAM START-UP

The program is started by typing the command QTY. This executes a DOS batch file that performs these operations:

1. Changes to the directory where the data and programs are stored.
2. Loads VIDPOP.COM, a memory resident screen manager³. More will be said about the screen generator in Section 2.3, Development Tools.
3. Runs the executable module, START.EXE. This starts the program, and the user can select various reports until he chooses to exit.
4. Removes VIDPOP.COM from memory.
5. Returns to the directory that was the default at step 1.

2.2 RESOURCE REQUIREMENTS

The Quantities Database system requires the following computer resources.

1. 400K of available memory (RAM).
2. 2.7 Megabytes of disk space.
3. DOS 2.1 or above.

The system is distributed as an executable module with all necessary auxiliary programs. You do not need to have dBase III Plus or any additional software to run the system.

2.3 DEVELOPMENT ENVIRONMENT AND TOOLS

This section describes briefly the different tools used to produce the Quantities Database system. It also includes some tips useful to reconstruct the programs. The primary software used in development is the Clipper compiler, but several other tools are used to provide additional features.

³VIDPOP.COM is a part of the screen generator SAYWHAT?!, produced by The Research Group.

2.3.1 Clipper Compiler⁴

The Quantities Database is written in Clipper, summer 1987 version. Clipper is a compiler for a language similar to dBase. Clipper was chosen because it provides the following advantages over the interpreted dBase languages:

1. Enhanced performance
2. The ability to distribute an executable module that will run independently. This avoids requiring users to purchase any additional software.
3. Some language extensions that cut programming time.

dBase III is also used to perform maintenance functions on the data tables.

In addition, the Make utility that is distributed with Clipper is used to keep track of system dependencies. This Make utility works like other such utilities, where a file is created which defines the system dependencies. The utility handles re-compiling such modules as necessary, after changes have been made to the source code. A file called QTY.MAK contains the information needed by the Make utility. After changing one or more modules, you can use this command to ensure that the program is updated:

MAKE QTY.MAK

This re-compiles all necessary files and links them together to form START.EXE, the main executable module.

2.3.2 Microsoft Linker⁵

The program modules are linked using the Microsoft Object Linker, Version 3.05. A linker response file, QTY.AR, contains the required parameters to link the system. Therefore, you can link the system using this command:

LINK @QTY.AR

This accesses all the necessary object modules and libraries needed to create the executable modules.

⁴Clipper is a product of Nantucket Corporation.

⁵Product of Microsoft Corporation.

2.3.3 SAYWHAT Screen Generator

The screens are produced using the SAYWHAT?! screen generator. Using this program, screens are easily designed and then saved as files with a .SQZ extension. These screen files are distributed as part of the Quantities Database. The SAYWHAT?! memory resident utility VIDPOP.COM must be installed in memory before the Quantities main program (START.EXE) is loaded. As mentioned in Section 2.1, the batch file QTY.BAT, distributed with the system, loads VIDPOP.COM before START.EXE. It also removes VIDPOP.COM from memory when the user exits the system, so that the memory will be freed for use by other programs.

There are two small assembly language programs needed to work with SAYWHAT?!. The first one, GETCHOIC, is distributed with SAYWHAT?!. Both the assembly language code and the object module are distributed. It handles obtaining the user's choice from a moving bar menu. The second one, CLIPPOP, handles sending instructions to the memory resident VIDPOP. CDB Programming staff wrote the second one and assembled it using Microsoft Assembler. The object modules are linked in with the other compiled modules to form START.EXE.

2.3.4 OverLay()⁶

OverLay() is a collection of functions that can be linked in with other Clipper programs. It is used in this system to free up memory so that the user can view his selected data on the screen. OverLay() frees memory by writing the main executable program to disk, allowing external programs to use that memory. When the external programs exit, the main program is re-loaded into memory, and execution continues. This technique allows the user to view a large report on the screen. The functions are supplied in a library which is linked with the other components to form the main executable program, START.EXE.

2.3.5 Microsoft C Compiler⁷

Some auxiliary programs used by the system are written in the C computer language and compiled using the Microsoft C Compiler Version 5.0. The C source code for these programs is included with the Clipper source code in the disks that accompany this document. Two of the programs, HDISPLAY.EXE and DBPRINT.EXE are described in the following sections, since they are used extensively in the Quantities Database System.

⁶OverLay() is a product of Gambit Software.

⁷Product of Microsoft Corporation.

2.4 MISCELLANEOUS

This section describes miscellaneous tools and utilities used in the Quantities Database.

2.4.1 HDISPLAY.EXE

This program is used to display a text file on the PC screen. It allows the user to browse through the file using the arrow keys. The Quantities program runs this program as an external program to display your reports on the screen, using Clipper's RUN command. HDISPLAY allows you to browse through the data, rather than scrolling it off the screen where you would be unable to review parts you had already seen. The program is written in Microsoft C, and the source code is included with the source code for the Quantities system.

To use this program, you would issue this command.

HDISPLAY FILENAME.EXT [options]

where FILENAME.EXT is any valid DOS filename and extension. HDISPLAY first looks for a second file with the same name and an extension of .HDG. It uses this file as a heading, and the contents of this file are displayed across the top portion of the screen. The heading file can have at most 10 lines. Then the contents of the primary text file are displayed underneath the heading lines, filling up the screen. When the user presses the arrow keys to scroll the text file, only the bottom portion of the file is scrolled; the top heading portion remains fixed. Creating a separate heading file avoids having to count lines and include page breaks during report generations, because it is all done after the database program has formatted the data lines for the report.

The options for the HDISPLAY program are listed below. Each option must be followed by a space, and then a number, **n**. You must use lower case for the option letters.

-h Heading option. The next **n** lines will be considered the heading; they will be displayed on the top of the screen, with the remainder of the file in a window below it. The heading part is not scrolled with the rest of the file.

-l Line length option. The line length will be **n**. If no line length is given, it will default to 80.

-t Trash option. The first **n** lines will be discarded.

2.4.2 DBPRINT.EXE

This utility program is very similar to HDISPLAY.EXE, described in the previous section. It is used to send text files to the printer rather than the screen. Like HDISPLAY, it looks for a heading file first, and displays the heading lines at the top of the page. It also numbers the pages. The program was written in Microsoft C, and the source code is included with the source code for the Quantities system.

DBPRINT has the following options. Each option must be followed by a space, and then a number, *n*. You must use lower case for the option letters.

-l Line length option. The line length will be *n*. If no line length is given, it will default to 80.

-p Page length option. The number of lines on a page will be *n*. If no page length is given, it will default to 57. This option is useful if you want to print reports in landscape mode. You can set the page length to be 44, and the heading will be correctly printed at the top of each page.

3.0 MENU FLOW CHART

Figure 1 outlines the major procedures that correspond to menu options.

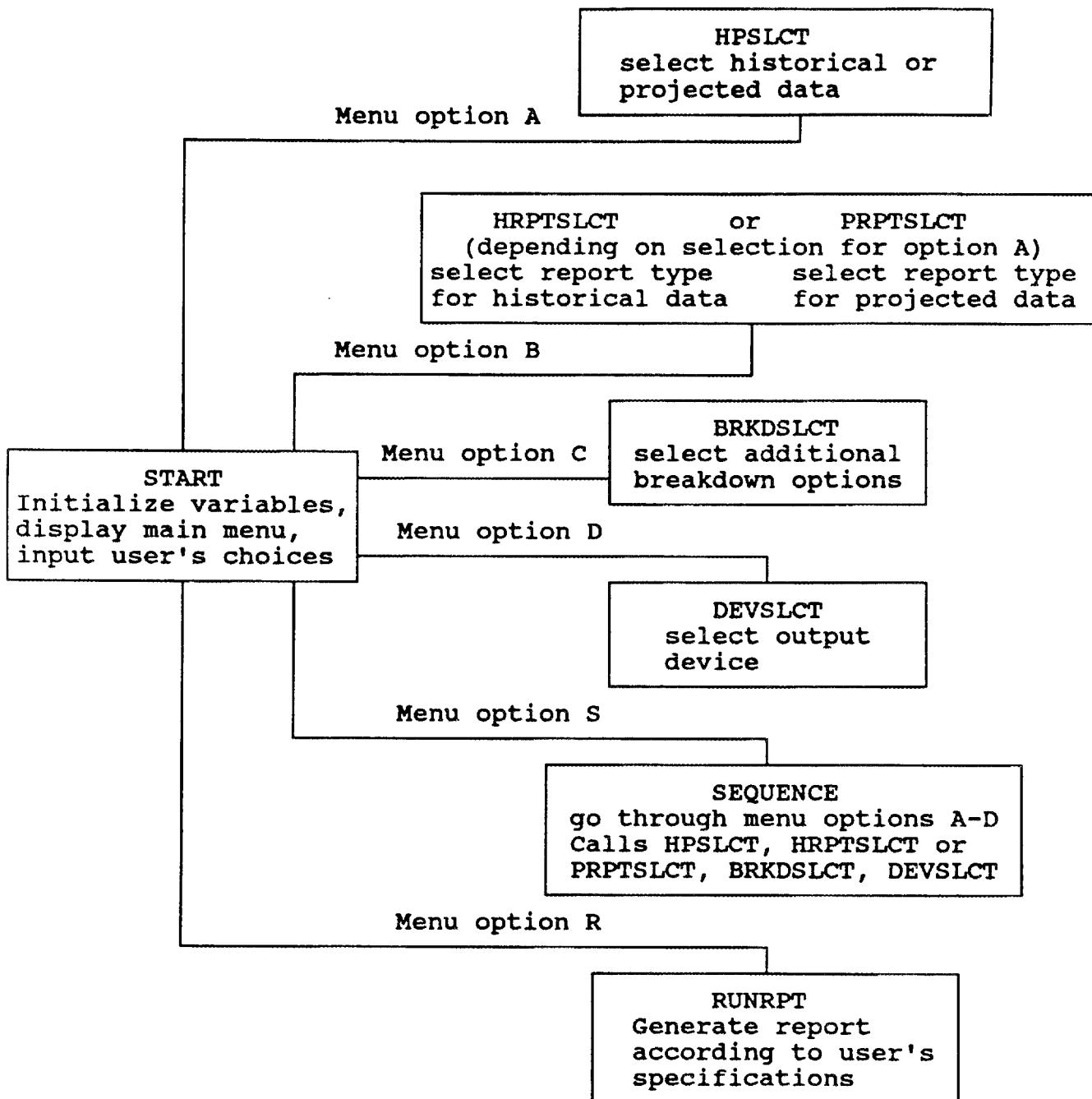


Figure 1. Procedures for Main Menu Options

The procedures HPSLCT, HRPTSLCT, PRPTSLCT, BRKDSLCT, DEVSLCT and SEQUENCE are contained in the file QTYMENU.PRG.

This diagram only shows the main procedures called when the user chooses a menu option. There are several other small subroutines that are also called in performing these functions. They are usually fairly short and simple and easy to understand. Section 5.0, PROGRAMS, will describe some of these additional procedures in detail, particularly the procedures that generate the reports from the database.

4.0 DATA FILE STRUCTURES

This section lists all the data files (extension of DBF) and gives a detailed description of each file. A few comments about the overall plan of the data files helps to understand the detailed listings, and eliminates the need to repeat the same comments for each individual file. Since each file contains one relational database table, we will use the words "file" and "table" interchangeably.

There are basically four files that contain the data about the discharged assemblies. These files are listed below.

QTY.DBF	Historically discharged assembly data used in preparing specific reports; that is, reports about a selected reactor, or utility, or assembly type, etc.
PQTY.DBF	The same as QTY.DBF, but for projected data.
SUMMARY.DBF	Historically discharged assembly data used in preparing general reports; that is, reports about all reactors, or all utilities, or all assembly types, etc.
PSUMMARY.DBF	The same as SUMMARY.DBF, but for projected data.

The tables for projected data are very similar to those for historical data; the first character of their name is 'P' for Projected.

The rest of the tables are generally reference tables, that is, they are used to look up codes for reactors, or assembly types, etc., used in the above data tables. The reference tables are also used to show the user a list of items from which he can select for a specific report. There are two sets of these reference tables, one set for historical data and one set for projected data. This is necessary so that only the appropriate names are shown, depending on whether the user has already selected historical or projected data. As with the primary data tables, the tables for projected data have a name very similar to the tables for the historical data, but with a 'P' as the first character.

There is one additional feature in the tables used to store projected data: they contain a field named "CASE" that could be used to identify between different projection cases. The original version of the Quantities Database contained data for two projection cases, and the user could select either case for generating reports. Therefore, the CASE field was necessary to differentiate between data for these different cases.

The current version of the Quantities Database contains only one projection case, so the field is not really necessary. In all tables, the CASE field is always set to one. The field

was left in the tables, so that we could easily add other cases in future revisions. This would save on development time, if additional projection cases are needed.

A detailed listing of each file structure is given below. For each file, you will find a brief comment on what the file contains, then a listing that describes each field in the file, followed by a description of all indices used with that file.

ASSMTYPE.DBF

Contains the assembly type codes and names for historical data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
ASSMTYPE	C5 C20	Assembly type code, In general, the first two character denote the array size, and the next character denotes the vendor, and the last two characters are for the model
NAME	C25	Name of assembly type
INDEX NAME	INDEXED ON	
ASSMTYPE ASSMNAME	ASSMTYPE UPPER(NAME)	

ASSMCLAS.DBF

Contains the assembly class codes and names for historical data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
ASSMCLAS	C3	Assembly class code
NAME	C25	Name of assembly type

INDEX NAME INDEXED ON

ASSMCLAS
CLASNAME ASSMCLAS
 UPPER(NAME)

PASSMCLS.DBF

Contains the assembly class codes and names for projected data. It is used the same as ASSMCLAS.DBF, but for projected data rather than historical data.

FIELD NAME	TYPE/SIZE	DEFINITION
ASSMCLAS	C3	Assembly class code
NAME	C25	Name of assembly type
CASE	C1	Identifies which projection case

INDEX NAME INDEXED ON

PASSMCLS
PACNAME CASE + ASSMCLAS
 CASE + UPPER(NAME)

POOL.DBF

Contains the pool codes and names for historical data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
POOL	C4	Pool code; standard INIS codes
NAME	C20	Name of pool

INDEX NAME INDEXED ON

POOL
POOLNAME POOL
 UPPER(NAME)

PQTY.DBF

Contains data from projections of assembly use. This table is used in generating the specific reports, that is, detailed reports for 1 particular reactor, or utility, etc.

FIELD NAME	TYPE/SIZE	DEFINITION
REACTYPE	C1	Reactor type; B->BWR, P->PWR
UTILITY	C2	Utility code
INIS	C4	Reactor INIS code
DISCHYEAR	C4	Year the assembly was discharged
ASSMCLAS	C3	Assembly class code
NUMASSM	C6	Number of assemblies
AVGBURN	C5	Average burnup for this group of assemblies
WEIGHT	N9.3	Average weight of uranium for this group of assemblies
ENRICH	N5.3	Average percent enrichment for this group of assemblies
BURNBIN	N2	Burnup bin, category of these assemblies based on burnup (AVGBURN)
CASE	C1	Identifies which projection case

INDEX NAME	INDEXED ON
PQRTYPE	CASE + REACTYPE + STR(DISCHYEAR,4) + STR(BURNBIN,2)+INIS
PQINIS	CASE + INIS + STR(DISCHYEAR,4) + STR(BURNBIN,2)
PQACLASS	CASE + ASSMCLAS + STR(DISCHYEAR,4) + STR(BURNBIN,2)
PQUTIL	CASE + UTILITY + STR(DISCHYEAR,4) + STR(BURNBIN,2)
PQYEAR	CASE + STR(DISCHYEAR,4) + STR(BURNBIN,2)

PREACTOR.DBF

Contains the reactor codes, names, and reactor types for projected data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
RECODE	C4	Reactor code; standard INIS codes
NAME	C20	Name of reactor
REACTYPE	C3	Reactor type; BWR or PWR
CASE	C1	Identifies which projection case

INDEX NAME	INDEXED ON
PREACTOR	CASE + RECODE
PRECNAME	CASE + UPPER(NAME)

PREACTYP.DBF

Contains the reactor type codes and names for projected data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
REACTYPE	C3	Reactor type; BWR or PWR
NAME	C20	Name of reactor type
CASE	C1	Identifies which projection case
INDEX NAME	INDEXED ON	
PRACTYP PRTNAME	CASE + REACTYPE CASE + NAME	

PSUMMARY.DBF

Contains summarized data from projections of assembly use. This table is used in generating the general reports, that is, reports that show all assemblies sorted by reactor, or by utility, etc.

FIELD NAME	TYPE/SIZE	DEFINITION
CLASS	C2	How this information was summarized: AL-->All Assemblies AC-->Assembly Class RE-->Reactor RT-->Reactor Type
PRIMNAME	C5	Code for which entity (reactor, pool, etc.)
DISCHYEAR	N4	Discharge year
NUMASSM	N5	Total number of assemblies
AVGBURN	N7	Average burnup for this group of assemblies

WEIGHT	N9.3	Average weight of uranium for this group of assemblies
AVGENRI	N6.3	Average percent enrichment for this group of assemblies
CASE	C1	Identifies which projection case
BURNBIN	N2	Burnup bin, category of these assemblies based on burnup (AVGBURN)
NAME	C25	Full name of the entity
INDEX NAME	INDEXED ON	
PSUMMARY		CASE + CLASS + NAME + STR(DISCHYEAR,4)

PUTILITY.DBF

Contains the utility codes and names for projected data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
UTILCODE	C2	Code for the utility; first 2 letters of reactor INIS code
NAME	C20	Name of reactor
CASE	C1	Identifies which projection case

INDEX NAME	INDEXED ON
PUTILITY	CASE + UTILCODE
PUTLNAME	CASE + UPPER(NAME)

QTY.DBF

Contains data on historically discharged assemblies. This table is used in generating the specific reports, that is, detailed reports for 1 particular reactor, or utility, etc.

FIELD NAME	TYPE/SIZE	DEFINITION
REACTYPE	C1	Reactor type; B-->BWR, P-->PWR
UTILITY	C2	Utility code
INIS	C4	Reactor INIS code
DISCHYEAR	C4	Year the assembly was discharged
ASSMTYPE	C5	Assembly type code
ASSMCLAS	C3	Assembly class code
POOL	C5	Pool code where assembly is currently stored
NUMASSM	N6	Number of assemblies
AVGBURN	N5	Average burnup for this group of assemblies
WEIGHT	N9.3	Average weight of uranium for this group of assemblies
ENRICH	N6.3	Average percent enrichment for this group of assemblies
BURNBIN	N2	Burnup bin, category of these assemblies based on burnup (AVGBURN)
DEFECT	C3	EIA defect code. If this field is blank, the assemblies are not defective

INDEX NAME	INDEXED ON
QRTYPE	REACTYPE + STR(DISCHYEAR,4) + STR(BURNUP,2) + INIS + POOL + ASSMTYPE
QINIS	INIS + STR(DISCHYEAR,4) + STR(BURNBIN,2) + POOL + ASSMTYPE
QPOOL	POOL + STR(DISCHYEAR,4) + STR(BURNBIN,2) + INIS + ASSMTYPE

REACTOR.DBF

Contains the reactor codes, names, and reactor types for historical data. It is used as a reference table. This table was supplied by Scott Moore of ASG. The Quantities Database uses only the Recode, Name, and Reactype fields. The other fields were left in for possible future use.

FIELD NAME	TYPE/SIZE	DEFINITION
RECODE	C4	Reactor code; standard INIS codes
NAME	C20	Name of reactor
DOCKET	C6	Federal docket number for reactor regulated by NRC
REACTYPE	C3	Reactor type; BWR or PWR
BWR_PROD	C1	General Electric product line
CLASS	C2	Assembly class code for reactor

INDEX NAME	INDEXED ON
PREACTOR	RECODE
RECNAME	UPPER(NAME)

REACTYPE.DBF

Contains the reactor type codes and names for historical data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
REACTYPE	C3	Reactor type; BWR or PWR
NAME	C20	Name of reactor type
INDEX NAME	INDEXED ON	
REACTYP RTNAME	REACTYP NAME	

SUMMARY.DBF

Contains summarized data on historically discharged assemblies. This table is used in generating the general reports, that is, reports that show all assemblies sorted by reactor, or by utility, etc.

FIELD NAME	TYPE/SIZE	DEFINITION
CLASS	C2	How this information was summarized: AL-->All Assemblies AT-->Assembly Type AC-->Assembly Class PO-->Pool RE-->Reactor RT-->Reactor Type
PRIMNAME	C5	Code for which entity (reactor, pool, etc.)
DISCHYEAR	N4	Discharge year
NUMASSM	N5	Total number of assemblies
DEFASSM	N5	Number of defective assemblies

AVGBURN	N7	Average burnup for this group of assemblies
WEIGHT	N9.3	Average weight of uranium for this group of assemblies
AVGENRI	N6.3	Average percent enrichment for this group of assemblies
BURNBIN	N2	Burnup bin, category of these assemblies based on burnup (AVGBURN)
NAME	C25	Full name of the entity
INDEX NAME	INDEXED ON	
SUMMARY	CLASS + NAME + STR(DISCHYEAR,4)	

UTILITY.DBF

Contains the utility codes and names for historical data. It is used as a reference table.

FIELD NAME	TYPE/SIZE	DEFINITION
UTILCODE	C2	Code for the utility; first 2 letters of reactor INIS code
NAME	C20	Name of reactor
INDEX NAME	INDEXED ON	
UTILITY	UTILCODE	
UTLNAME	UPPER(NAME)	

5.0 PROGRAMS

This section will explain briefly the major program modules in the Quantities Database. A complete source code listing is included as Appendix A of this document. The purpose of this section is to provide a overview of the program components and how they work together. This section will also include brief explanations for why the program was written and organized in this manner.

The first part, section 5.1, will describe the Clipper code that comprises the main program. The second section, 5.2, describes the batch files and other programs used to install the Quantities Database.

5.1 CLIPPER PROGRAMS

Figure 2 shows the major modules of the system.

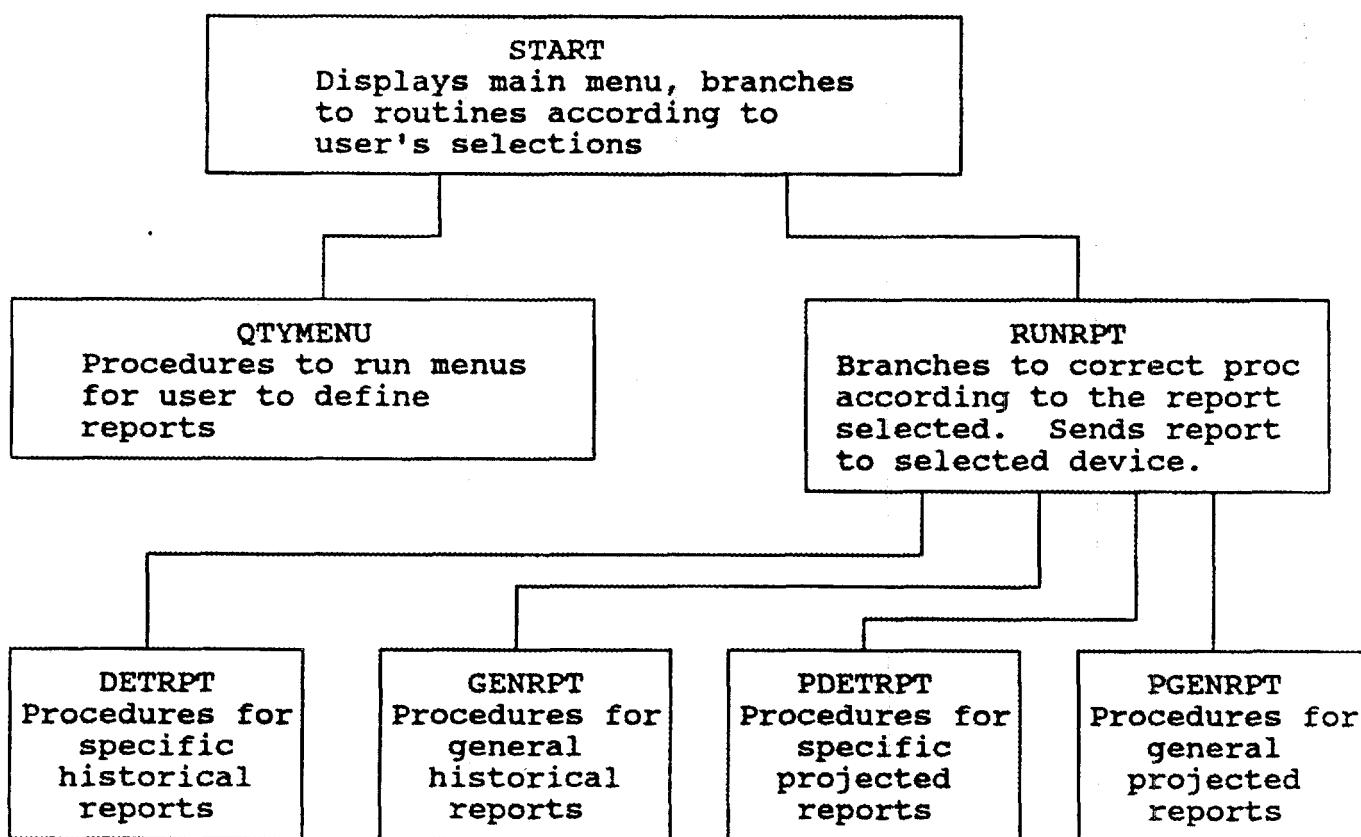


Figure 2. Major program modules

Execution begins in the module START. START sets up global variables and displays the main menu. It accepts the user's choices of what to do: either change a report definition or run the report. When the user chooses one of the options to change a report definition, control passes to the appropriate procedure in the procedure file QTYMENU. When the user chooses to run the report, control passes to the program RUNRPT. RUNRPT sets up the appropriate procedure file depending on what kind of report has been selected, and branches to the appropriate report set up routine in that procedure file.

The following sections describe some of the major procedures, particularly where some unusual logic or control has been used. For the most part, the procedures are quite straightforward and easy to understand. The Reports procedures have some complexities in design that are explained.

5.1.1 QTYMENU Program

As stated above, the QTYMENU file has procedures for handling the different menus that allow users to define the report. Users can choose which report type, breakdown options, and output device. The breakdown options vary with the report type selected, and if the data is historical or projected. More breakdown options are available for historical data.

Most of the procedures in this file are easy to understand. Basically, they display the choices available to the user on the bottom half of the screen, input and validate the selection, and then set the global variables that store the selection and are used by the report procedures. The different menu options available are shown in Figure 1.

5.1.2 RUNRPT Program

This program is called from START when the user decides to run the report. It sets some logical variables and sets the correct procedure file according to the user's selections for report type and historical/projected data. The four main reports procedure files are listed here.

DETRPT	Specific historical reports
GENRPT	General historical reports
PDETRPT	Specific projected reports
PGENRPT	General projected reports

The programs for projected and historical data are very similar; that is DETRPT is very similar to PDETRPT, and GENRPT is very similar to PGENDRPT. The major difference is that the programs for projected data use different data files, account for the different projection cases, and have slightly different fields.

The procedures in these files generate two text files. The file QTY.HDG has the heading lines. The file QTY.RPT has the body of the report. RUNRPT then routes these files to the user's selected output device. If the output device is the screen, the utility HDISPLAY is used to display the output on the screen. If the output device is the printer, then the utility DBPRINT is used to print the file. If the destination is a disk file, the report is copied to the file name specified by the user.

The next two sections describe the procedure files that produce the historical reports. The projected report procedures are very similar and are not explicitly covered.

5.1.3 DETRPT Program

This file contains the procedures to produce specific historical reports. The basic algorithm for producing the reports is as follows:

1. Open the files to be used. Typically these files will include the following.
 - > QTY, the main data file, indexed by the entity of interest. For example, if it's producing a report on utilities, it will use the index by utility, QUTIL.
 - > A reference database file that has the names of the entity. This is used to look up the codes that are stored in QTY.
2. Set variables that contain the key fields and values of the key fields.
3. Create a temporary data base, TEMP, that has only the pertinent records of interest from QTY.
4. Index this temporary file according to the user's breakdown options. For example, if he wanted the data broken down by assembly type and discharge year, the temporary file is indexed on these two fields.
5. Format the heading, and write it to a text file.
6. Format the body of the report from the data in the temporary database file TEMP. By trying to write a general procedure for many different breakdown options, these procedures that format the report are a little complicated. As an example, consider a specific report for the reactor Browns Ferry 1,

broken down by discharge year. To produce this report, the following steps would be taken:

- A. Go through the temporary file a year at a time. Sum the number of assemblies, and the number of assemblies times the burnups, weights, and enrichments (for weighted averages).
- B. Print out the line of data for one year.
- C. Repeat steps 1 and 2 until the end of file.
- D. Calculate weighted averages.
- E. Print out a totals line.

More complicated reports that have additional breakdown categories require extra loops to sum over categories, but the basic logic is the same.

5.1.4 GENRPT PROGRAM

This file contains the procedures to produce historical general reports. A general report would list all the reactors (or utilities, or storage pools, etc.) in the data file SUMMARY. The only breakdown options available are by discharge year. Because of the limited breakdown options, these procedures are much simpler than the ones for specific reports.

The basic logic of these procedures is closely tied to the structure of the data file SUMMARY. This file was described in section 4.0. This file is indexed by the fields CLASS and PRIMNAME. For example, all the reactor records (CLASS = 'RE') are in ascending order by INIS number.

Producing the general report is a fairly simple procedure. The logic is outlined below:

1. Set the CLASS variable according to the report type selected. Open the appropriate reference file.
2. Zero subtotal and grand total accumulators.
3. Seek the first record of the class.
4. For each year (each record is one year), add the numeric fields to the accumulators. If the report is broken down by discharge year, print a line for each year.

5. Repeat step 4 until the PRIMNAME field changes. At that point, we're finished with all records for that entity.
6. Calculate weighted averages and print the totals and averages for that group.
7. Repeat steps 4-6 until the CLASS field changes (or end of file).
8. Calculate and print the grand totals/averages.

For the report on all assemblies, or assemblies by reactor type, the report may also be broken down by burnup bin. This would require printing out the burnup bin lines, in addition to totaling the values for the accumulators.

5.1.5 FIXNDX Program

FIXNDX.PRG is a program that contains the code to create all indexes used in the system. It is compiled and linked to form a separate executable module, FIXNDX.EXE. This executable program is distributed with the Quantities Database. It is run as part of the installation procedure to create all the required index files. This program could also be useful if a power failure or some other problem damaged the index files on a user's machine. They could try running this FIXNDX program (just enter FIXNDX at the DOS prompt) to re-create the index files.

5.2 PROGRAMS USED TO INSTALL THE QUANTITIES DATABASE

INSTALL.BAT, a DOS batch file, is used to install the system onto the user's hard disk. To install the system, the user places the first distribution diskette in one of his floppy drives and issues the following command.

INSTALL C

where "C" is the letter of the hard disk where he wants to install the database. It can be any letter that denotes a disk drive. The batch file accepts the disk drive argument with a colon (such as "C:") or without it.

Basically, INSTALL.BAT is a straightforward batch file. It makes a subdirectory called QTY on the specified hard disk, copies the contents of the distribution diskettes into that subdirectory, and then runs FIXNDX.EXE, which creates all the indices for the system. We saved considerable space on the distribution diskettes by creating the index files at installation, rather than including them on the distribution diskettes.

In addition to the above basic operations, INSTALL.BAT performs two other tasks listed below.

1. Checks available disk space using a program DISKSP.EXE (written in C). If there is sufficient disk space available to install the system (2.7MB), the installation will proceed. If there's not enough disk space, a message is printed and the installation process is halted. This avoids problems and confusion that would occur if the user ran out of disk space part way through the installation.
2. Checks if the current directory on the specified disk drive is the root directory using the program CHROOT.EXE (written in C). If the directory is not the root, then CHROOT asks the user if he wants the new software appended into the current directory. If the user answers yes, then a subdirectory named QTY will be created a level below the current directory, and the Quantities system installed there. If the user answers no, then the subdirectory QTY will be created a level below the root directory, and the Quantities system installed there. This provides added flexibility in installing the system.

APPENDIX A -- PROGRAM LISTINGS DETRPT.PRG

1

```
1 ****
2 * DETERPT.PRG -- Procedures for the reports of the Quantities Data Base.      *
3 * The reports can be for historical or projected quantities. The historical*
4 * reports are further divided between general reports (all discharged      *
5 * assemblies for all utility, or all reactors, etc.) or specific reports   *
6 * (for 1 user-specified utility, or reactor, etc.).                         *
7 *
8 * 10/89 key Modified for new data format. The defect code is in the
9 * QTY record, we don't need to look at a separate file.
10 *
11 ****
12 * DetSetup. Setup parameters for a detailed report. Then call DetRpt1.  *
13 ****
14 *
15 *
16 Procedure DetSetup
17     parameter gotData
18
19 *if (debug)
20 *    set color to gr+/n
21 *    @ 21,0 clear
22 *    @ 21,10 say "In DetSetup. rptNum: "
23 *    @ 21,40 say str(rptNum,2)
24 *    wait "" to dummy
25 *endif
26
27 select 1
28 lookArea = "3"           && Set up lookup table here
29
30 do case
31     case rptNum = 8          && Utility
32         key1 = utilName
33         keyName = "substr(inis,1,2)"
34         use &dbfPath.qty index &dbfPath.qUtil
35         select 3                && Lookup area
36         use &dbfPath.utility index &dbfPath.utility
37
38     case rptNum = 9          && Reactor
39         key1 = reacName
40         keyName = "inis"
41         use &dbfPath.qty index &dbfPath.qInis
42         select 3                && Lookup area
43         use &dbfPath.reactor index &dbfPath.reactor
44
45     case rptNum = 10         && Storage Pool
46         key1 = poolName
47         keyName = "pool"
48         use &dbfPath.qty index &dbfPath.qPool
49         select 3                && Lookup area
50         use &dbfPath.pool index &dbfPath.pool
51
52     case rptNum = 11         && Assembly Type
53         key1 = assmTName
54         keyName = "assmType"
55         use &dbfPath.qty index &dbfPath.qAType
56         select 3                && Lookup area
57         use &dbfPath.assmType index &dbfPath.assmType
58
59     case rptNum = 12         && Assembly Class
60         key1 = assmCName
61         keyName = "assmClas"
62         use &dbfPath.qty index &dbfPath.qAClass
63         select 3                && Lookup area
64         use &dbfPath.assmClas index &dbfPath.assmClas
65
66     case rptNum = 13         && Reactor Type
67         key1 = substr(rtName,1,1)
68         keyName = "reactType"
69         use &dbfPath.qty index &dbfPath.qRType
70         select 3                && Lookup area
71         use &dbfPath.reactType index &dbfPath.reactType
```

APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG

2

```
72      if (debug)
73          @ 20,0 clear
74          @ 20,10 say "Top of DetSetup. key1: " + key1
75          @ 21,10 say "keyName: " + keyName
76          wait "" to dummy
77      endif
78
79  endcase
80
81  select 1
82  ** build temporary tables
83
84  gotData = .f.
85  defcData = .f.
86
87  do BldTemp with key1, keyName, gotData, defcData
88  if (debug)
89      @ 20,0 clear
90      @ 20,10 say "After BldTemp.  gotData: "
91      @ 20,40 say gotData picture 'Y'
92      wait "" to dummy
93      AltD()
94  endif
95
96  if (.not. gotData)
97      select 3
98      use
99  if (debug)
100     @ 20,0 clear
101     @ 20,10 say "Returning because of no data."
102     wait "" to dummy
103     AltD()
104  endif
105  return
106 endif
107
108 ** Create reports
109 if (debug)
110     @ 20,0 clear
111     @ 20,10 say "Back from BldTemp.  keyName: " + keyName
112     wait "" to dummy
113 endif
114
115 do DetRpt1 with key1, keyName, lookArea, defcData
116
117 select 3
118 use
119 return
120
121 ****
122 * BldTemp -- Build a temporary table with the records of interest for this *
123 * report.
124 ****
125
126 Procedure BldTemp
127     parameters key1, keyName, found, defcData
128
129 select 4
130 use temp
131 delete all
132 pack
133
134 select 1          && QTY
135 seek key1          && Any data for this key?
136 if (.not. found())
137 *   @ 21,0 clear
138 *   @ 21,10 say "Can't find key: " + key1
139 *   wait "" to dummy
140     return
141 else
142     found = .t.
```

APPENDIX A – PROGRAM LISTINGS
DETRPT.PRG

3

```

143 endif
144
145 do while (key1 = &keyName .and. .not. eof())
146   select 4                               && Temp
147   append blank
148   replace inis with qty->inis
149   replace dischYear with qty->dischYear
150   replace burnBin with qty->burnBin
151   replace pool with qty->pool
152   replace numAssm with qty->numAssm
153   replace avgBurn with qty->avgBurn
154   replace weight with qty->weight
155   replace reacType with qty->reacType
156   replace assmType with qty->assmType
157   replace assmClas with qty->assmClas
158   replace enrich with qty->enrich
159   replace defect with qty->defect
160
161   select 1                               && qty
162   skip
163   if (key1 <> &keyName)
164     exit
165   endif
166 enddo
167
168 ** Set up correct index
169 ndxKeys = keyName
170
171 if (by_Pool)
172   ndxKeys = ndxKeys + " + Pool"
173 endif
174
175 if (by_AssmT)
176   ndxKeys = ndxKeys + " + assmType"
177 endif
178
179 if (by_Year)
180   ndxKeys = ndxKeys + " + str(dischYear,4)"
181 endif
182
183 if (by_Bin)
184   ndxKeys = ndxKeys + " + str(dischYear,4) + str(burnBin,2)"
185 endif
186
187 select 4
188 index on &ndxKeys to temp
189 go top
190
191 return
192 ****
193 * DetRpt1. Produce a detailed reports. Called from the DetSetup routine. *
194 ****
195 ****
196
197 Procedure DetRpt1
198   parameters key1, keyName, lockArea, defcData
199
200 select 4                               && Temp
201 if (debug)
202   set color to gr+/n
203   @ 21,0 clear
204   @ 21,10 say "In DetRpt1. key1: " + key1
205   @ 22,10 say "keyName: " + keyName
206   wait "" to dummy
207 endif
208
209 select 4                               && Temp
210 go top
211
212 do DetHdg with key1, keyName          && Produce heading
213 set alternate to qty.rpt

```

**APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG**

4

```

214 set alternate on
215
216 ** Zero grand total accumulators
217 gt_NumAssm = 0
218 gt_DefAssm = 0
219 gt_Burn = 0
220 gt_Wt = 0
221 gt_Enri = 0
222 gt_AvgBurn = 0
223 gt_Wt = 0
224 gt_AvgEnri = 0
225
226 if (by_Pool .or. by_AssmT)
227 if (debug3)
228   @ 21,0 clear
229   @ 21,10 say "About to call GrpRpt."
230   wait "" to dummy
231 endif
232
233 do GrpRpt with key1, keyName
234 else
235 do OneGRpt with key1, keyName
236 ? ''
237 endif
238
239 ** Calculate averages
240 if (gt_NumAssm = 0)
241   gt_AvgBurn = 0
242   gt_AvgEnri = 0
243 else
244   gt_AvgBurn = round(gt_Burn/gt_NumAssm,0)
245   gt_AvgEnri = round(gt_Enri/gt_NumAssm,3)
246 endif
247
248 ** Print out totals line
249 line1 = " --- TOTALS "
250 if (by_Bin)
251   line1 = line1 + space(10) + str(gt_NumAssm,5) + space(11) +
252           str(gt_AvgBurn,6) + space(6) + str(gt_Wt,8,1) + space(7);
253           + str(gt_AvgEnri,5,3)
254 else
255   line1 = line1 + str(gt_NumAssm,5) + space(10)
256   if (noDefc1)
257     line1 = line1 + ' * '
258   else
259     if (gt_DefAssm > 0)
260       line1 = line1 + str(gt_DefAssm,4)
261     else
262       line1 = line1 + space(4)
263     endif
264   endif
265   line1 = line1 + space(6) + str(gt_AvgBurn,6) + space(6) + str(gt_Wt,8,1);
266           + space(8) + str(gt_AvgEnri,5,3)
267 endif
268 ? line1
269
270 ** Special case -- incorrect defective data for Commonwealth Edison
271 if ((noDefc1) .and. (.not. by_Bin))
272   ? ''
273   ? " Data on defective assemblies is not available for this report."
274 endif
275
276 return
277
278 ****
279 * GrpRpt -- Produce report for groups (of pools, or assembly type, or all *
280 * data if no subtotals were indicated. *
281 ****
282
283
284 Procedure GrpRpt

```

APPENDIX A – PROGRAM LISTINGS
DETRPT.PRG

5

```

285     parameters key, keyName
286
287     private key2Name, key2Fld
288
289     if (debug)
290         @ 20,0 clear
291         @ 20,10 say "In GrpRpt.  keyName:  " + keyName
292         wait "" to dummy
293     endif
294
295     ** Zero accumulators for grand totals
296     s2_NumAssm = 0
297     s2_DefAssm = 0
298     s2_Burn = 0
299     s2_Wt = 0
300     s2_Enri = 0
301
302     select 5
303     if (by_Pool)
304         use &dbfPath.pool index &dbfPath.pool
305         key2Fld = "Pool"
306         key2Name = "Pool"
307     endif
308
309     if (by_AssmT)
310         use &dbfPath.assmType index &dbfPath.assmType
311         key2Fld = "assmType"
312         key2Name = "Assem. Type"
313     endif
314
315     select 4          @@ temp
316     do while (.not. eof())
317
318     ** Zero Accumulators for this group
319     .   gt_NumAssm = 0
320     .   gt_DefAssm = 0
321     .   gt_Burn = 0
322     .   gt_Wt = 0
323     .   gt_Enri = 0
324
325     select 5          @@ Assmtype or pool
326     if (by_Pool)
327         key2 = temp->pool
328     endif
329
330     if (by_AssmT)
331         key2 = temp->assmType
332     endif
333
334     do DetHdg3 with key2, key2Name, "5"
335
336     if (debug3)
337         @ 21,0 clear
338         @ 21,10 say "In GrpRpt.  About to call OneGrpt.  key2:  " + key2
339         @ 22,10 say "key2Name:  " + key2Name
340         wait "" to dummy
341     endif
342
343     do OneGRpt with key2, key2Fld
344
345     ** Add group accumulators to grand totals
346     s2_NumAssm = s2_NumAssm + gt_NumAssm
347     s2_DefAssm = s2_DefAssm + gt_DefAssm
348     s2_Burn = s2_Burn + gt_Burn
349     s2_Wt = s2_Wt + gt_Wt
350     s2_Enri = s2_Enri + gt_Enri
351
352     gt_AvgBurn = round(gt_Burn/gt_NumAssm,0)
353     gt_AvgEnri = round(gt_Enri/gt_NumAssm,3)
354
355     ** Print out total line

```

APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG

6

```

356      ? ''
357      if (by_Pool)
358          line1 = "--Pool Totals      "
359      else
360          line1 = "--Assm. Type Totals  "
361      endif
362
363      if (by_Bin)
364          line1 = line1 + space(10) + str(gt_NumAssm,4) + space(11) +
365              str(gt_AvgBurn,6) + space(6) + str(gt_Wt,8,1) +
366              space(7) + str(gt_AvgEnri,5,3)
367      else
368          line1 = line1 + str(gt_NumAssm,4) + space(10)
369          if (noDefc1)
370              line1 = line1 + " * "           /* Special cases, no defective data
371          else
372              if (gt_DefAssm > 0)
373                  line1 = line1 + str(gt_DefAssm,4)
374              else
375                  line1 = line1 + space(4)
376              endif
377          endif
378          line1 = line1 + space(6) + str(gt_AvgBurn,6) + space(6) +
379              str(gt_Wt,8,1) + space(8) + str(gt_AvgEnri,5,3)
380      endif
381
382      ? line1
383      if (debug3)
384          @ 21,0 clear
385          @ 21,10 say "End of GrpRpt. Printing a totals line, and 2 blank ones."
386          wait "" to dummy
387      endif
388      ? ''
389      ? ''
390
391 enddo
392
393 gt_NumAssm = s2_NumAssm
394 gt_DefAssm = s2_DefAssm
395 gt_Burn = s2_Burn
396 gt_Wt = s2_Wt
397 gt_Enri = s2_Enri
398
399 select 5
400 use
401 return
402 ****
403 * OneGRpt -- Produce report for one group (of pools, or assembly type, or *
404 * all data if no subtotals were indicated. *
405 ****
406 ****
407
408 Procedure OneGRpt
409     parameters key2, key2Fld
410
411     if (debug)
412         @ 21,0 clear
413         @ 21,10 say "In OneGRpt. key2: " + key2
414         @ 22,10 say "key2Fld: " + key2Fld
415 *     @ 22,40 say "temp->&key2Fld: " + temp->&key2Fld
416         wait "" to dummy
417 *     AltD()
418     endif
419
420 do while (key2 = &key2Fld .and. .not. eof())
421
422 ** Zero sub total accumulators
423     st_NumAssm = 0
424     st_DefAssm = 0
425     st_Burn = 0
426     st_Wt = 0

```

APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG

7

```
427     st_Enri = 0
428     st_AvgBurn = 0
429     st_Wt = 0
430     st_AvgEnri = 0
431
432     ** Handle data for one discharge year
433     do OneYear with key2, key2Fld
434
435     ** Add year's totals to grand totals
436     gt_NumAssm = gt_NumAssm + st_NumAssm
437     gt_DefAssm = gt_DefAssm + st_DefAssm
438     gt_Burn = gt_Burn + st_Burn
439     gt_Wt = gt_Wt + st_Wt
440     gt_Enri = gt_Enri + st_Enri
441
442     select 4
443
444     if (key2 <> &key2Fld)      /* Won't re-evaluate condition in while loop
445         exit
446     endif
447     if (debug3)
448         set color to gr+/n
449         @ 23,0 clear
450         @ 23,10 say "End of loop in DetRpt1.  Key2:  " + key2
451         @ 24,10 say "&key2Fld:  "
452         @ 24,40 say &key2Fld
453         wait "" to dummy
454     endif
455     enddo
456     return
457
458 ****
459 * DetHdg.  Produce the heading for detailed reports.  Called from the      *
460 * DetRpt1 procedure.                                              *
461 ****
462
463 Procedure DetHdg
464     parameter key1, keyName
465
466     set alternate to qty.hdg
467     set alternate on
468
469     ?? title1
470     if (rptNum < 10)
471         strNum = str(rptNum,1)
472     else
473         strNum = str(rptNum,2)
474     endif
475
476     line1 = "Historical Data"
477     startPos = (80 - len(line1))/2
478     line1 = space(startPos) + line1
479     ? line1
480
481     line2 = ""
482     line1 = "Data Broken Down By:  "
483
484     select &clockArea
485     seek key1
486
487     if (by_Pool)
488         line1 = line1 + "Storage Pool, "
489     endif
490
491     if (by_AssmT)
492         line1 = line1 + "Assembly Type, "
493     endif
494
495     if (by_Year)
496         line1 = line1 + "Discharge Year"
497     endif
```

APPENDIX A -- PROGRAM LISTINGS
DETRPT.PRG

8

```

498 if (by_Bin)
499     line1 = line1 + "Discharge Year and Burnup Bin"
500 endif
501
502 if (.not. (by_Pool .or. by_Assmf .or. by_Year .or. by_Bin))
503     line1 = line1 + "No Subtotals"
504 endif
505
506 l = len(line1)
507 if substr(line1,l-1,2) = ', '
508     line1 = substr(line1,l,1-2)
509 endif
510
511 startPos = (80 - len(line1))/2
512 line1 = space(startPos) + line1
513 ? line1
514
515 line1 = hRTtitle&strNum + trim(Name)           && Put in specific name
516 startPos = (80 - len(line1))/2
517 line1 = space(startPos) + line1
518 ? line1
519
520 do DetHdg2                                && Get column headings
521
522 select 4                                    && Temp
523 return
524
525 ****
526 * DetHdg2. Produce the column headings for detailed reports. Called from *
527 * DetHdg procedure.
528 ****
529
530 Procedure DetHdg2
531
532 if (by_Bin)
533     ? space(49) + 'AVG' + space(10) + 'TOTAL' + space(7) + 'AVG'
534 else
535     if (by_Year)
536         ? space(47) + 'AVG' + space(11) + 'TOTAL' + space(8) + ' AVG'
537     else
538         ? space(47) + 'AVG' + space(11) + 'TOTAL' + space(8) + ' AVG'
539     endif
540 endif
541
542 if (by_Bin)
543     line1 = 'DISCHARGE' + space(7) + 'BURNUP' + space(9) + "NUMBER" +;
544         space(11) + "BURNUP" + space(7) + 'WEIGHT' + space(6) + " INIT ";
545     line2 = ' YEAR ' + space(7) + ' BIN ' + space(9) + "ASSMB" +;
546         space(10) + "(Mwd/MTIHM)" + space(4) + "(MTIHM)" + space(5);
547         + "ENRICH"
548 else
549     line1 = "NUMBER" + space(8) + "DEFEC." + space(5) + "BURNUP" + space(8) +;
550         'WEIGHT' + space(7) + " INIT "
551     line2 = "ASSMB" + space(9) + "ASSMB" + space(3) + "(Mwd/MTIHM)" + space(
5) +;
552         "(MTIHM)" + space(6) + "ENRICH"
553
554     if (by_Year)
555         line1 = 'DISCHARGE' + space(12) + line1
556         line2 = ' YEAR ' + space(12) + line2
557     else
558         line1 = space(21) + line1
559         line2 = space(21) + line2
560     endif
561 endif
562
563 ? line1
564 ? line2 + carbRet + lineFeed
565
566 return
567

```

APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG

9

```

568 ****
569 * DetHdg3 -- Print heading line for a particular group -- pool or assembly *
570 * type. Called from GrpRpt.
571 ****
572
573 Procedure DetHdg3
574     parameters key, keyName, area
575
576 *@ 21,0 clear
577 *@ 21,10 say "In DetHdg3. Looking for " + key
578 *wait "" to dummy
579
580 select &area
581 line1 = keyName + ":" +
582
583 seek key
584 if (found())
585     line1 = line1 + name
586 else
587     line1 = line1 + key
588 endif
589
590 ? line1
591 ?
592
593 select 4           ?? Temp
594 return
595
596 ****
597 * OneYear. Handle Historical Quantity data for one year. Add the year's *
598 * values to the subtotals. If the user wants to see the data by burnup bin,*
599 * print out detail lines.
600 ****
601
602 Procedure OneYear
603     parameters key2, key2Fld
604
605 curYear = dischYear
606 printYear = .f.
607
608 do while (key2 = &key2Fld .and. dischYear = curYear .and. .not. eof())
609
610 * @ 21,0 clear
611 * @ 21,10 say "At top of year loop. key2Fld: "
612 * @ 21,60 say &key2Fld
613 * @ 22,10 say "Burnup Bin: "
614 * @ 22,25 say burnbin
615 * @ 22,35 say "DischYear: "
616 * @ 22,55 say dischYear
617 * @ 23,10 say "curYear: "
618 * @ 23,25 say curYear
619 * 7 curYear = dischYear
620 * wait "" to dummy
621
622 ** Get totals for this burnup bin
623 curBin = burnBin
624 bb_NumAssm = 0
625 bb_DefAssm = 0
626 bb_Burn = 0
627 bb_Wt = 0
628 bb_Enri = 0
629
630 do while (key2 = &key2Fld .and. curBin = burnBin .and. ;
631         dischYear = curYear .and. .not. eof())
632 if (debug)
633     @ 20,0 clear
634     @ 20,10 say "At top of burnup bin loop in One Year. CurYear: " + str(curY
ear,4)
635     @ 21,10 say "numAssm: "
636     @ 21,25 say numAssm
637     @ 21,40 say "AvgWt: "

```

APPENDIX A -- PROGRAM LISTINGS
DETRPT.PRG

10

```

638    @ 21,55 say AvgWt
639    @ 22,10 say "enrich: "
640    @ 22,40 say enrich
641    wait "" to dummy
642  endif
643      bb_NumAssm = bb_NumAssm + numAssm
644      bb_Burn = bb_Burn + (numAssm * avgBurn)
645      bb_Wt = bb_Wt + weight
646      bb_Enri = bb_Enri + (numAssm * enrich)
647      if (defect <> space(3))
648          bb_DefAssm = bb_DefAssm + numAssm
649      endif
650      skip
651      if .not. (key2 = &key2Fld .and. curBin = burnBin .and. ;
652                  dischYear = curYear)
653          exit
654      endif
655  enddo          && Finished this burnup bin
656
657  if (by_Bin)
658      bb_AvgBurn = round(bb_Burn/bb_NumAssm,0)
659      bb_AvgEnri = round(bb_Enri/bb_NumAssm,3)
660
661      line1 = ' ' + str(curYear,4) + space(6)
662      if (curBin < 10)
663          strNum = str(curBin,1)
664      else
665          strNum = str(curBin,2)
666      endif
667
668      line1 = line1 + bb&strNum + space(8) + str(bb_NumAssm,5) + space(11) +;
669          str(bb_avgBurn,6) + space(6) * str(bb_Wt,8,1) + space(7);
670          + str(bb_avgEnri,5,3)
671      ? line1
672  endif
673
674  ** Add burnup bin totals to subtotals
675  st_NumAssm = st_NumAssm + bb_NumAssm
676  st_Burn = st_Burn + bb_Burn
677  st_Wt = st_Wt + bb_Wt
678  st_Enri = st_Enri + bb_Enri
679  st_DefAssm = st_DefAssm + bb_DefAssm
680
681  if .not. (key2 = &key2Fld .and. dischYear = curYear)
682      exit
683  endif
684 enddo
685
686  if (by_Year)          && Print lines subtotalized by year
687      st_AvgBurn = round(st_Burn/st_NumAssm,0)
688      st_AvgEnri = round(st_Enri/st_NumAssm,3)
689
690      line1 = ' ' + str(curYear,4) + space(16) + str(st_NumAssm,4)+;
691          space(10)
692      if (noDefc1)
693          line1 = line1 + " * "
694      else
695
696          if (st_DefAssm > 0)
697              line1 = line1 + str(st_DefAssm,4)
698          else
699              line1 = line1 + space(4)
700          endif
701      endif
702
703      line1 = line1 + space(6) + str(st_AvgBurn,6) + space(6) + str(st_Wt,8,1);
704          + space(8) + str(st_AvgEnri,5,3)
705      ? line1
706      if (debug3)
707          @ 21,0 clear
708          @ 21,10 say "End of OneYear. Printing a line of data for one year."

```

APPENDIX A - PROGRAM LISTINGS
DETRPT.PRG

11

```
709      wait "" to dummy
710      endif
711  endif
712
713  if (by_Bin)
714    if (debug3)
715      @ 21,0 clear
716      @ 21,10 say "End of OneYear.  Printing a blank line."
717      wait "" to dummy
718    endif
719    ?
720  endif
721  return
722
```

```

1 set color to gr+/b
2 clear
3 line1 = "LWR Quantities Database"
4 startPos = (80 - len(line1))/2
5 @ 1,startPos say line1
6 @ 2,startPos say "-----"
7
8 line1 = "Creating Index Files"
9 startPos = (80 - len(line1))/2
10 @ 4, startPos say line1
11
12 line1 = "Working on QTY database"
13 startPos = (80 - len(line1))/2
14 @ 10, startPos say line1
15
16 use qty
17
18 index on reacType + str(dischYear,4) + str(burnBin,2)+ inis + pool + assmType
;
19     to qRTyp
20 index on inis + str(dischYear,4) + str(burnBin,2)+ pool + assmType ;
21     to qinis
22 index on pool + str(dischYear,4) + str(burnBin,2)+ inis + assmType ;
23     to qPool
24 index on assmType + str(dischYear,4) + str(burnBin,2)+ inis + pool ;
25     to qAType
26 index on assmClas + str(dischYear,4) + str(burnBin,2)+ inis + pool ;
27     to qAClass
28 index on utility + str(dischYear,4) + str(burnBin,2)+ pool;
29     + assmType to qutil
30 index on str(dischYear,4) + str(burnBin,2) to qYear
31
32 line1 = "Working on Summary database"
33 startPos = (80 - len(line1))/2
34 @ 10, startPos say line1
35
36 use summary
37 index on class + name + str(dischYear,4) to summary
38
39 use reactor
40 index on recode to reactor
41 index on upper(name) to recName
42 use utility
43 index on utilcode to utility
44 index on upper(name) to utlname
45 use pool
46 index on pool to pool
47 index on upper(name) to poolname
48 use reactype
49 index on reactype to reactype
50 index on name to rtname
51 use assmtype
52 index on assmtype to assmtype
53 index on upper(name) to assmname
54 use assmclas
55 index on assmclas to assmclas
56 index on upper(name) to clasname
57
58 line1 = "Working on Projected QTY database"
59 startPos = (80 - len(line1))/2
60 @ 10, startPos say line1
61
62 use pqty
63
64 index on case + reactype + str(dischYear,4) + str(burnBin,2) + inis to pqRTyp
*
65 index on case + inis + str(dischYear,4) + str(burnBin,2) to pqinis
66 index on case + utility + str(dischYear,4) + str(burnBin,2) to pqutil
67 index on case + assmclas + str(dischYear,4) + str(burnBin,2) to pqAClass
68 index on case + str(dischYear,4) + str(burnBin,2) to pqYear

```

APPENDIX A - PROGRAM LISTINGS
FIXNDX.PRG

13

```
69
70 line1 = "Working on Projected Summary database"
71 startPos = (80 - len(line1))/2
72 @ 10, startPos say line1
73
74 use psummary
75 index on case + class+ name + str(dischyear,4) to psummary
76
77 use preactor
78 index on case + recode to preactor
79 index on case + upper(name) to precname
80
81 use putility
82 index on case+ utilcode to putility
83 index on case + upper(name) to putlname
84
85 use pAssmCls
86 index on case + assmclas to pAssmCls
87 index on case + upper(name) to pACName
88
89 use preactyp
90 index on case+reactype to preactyp
91 index on case + name to prtname
92
93 ** 12/8/89 -- Added 2 new files -- summary data for Assembly class. This
94 ** is so we report on these by codes rather than names.
95
96 use aCSum
97 index on class + primName + str(dischyear,4) to aCSum
98 use pACsum
99 index on case + class + primName + str(dischyear,4) to pACSum
100
101
```

APPENDIX A -- PROGRAM LISTINGS

GENRPT.PRG

14

```

1 ****
2 * GENRPT.PRG--Procedures for general reports for the LWR Quantities      *
3 * Database system.  Part of the Waste Characterization task headed by       *
4 * Karl Notz.                                                               *
5 ****
6
7 ****
8 * GenSetup -- Set up databases and variables for the correct general        *
9 * report.                                                                *
10 ****
11
12 Procedure GenSetup
13     parameter gotData
14
15     gotData = .t.
16
17     select 1
18     use &dbfPath.summary index &dbfPath.summary
19
20     lookArea = "3"
21     select &lookArea
22
23     do case
24         case rptNum = 1           && All Assemblies
25             mClass = 'AL'
26             sortBy = ''
27
28         case rptNum = 2           && Utility
29             mClass = "UT"
30             use &dbfPath.utility index &dbfPath.utility
31             sortBy = "UTILITY"
32
33         case rptNum = 3           && Reactor
34             mClass = 'RE'
35             use &dbfPath.reactor index &dbfPath.reactor
36             sortBy = 'REACTOR'
37
38         case rptNum = 4           && Pool
39             mClass = 'PO'
40             use &dbfPath.pool index &dbfPath.pool
41             sortBy = 'POOL'
42
43         case rptNum = 5           && Assembly Type
44             mClass = 'AT'
45             use &dbfPath.assmtype index &dbfPath.assmtype
46             sortBy = 'ASSEMBLY TYPE'
47
48     ** Special case for assembly class, since we need that ordered by code,
49     ** instead of by name, like everything else.
50
51     case rptNum = 6           && Assembly Class
52         select 1
53         use &dbfPath.ACSum alias summary
54         set index to &dbfPath.ACSum
55         select &lookArea
56         mClass = 'AC'
57         use &dbfPath.assmclas index &dbfPath.assmclas
58         sortBy = 'ASSEMBLY CLASS'
59
60     case rptNum = 7           && Reactor Type
61         mClass = "RT"
62         use &dbfPath.reactType index &dbfPath.reactType
63         sortBy = 'REACTOR TYPE'
64
65     endcase
66
67     do GenRpt1 with mClass, lookArea, sortBy
68
69     select 3
70     use
71

```

```

72 *select 1
73 *if (mClass = "AC")
74 *  use &dbfPath.summary index &dbfPath.summary
75 *endif
76 return
77 ****
78 * GenRpt1 -- Produces a general report (options 1-6). Called from GenSetup.*
79 ****
80 ****
81
82 Procedure GenRpt1
83   parameters mClass, lookArea, sortBy
84
85 select 1           && Summary
86
87 if (debug)
88   @ 20,0 clear
89   @ 20,10 say "In GenRpt1. mClass: " + mClass
90   wait "" to dummy
91   do browse
92 endif
93
94 seek mClass
95 if (.not. found())
96   @ 21,0 clear
97   @ 21,10 say "Can't find any data for " + sortBy
98   @ 22,10 say "Strike any key to continue"
99   wait "" to dummy
100 return
101 endif
102
103 do GenHdg with sortBy
104 set alternate to qty.rpt
105 set alternate on
106
107 ** Zero grand total accumulators
108 gt_NumAssm = 0
109 gt_DefAssm = 0
110 gt_Burn = 0
111 gt_Wt = 0
112 gt_Enri = 0
113
114 select 1           && Summary
115 do while (mClass = summary->class .and. .not. eof())
116
117 ** Zero sub-total accumulators
118   st_NumAssm = 0
119   st_DefAssm = 0
120   st_Burn = 0
121   st_Wt = 0
122   st_Enri = 0
123
124 curName = primName
125
126 if (rptNum = 1)      && All assemblies, nothing to look up
127   curGroup = "All assemblies"
128 else
129   curGroup = summary->name
130   if (len(curGroup) > 24)
131     curGroup = substr(curdgroup,1,24)
132   endif
133 endif
134
135 if (debug)
136   @ 21,0 clear
137   @ 21,10 say "found: "
138   @ 21,30 say found() picture 'y'
139   @ 22,10 say 'curGroup: ' + curGroup
140   wait "" to dummy
141   @ 10,0 say ''
142 endif

```

APPENDIX A -- PROGRAM LISTINGS
GENRPT.PRG

16

```

143
144     first = .t.
145     select 1           && summary
146
147     do while (curName = primName .and. mClass = summary->class .and.-
148             .not. eof())
149
150     ** If report is all assemblies or assemblies by Reactor Type, we have
151     ** burnup bin data
152
153     if (rptNum = 1 .or. rptNum = 7)
154         do Gen1YrB
155     else
156         st_NumAssm = st_NumAssm + numAssm
157         st_DefAssm = st_DefAssm + defAssm
158         st_Burn = st_Burn + (numAssm * avgBurn)
159         st_Wt = st_Wt + weight
160         st_Enri = st_Enri + (numAssm * avgEnri)
161
162         if (by_Year)
163             if (first)      && Need to print the primary name?
164                 line1 = curGroup
165                 l = len(line1)
166                 line1 = line1 + space(25-l)
167                 first = .f.
168             else
169                 line1 = space(25)
170             endif
171
172             line1 = line1 + str(dischYear,4) + space(6) + str(numAssm,5) +;
173             space(5)
174
175             if (defAssm > 0)
176                 line1 = line1 + str(defAssm,5)
177             else
178                 line1 = line1 + space(5)
179             endif
180
181             line1 = line1 + space(5) + str(avgBurn,5) +;
182             space(3) + str(weight,8,1) + space(4) + str(avgEnri,5,3)
183             ? line1
184             endif
185             skip
186         endif          && If rptNum = 1 or rptNum = 7
187     enddo
188
189     ** Print out totals line or line for the group
190     if (by_Year)
191         line1 = '--- SUB TOTALS'
192         ? '
193     else
194         line1 = curGroup
195         l = len(line1)
196         line1 = line1 + space(25 - l)
197     endif
198
199     st_AvgBurn = round(st_Burn/st_NumAssm,0)
200     st_AvgEnri = round(st_Enri/st_NumAssm,3)
201
202     if (st_DefAssm > 0)
203         strDef = str(st_DefAssm,5)
204     else
205         strDef = space(5)
206     endif
207
208     if (by_Bin)
209         line1 = '--- SUB TOTALS'    ' + space(11) + str(st_numAssm,5) + space(5
210     ) * strDef +;
211         space(9) + str(st_AvgBurn,6) + space(4) + str(st_Wt,8,1) + space(4);
212         + str(st_AvgEnri,5,3)
213     else

```

APPENDIX A – PROGRAM LISTINGS
GENRPT.PRG

17

```

213      if (by_Year)
214          line1 = line1 + space(10) + str(st_NumAssm,5) + space(5) ;
215          + strDef + space(5) + str(st_AvgBurn,5) + space(3) +;
216          str(st_Wt,8,1) + space(4) + str(st_AvgEnri,5,3)
217      else
218          && Totals only
219          line1 = line1 + space(3) + str(st_NumAssm,5) + space(4)      ;
220          + strDef + space(7) + str(st_AvgBurn,6) + space(5) +;
221          str(st_Wt,8,1) + space(6) + str(st_AvgEnri,5,3)
222      endif
223  endif
224 ? line1
225 ? ''
226 ** Add subtotals to grand totals
227 gt_NumAssm = gt_NumAssm + st_NumAssm
228 gt_DefAssm = gt_DefAssm + st_DefAssm
229 gt_Burn = gt_Burn + st_Burn
230 gt_Wt = gt_Wt + st_Wt
231 gt_Enri = gt_Enri + st_Enri
232 enddo
233
234 gt_AvgBurn = round(gt_Burn/gt_NumAssm,0)
235 gt_AvgEnri = round(gt_Enri/gt_NumAssm,3)
236
237 if (gt_DefAssm > 0)
238     strDef = str(gt_DefAssm,5)
239 else
240     strDef = space(5)
241 endif
242
243 if (by_Bin)
244     line1 = '--- GRAND TOTALS  ' + space(11) + str(gt_numAssm,5) + space(5) +
245     strDef +;
246     space(9) + str(gt_AvgBurn,6) + space(4) + str(gt_Wt,8,1) + space(4);
247     + str(gt_AvgEnri,5,3)
248 else
249     if (by_Year)
250         line1 = '--- GRAND TOTALS           ' + space(9) + str(gt_NumAssm,6) +;
251         space(5) + strDef + space(5) + str(gt_AvgBurn,5) + space(3) +;
252         str(gt_Wt,8,1) + space(4) + str(gt_AvgEnri,5,3)
253     else
254         line1 = '--- GRAND TOTALS           ' + space(2) + str(gt_NumAssm,6) +;
255         space(4) + strDef + space(8) + str(gt_AvgBurn,5) + space(5) +;
256         str(gt_Wt,8,1) + space(6) + str(gt_AvgEnri,5,3)
257     endif
258 endif
259 ? line1
260 return
261 ****
262 * GenHdg -- Headings for a general report. Called from GenRpt1. *
263 ****
264 ****
265
266 Procedure GenHdg
267     parameter sortBy
268
269 set alternate to qty.hdg
270 set alternate on
271
272 ?? title1
273 strNum = str(rptNum,1)
274 line1 = "Historical Data"
275 startPos = (80 - len(line1))/2
276 line1 = space(startPos) + line1
277 ? line1
278
279 line1 = hRTtitle&strNum
280 startPos = (80 - len(line1))/2
281 line1 = space(startPos) + line1
282 ? line1

```

APPENDIX A -- PROGRAM LISTINGS
GENRPT.PRG

18

```

283
284 ** Column Headings
285 if (by_Bin)
286   ? space(54) + " AVG" + space(7) + 'TOTAL' + space(5) + 'AVG'
287 else
288   if (by_Year)
289     ? space(54) + " AVG" + space(7) + 'TOTAL' + space(5) + 'AVG'
290   else
291     ? space(49) + " AVG" + space(9) + 'TOTAL' + space(7) + 'AVG'
292   endif
293 endif
294
295 line1 = space(23)
296 if (by_Bin)
297   line1 = "    DISCHARGE      BURNUP      NUMBER      DEFEC.      BURNUP      W
EIGHT      INIT"
298 else
299   if (by_Year)
300     line1 = line1 + 'DISCHARGE' + space(3) + "NUMBER" + space(4) + 'DEFEC.'
;
301     + space(3) + 'BURNUP' + space(4) + 'WEIGHT' + space(4) + 'INIT '
302   else
303     line1 = line1 + space(4) + "NUMBER" + space(4) + 'DEFEC.';
304     + space(6) + 'BURNUP' + space(7) + 'WEIGHT' + space(5) + 'INIT '
305   endif
306 endif
307
308 ? line1
309
310 line1 = sortBy
311 l = len(line1)
312 line1 = line1 + space(25-l)
313 if (by_bin)
314   line1 = "    YEAR      BIN      ASSM      ASSM      (Mwd/MTIHM)  (
MTIHM)      ENRICH"
315 else
316   if (by_Year)
317     line1 = line1 + 'YEAR' + space(7) + 'ASSM' + space(6) + 'ASSM';
318     + space(2) + '(Mwd/MTIHM)' + space(1) + '(MTIHM)' + space(3) + ;
319     'ENRICH'
320   else
321     line1 = line1 + space(3) + 'ASSM' + space(6) + 'ASSM';
322     + space(4) + '(Mwd/MTIHM)' + space(4) + '(MTIHM)' + space(5) + ;
323     'ENRICH'
324   endif
325 endif
326 ? line1 + carbRet + lineFeed
327
328 return
329
330 -----
331 * Gen1YrB -- Total up data for 1 year, all burnup bins. Called from
332 * procedure GenRpt1 when working on report for All Assemblies or All
333 * Assemblies by Reactor Type. These 2 reports are available with burnup
334 * bin breakdowns, so the data is handled a little differently from the
335 * other general reports, where burnup bin breakdown is not available.
336 -----
337
338 Procedure Gen1YrB
339
340 private curYear
341
342 curYear = diagYear
343 ** Zero accumulators for this year
344 st2_NumAssm = 0
345 st2_DefAssm = 0
346 st2_Burn = 0
347 st2_Wt = 0
348 st2_Enri = 0
349
350 do while (curName = primName .and. mClass = summary->class .and.;
```

APPENDIX A -- PROGRAM LISTINGS
GENRPT.PRG

19

```

351     dischYear = curYear .and. .not. eof())
352
353     st2_NumAssm = st2_NumAssm + numAssm
354     st2_DefAssm = st2_DefAssm + defAssm
355     st2_Burn = st2_Burn + (numAssm * avgBurn)
356     st2_Wt = st2_Wt + weight
357     st2_Enri = st2_Enri + (numAssm * avgEnri)
358
359     if (by_Bin)
360         if (first)          && Need to print the primary name?
361             line1 = curGroup      && Print on separate line to save space
362             first = .f.
363             ? line1
364             ? ''
365         endif
366
367         line1 = '      ' + str(curYear,4) + space(5)
368         if (burnBin < 10)
369             strNum = str(burnBin,1)
370         else
371             strNum = str(burnBin,2)
372         endif
373
374         line1 = line1 + bb&strNum + space(4) + str(numAssm,5) + space(5)
375         if (defAssm > 0)
376             line1 = line1 + str(defAssm,5)
377         else
378             line1 = line1 + space(5)
379         endif
380
381         line1 = line1 + space(9) + str(avgBurn,6) + space(4) + str(weight,8,1)
382         + space(4);
383         + str(avgEnri,5,3)
384     .     ? line1
385     endif
386
387     skip
388 enddo
389
390 if (by_Bin)
391     ?
392 else
393     if (by_Year)        && Printing only 1 line for the year
394         if (first)          && Need to print the primary name?
395             line1 = curGroup
396             l = len(line1)
397             line1 = line1 + space(25-l)
398             first = .f.
399         else
400             line1 = space(25)
401         endif
402
403         line1 = line1 + str(curYear,4) + space(5) + str(st2_numAssm,5) +
404             space(5)
405         if (st2_defAssm > 0)
406             line1 = line1 + str(st2_defAssm,5)
407         else
408             line1 = line1 + space(5)
409         endif
410
411         st2_AvgBurn = round(st2_Burn/st2_NumAssm,0)
412         st2_AvgEnri = round(st2_Enri/st2_NumAssm,3)
413
414         line1 = line1 + space(5) + str(st2_AvgBurn,5) +
415             space(3) + str(st2_Wt,8,1) + space(4) + str(st2_AvgEnri,5,3)
416     ? line1
417 endif
418
419 ** Add this year's accumulators to subtotals for this group
420 st_NumAssm = st_NumAssm + st2_numAssm

```

APPENDIX A -- PROGRAM LISTINGS
GENRPT.PRG

20

```
421 st_DefAssm = st_DefAssm + st2_DefAssm
422 st_Burn = st_Burn + st2_Burn
423 st_Wt = st_Wt + st2_Wt
424 st_Enri = st_Enri + st2_Enri
425 return
```

APPENDIX A - PROGRAM LISTINGS HELP.PRG

21

```
1 ****
2 * Help.Prg -- Help program. When user strikes F1, control will be *
3 * transferred to this routine. Based on the routine that the user was *
4 * currently in, an appropriate help screen will be displayed. This can *
5 * only be used with Clipper. *
6 ****
7
8 parameters callPrg, lineNum, inputVar
9
10 if (debug)
11   @ 21,0 clear
12   @ 21,10 say "callPrg: " + callPrg + "    lineNum: " + str(lineNum,4)
13   @ 22,10 say "InputVar: " + inputVar
14   wait "" to dummy
15 endif
16
17 if (callPrg = "Help")
18   return
19 endif
20
21 call ClipPop with "Fill Page 1"
22 do case
23   case callPrg = "HPSLCT"
24     call ClipPop with "HPHelp"
25     @ 21,0 say ""          && Position cursor
26     wait "" to dummy
27
28   case callPrg = "HRPTSLCT" .or. callPrg = "PRPTSLCT"
29     call ClipPop with "RptHelp"
30     @ 21,0 say ""          && Position cursor
31     wait "" to dummy
32
33   case callPrg = "BRKDSLCT"
34     call ClipPop with "BrkDHelp"
35     @ 21,0 say ""          && Position cursor
36     wait "" to dummy
37
38   case callPrg = "DEVSLCT"
39     call ClipPop with "DevHelp"
40     @ 21,0 say ""          && Position cursor
41     wait "" to dummy
42
43   case callPrg = "START"
44     call ClipPop with "StrtHelp"
45     @ 21,0 say ""          && Position cursor
46     wait "" to dummy
47 endcase
48
49 call ClipPop with "Display Page 1"
50 return
51
52
```

APPENDIX A -- PROGRAM LISTINGS INSTALL.BAT

22

```
1 rem Install.Bat -- program to install quantities database.
2 echo off
3 if . == $1. goto Err1
4 cls
5 echo                               OCRWM/ORNL
6 echo          LWR Spent Fuel Quantities Database System

7 echo .
8 echo This installation procedure will transfer the Quantities Database
9 echo System from floppy disk to a hard disk.
10 echo .
11 echo The hard disk you have selected is drive- $1
12 echo .
13 echo The LWR Quantities System requires at least 2.5 Megabytes of
14 echo available disk space. This procedure will check that there is at least
15 echo that much room available on the fixed drive $1.
16 echo If there is not that much room available on the drive, you can
17 echo exit this installation procedure now and install on another drive
18 echo or make some more room available on this drive.
19 echo .
20 disksp 2500000 $1
21 if errorlevel 2 goto NoRoom
22 if errorlevel 1 goto Problem
23 echo There is sufficient room on the hard disk.
24 echo Do you wish to continue the installation?
25 query
26 if not errorlevel 1 goto Done
27 echo .

28
29 rem Check if they are installing into root or not
30 ChkRoot $1
31 if errorlevel 1 goto Done
32
33 chkArg $1
34 if errorlevel 1 goto Colon
35
36 copy QTY.bat $1:
37 md $1:QTY
38 cd $1:QTY
39 copy *.* $1:
40 echo Remove the Quantities Database Diskette #1, and insert
41 echo the Quantites Database Diskette #2 in the floppy drive.
42 pause
43 copy *.* $1:
44 echo Remove the Quantities Database Diskette #2, and insert
45 echo the Quantities Database Data #3 Diskette in the floppy drive.
46 pause
47 copy *.* $1:
48 echo Remove the Quantities Database Diskette #3, and insert
49 echo the Quantities Database Data #4 Diskette in the floppy drive.
50 pause
51 copy *.* $1:
52
53 cls
54 $1:
55 echo The data files have now been copied. Several index files must be
56 echo created. This procedure will take several minutes.
57 pause
58 goto Last
59
60 rem Come here if argument had a colon -- same as above but leave off ':'
61 :Colon
62 copy QTY.bat $1
63 md $1:QTY
64 cd $1:QTY
65 copy *.* $1
66 echo Remove the Quantities Database Diskette #1, and insert
67 echo the Quantites Database Diskette #2 in the floppy drive.
68 pause
69 copy *.* $1
70 echo Remove the Quantities Database Diskette #2, and insert
```

APPENDIX A - PROGRAM LISTINGS INSTALL.BAT

23

```
71 echo the Quantities Database Data #3 Diskette in the floppy drive.
72 pause
73 copy *.* %1
74 echo Remove the Quantities Database Diskette #3, and insert
75 echo the Quantities Database Data #4 Diskette in the floppy drive.
76 pause
77 copy *.* %1
78
79 cls
80 %1
81 echo The data files have now been copied. Several index files must be
82 echo created. This procedure will take several minutes.
83 pause
84
85 :Last
86 fixndx
87 cd ..
88 cls
89 echo .
90 echo The installation is now complete. You may remove the floppy disk from
91 echo the drive. A batch file named QTY has been placed in the current
92 echo directory of your fixed disk. You may run the Quantities
93 echo Database system by typing QTY when you are in that directory.
94 goto Done
95
96 :Err1
97 echo You must indicate the letter of your hard disk. For example, if your
98 echo hard disk letter is C, you would enter the command
99 echo INSTALL C. Please start over.
100 goto Done
101
102 :NoRoom
103 echo There is not enough room on this fixed disk to install the Quantities
104 echo System. Please install the system on another drive, or make some
105 echo more room on this drive.
106 goto Done
107
108 :Problem
109 echo There has been a problem in checking out the available disk space.
110 echo Please try again, or use another drive if one is available.
111
112 :Done
113 echo .
```

APPENDIX A – PROGRAM LISTINGS

PDETRPT.PRG

24

```

1 ****
2 * PREPORTS.PRG -- Procedures for the reports on projected data for the *
3 * Quantities Data Base.
4 * These reports can be either general reports (all assemblies or by utility,*
5 * reactor, or reactor type) or specific reports (one particular entity).
6 * The procedures for the Historical data reports are in reports.prg.
7 ****
8
9 ****
10 * PDSetup. Setup parameters for a detailed report. Then call PDetRpt1. *
11 ****
12
13 Procedure PDSetup
14     parameter pCase, getData
15
16 if (debug)
17     set color to gr+/n
18     @ 21,0 clear
19     @ 21,10 say "In DetSetup. rptNum: "
20     @ 21,40 say str(rptNum,2)
21     wait "" to dummy
22 endif
23
24 select 1
25 lookArea = "3"           && Set up lookup table here
26
27 do case
28     case rptNum = 8          && Utility
29         key2 = utilName
30         keyName = "substr(inis,1,2)"
31         use &dbfPath.pqty index &dbfPath.pqUtil
32         select 3               && Lookup area
33         use &dbfPath.putility index &dbfPath.putility
34
35     case rptNum = 9          && Reactor
36         key2 = reacName
37         keyName = "inis"
38         use &dbfPath.pqty index &dbfPath.pqInis
39         select 3               && Lookup area
40         use &dbfPath.preactor index &dbfPath.preactor
41
42     case rptNum = 12          && Assembly Class
43         key2 = assmCName
44         keyName = "assmClas"
45         use &dbfPath.pqty index &dbfPath.pqAClass
46         select 3               && Lookup area
47         use &dbfPath.pAssmCls index &dbfPath.pAssmCls
48
49     case rptNum = 13          && Reactor Type
50         key2 = substr(rTName,1,1)
51         keyName = "reacType"
52         use &dbfPath.pqty index &dbfPath.pqRTType
53         select 3               && Lookup area
54         use &dbfPath.preactTyp index &dbfPath.preactTyp
55
56 endcase
57
58 select 1
59 ** build temporary tables
60
61 gotData = .f.
62 key = pCase + key2
63
64 if (debug)
65     set color to gr+/n
66     @ 21,0 clear
67     @ 21,10 say "About to call PBldTemp. key: " + key
68     wait "" to dummy
69 endif
70
71 do PBldTemp with key, keyName, getData

```

APPENDIX A – PROGRAM LISTINGS
PDETRPT.PRG

25

```

72 if (debug)
73   set color to gr+/n
74   @ 21,0 clear
75   @ 21,10 say "Back from PBldTemp. key: " + key
76   wait "" to dummy
77 endif
78 if (.not. getData)
79   select 3
80   use
81   return
82 endif
83
84 ** Create reports
85 do PDetRpt1 with key, key2,KeyName, lookArea
86
87 select 3           && close lookup dbf
88 use
89 return
90
91 ****
92 * PBldTemp -- Build a temporary table with the records of interest for this *
93 * report.
94 ****
95
96 Procedure PBldTemp
97   parameters key, keyName, found
98
99 select 4
100 use temp
101 delete all
102 pack
103
104 select 1           && pqty
105 seek key           && Any data for this key?
106 if (.not. found())
107   * @ 21,0 clear
108   * @ 21,10 say "Can't find key: " + key
109   * wait "" to dummy
110   return
111 else
112   found = .t.
113 endif
114
115 if (debug)
116   set color to gr+/n
117   @ 21,0 clear
118   @ 21,10 say "In PBldTemp. key: " + key
119   @ 22,10 say "Found the key. case: " + case
120   wait "" to dummy
121 endif
122
123 do while (key = (case + &keyname) .and. .not. eof())
124   select 4           && Temp
125   append blank
126   replace inis with pqty->inis
127   replace dischYear with pqty->dischYear
128   replace numAssm with pqty->numAssm
129   replace weight with pqty->weight
130   replace reactType with pqty->reactType
131   replace avgBurn with pqty->avgBurn
132   replace burnBin with pqty->burnBin
133   replace enrich with pqty->enrich
134   replace assmclas with pqty->assmclas
135
136 if (debug)
137   set color to gr+/n
138   @ 21,0 clear
139   @ 21,10 say "Added a record. inis: " + temp->inis
140   wait "" to dummy
141 endif
142

```

APPENDIX A - PROGRAM LISTINGS
PDETRPT.PRG

26

```

143      select 1                      && qty
144      skip
145      if (key <> (case + &keyName))
146          exit
147      endif
148  enddo
149
150 ** Set up correct index
151 ndxKeys = keyName
152
153 if (by_Year)
154     ndxKeys = ndxKeys + " + str(dischYear,4)"
155 endif
156
157 if (by_Bin)
158     ndxKeys = ndxKeys + " + str(dischYear,4) + str(burnBin,2)"
159 endif
160
161 select 4
162 index on &ndxKeys to temp
163 go top
164 if (debug)
165     set color to gr+/n
166     @ 21,0 clear
167     @ 21,10 say "Leaving PBldTemp.  ndxKeys:  " + ndxKeys
168     wait "" to dummy
169 endif
170
171 return
172
173 ****
174 * PDetRpt1.  Produce a detailed reports.  Called from the DetSetup routine. *
175 ****
176
177 Procedure PDetRpt1
178     parameters key, key2, keyName, lockArea
179
180 select 4                      && Temp
181 if (debug)
182     set color to gr+/n
183     @ 21,0 clear
184     @ 21,10 say "In DetRpt.  key:  " + key
185     wait "" to dummy
186 endif
187
188
189 select 4                      && Temp
190 go top
191
192 do PDetHdg with key, keyName      && Produce heading
193 set alternate to qty.rpt
194 set alternate on
195
196 ** Zero grand total accumulators
197 gt_NumAssm = 0
198 gt_Wt = 0
199 gt_Burn = 0
200 gt_Enri = 0
201
202 do POneGRpt with key2, keyName
203
204 ** Calculate averages
205 gt_AvgBurn = round(gt_Burn/gt_NumAssm,0)
206 gt_AvgEnri = round(gt_Enri/gt_NumAssm,3)
207
208 ** Print out totals line
209 ? ''
210 line1 = " --- TOTALS      "
211 if (by_Bin)
212     line1 = line1 + space(10) + str(gt_NumAssm,6) + space(11) +
213             str(gt_AvgBurn,6) + space(5) + str(gt_Wt,8,1) + space(7);

```

APPENDIX A - PROGRAM LISTINGS
PDETRPT.PRG

27

```

214     + str(gt_AvgEnri,5,3)
215 else
216     line1 = line1 + str(gt_NumAssm,6) + space(11) + str(gt_AvgBurn,5) +;
217     space(8) + str(gt_Wt,8,1) + space(9) + str(gt_AvgEnri,5,3)
218 endif
219
220 ? line1
221
222 return
223 ****
224 * POneGRpt -- Produce report for one group, or all data if no subtotals were*
225 * indicated. *
226 ****
227 ****
228
229 Procedure POneGRpt
230   parameters key2, keyName
231
232 do while (.not. eof())
233
234 if (debug)
235   @ 21,0 clear
236   @ 21,10 say "In OneGRpt.  key: " + key
237   @ 22,10 say "keyName: " + keyName
238   wait "" to dummy
239 endif
240
241 ** Zero sub total accumulators
242   st_NumAssm = 0
243   st_Wt = 0
244   st_Burn = 0
245   st_Enri = 0
246
247 ** Handle data for one discharge year
248   do POneYear with key2, keyName
249
250 ** Add year's totals to grand totals
251   gt_NumAssm = gt_NumAssm + st_NumAssm
252   gt_Wt = gt_Wt + st_Wt
253   gt_Burn = gt_Burn + st_Burn
254   gt_Enri = gt_Enri + st_Enri
255   select 4
256
257 if (debug)
258   set color to gr+/n
259   @ 23,0 clear
260   @ 23,10 say "End of loop in DetRpt.  Key: " + key
261   @ 24,10 say "keyName: "
262   @ 24,40 say &keyName
263   wait "" to dummy
264 endif
265 enddo
266 return
267 ****
268 * PDetHdg.  Produce the heading for detailed reports.  Called from the *
269 * PDetRpt procedure. *
270 ****
271 ****
272
273 Procedure PDetHdg
274   parameter key, keyName
275
276 set alternate to qty.hdg
277 set alternate on
278
279 ?? title1
280 if (rptNum < 10)
281   strNum = str(rptNum,1)
282 else
283   strNum = str(rptNum,2)
284 endif

```

**APPENDIX A ~ PROGRAM LISTINGS
PDETRPT.PRG**

28

```

285
286 line1 = "Projected Data: "
287 if (histproj = 2)           && No New Orders Case
288   line1 = line1 + "No New Orders Case"
289 else
290   line1 = line1 + "Upper Reference Case"
291 endif
292 line1 = line1 + "with Extended Burnup"
293
294 startPos = (80 - len(line1))/2
295 line1 = space(startPos) + line1
296 ? line1
297
298 line2 = ""
299 line1 = "Data Broken Down By: "
300
301 select &clockArea
302 seek key
303
304 if (by_Year)
305   line1 = line1 + "Discharge Year"
306 else
307   if (by_Bin)
308     line1 = line1 + "Discharge Year and Burnup Bin"
309   else
310     line1 = line1 + "No Subtotals"
311   endif
312 endif
313
314 startPos = (80 - len(line1))/2
315 line1 = space(startPos) + line1
316 ? line1
317
318 line1 = pRTtitle&strNum + trim(Name)          && Put in specific name
319 startPos = (80 - len(line1))/2
320 line1 = space(startPos) + line1
321 ? line1
322
323 do PDetHdg2                                && Get column headings
324
325 select 4                                     && Temp
326 return
327
328 ****
329 * PDetHdg2. Produce the column headings for detailed reports. Called from *
330 * DetHdg procedure.
331 ****
332
333 Procedure PDetHdg2
334
335 line0 = space(37) +      * AVG        TOTAL        AVG*
336
337 line1 = "NUMBER          BURNUP        WEIGHT        INIT"
338 line2 = "ASSEMBLIES      (Mwd/MTITEM)    (MTITEM)    ENRICH"
339 if (by_Bin)
340   line0 = space(49) + "AVG        TOTAL        AVG"
341   line1 = 'DISCHARGE' + space(7) + 'BURNUP' + space(9) + "NUMBER" +
342         space(11) + "BURNUP" + space(6) + 'WEIGHT' + space(6) + "INIT"
343   line2 = 'YEAR' + space(7) + 'BIN' + space(9) + "ASSMB" +
344         space(9) + '(Mwd/MTITEM)' + space(4) + "(MTITEM)" + space(5) +
345         + "ENRICH"
346 else
347   if (by_Year)
348     line1 = 'DISCHARGE' + space(12) + line1
349     line2 = 'YEAR' + space(10) + line2
350   else
351     line1 = space(9) + space(12) + line1
352     line2 = space(9) + space(10) + line2
353   endif
354 endif
355
```

APPENDIX A -- PROGRAM LISTINGS
PDETRPT.PRG

29

```

356 ? line0
357 ? line1
358 ? line2 + carbRate + lineFeed
359
360 return
361 ****
362 * POneYear. Handle Historical Quantity data for one year. Add the year's *
363 * values to the subtotals. If the user wants to see the data by burnup bin,*
364 * print out detail lines.
365 * print out detail lines.
366 ****
367
368 Procedure POneYear
369     parameters key2, keyName
370
371 curYear = dischYear
372
373 if (debug)
374     @ 20, 0 clear
375     @ 20,10 say "In POneYear"
376     @ 21,10 say "curYear: " + str(curYear,4)
377     @ 22,10 say "key2: " + key2 + " keyName: " + keyName
378     @ 23,10 say "key2 = &keyName: "
379     @ 23,35 say key2 = &keyName picture 'Y'
380     wait "" to dummy
381 endif
382 do while (key2 = &keyName .and. dischYear = curYear .and. .not. eof())
383     curBin = burnBin
384     bb_NumAssm = 0
385     bb_DefAssm = 0
386     bb_Burn = 0
387     bb_Wt = 0
388     bb_Enri = 0
389
390     do while (key2 = &keyName .and. dischYear = curYear .and. curBin = burnBin
391         .and. ;
392             .not. eof())
393             if (debug)
394                 @ 20,0 clear
395                 @ 20,10 say "At top of burnup bin loop in POneYear. CurYear: " + st
r(curYear,4)
396                 @ 21,10 say "numAssm: "
397                 @ 21,25 say numAssm
398                 @ 21,40 say "weight: "
399                 @ 21,55 say weight
400                 @ 22,10 say "enrich: "
401                 @ 22,40 say enrich
402                 wait "" to dummy
403             endif
404             bb_NumAssm = bb_NumAssm + numAssm
405             bb_Burn = bb_Burn + (numAssm * avgBurn)
406             bb_Wt = bb_Wt + weight
407             bb_Enri = bb_Enri + (numAssm * enrich)
408
409             skip
410             if .not. (key2 = &keyName .and. curBin = burnBin .and. ;
411                 dischYear = curYear)
412                 exit
413             endif
414         enddo             && Finished this burnup bin
415
416         if (by_Bin)
417             bb_AvgBurn = round(bb_Burn/bb_NumAssm,0)
418             bb_AvgEnri = round(bb_Enri/bb_NumAssm,3)
419
420             line1 = ' ' + str(curYear,4) + space(6)
421             if (curBin < 10)
422                 strNum = str(curBin,1)
423             else
424                 strNum = str(curBin,2)

```

```
425      endif
426
427      line1 = line1 + bb$&strNum + space(8) + str(bb_NumAssm,5) + space(11) +
428          str(bb_avgBurn,6) + space(5) + str(bb_Wt,8,1) + space(7);
429          + str(bb_avgEnri,5,3)
430 ? line1
431 if (debug)
432     @ 21,0 clear
433     @ 21,10 say "Output a line for burnup bin."
434     @ 22,10 say line1
435     wait "" to dummy
436 endif
437 endif
438
439 ** Add burnup bin totals to subtotals
440 st_NumAssm = st_NumAssm + bb_NumAssm
441 st_Burn = st_Burn + bb_Burn
442 st_Wt = st_Wt + bb_Wt
443 st_Enri = st_Enri + bb_Enri
444
445 if .not. (key2 = &keyName .and. dischYear = curYear)
446     exit
447 endif
448 enddo
449
450 if (by_Year)           && Print lines subtotalled by year
451     st_AvgBurn = round(st_Burn/st_NumAssm,0)
452     st_AvgEnri = round(st_Enri/st_NumAssm,3)
453     line1 = ' ' + str(curYear,4) + space(16) + str(st_NumAssm,4) +
454         space(11) + str(st_AvgBurn,5) + space(8) + str(st_Wt,8,1) +
455         space(9) + str(st_AvgEnri,5,3)
456 ? line1
457 endif
458
459 if (by_Bin)           && Print a blank line
460 ? ''
461 endif
462
463 return
```

APPENDIX A - PROGRAM LISTINGS
PGENRPT.PRG

31

```

1 **** **** **** **** **** **** **** **** **** **** **** ****
2 * PGENRPT.PRG -- Procedures for projected data, General Reports. For the *
3 * LWR Quantities Database syste, which is part of the Waste Characteriza-
4 * task headed by Karl Notz of Chem Tech Division. *
5 **** **** **** **** **** **** **** **** **** **** ****
6
7 **** **** **** **** **** **** **** **** **** **** ****
8 * PGSetup -- Set up databases and variables for the correct general *
9 * report. *
10 **** **** **** **** **** **** **** **** **** **** ****
11
12 Procedure PGSetup
13     parameter pCase, gotData
14
15     gotData = .t.
16     select 1
17     use &dbfPath.psummary index &dbfPath.psummary
18
19     lookArea = "3"
20     select &lookArea
21
22     do case
23         case rptNum = 1           && All Assemblies
24             mClass = 'AL'
25             sortBy = ''
26
27         case rptNum = 2           && Utility
28             mClass = "UT"
29             use &dbfPath.utility index &dbfPath.utility
30             sortBy = "UTILITY"
31
32         case rptNum = 3           && Reactor
33             mClass = 'RE'
34             use &dbfPath.preactor index &dbfPath.preactor
35             sortBy = 'REACTOR'
36
37         case rptNum = 6           && Assembly class
38             mClass = 'AC'
39             select 1
40             use &dbfPath.pACSum alias pSummary
41             set index to &dbfPath.pACSum
42             select &lookArea
43
44             use &dbfPath.pAssmCls index &dbfPath.pAssmCls
45             sortBy = "ASSEMBLY CLASS"
46
47         case rptNum = 7           && Reactor Type
48             mClass = "RT"
49             use &dbfPath.preactTyp index &dbfPath.preactTyp
50             sortBy = 'REACTOR TYPE'
51
52     endcase
53
54     key = pCase + mClass
55
56     if (debug)
57         @ 21,0 clear
58         @ 21,10 say "About to call PGenRpt1. key: " + key
59         wait "" to dummy
60     endif
61
62     do PGenRpt1 with key, pCase, lookArea, sortBy
63
64     select 3           && close lookup dbf
65     use
66     return
67
68 **** **** **** **** **** **** **** **** **** **** ****
69 * PGenRpt1 -- Produce a general report (options 1-5). Called from PGSetup.*
70 **** **** **** **** **** **** **** **** **** ****
71

```

```
72 Procedure PGenRpt1
73   parameters key, pCase, lookArea, sortBy
74
75   select 1           && PSummary
76
77   if (debug)
78     @ 21,0 clear
79     @ 21,10 say "In PGenRpt1.  key: " + key
80     wait "" to dummy
81   endif
82
83   seek key
84   if (.not. found())
85     @ 21,0 clear
86     @ 21,10 say "Can't find any data for " + sortBy
87     @ 22,10 say "Strike any key to continue"
88     wait "" to dummy
89   return
90   endif
91
92   do PGenHdg with sortBy
93   set alternate to qty.rpt
94   set alternate on
95
96   ** Zero grand total accumulators
97   gt_NumAssm = 0
98   gt_Wt = 0
99   gt_Burn = 0
100  gt_Enri = 0
101
102  select 1           && PSummary
103  do while (key = (psummary->case + psummary->class) .and. .not. eof())
104
105  ** Zero sub-total accumulators
106  ' st_NumAssm = 0
107  st_Wt = 0
108  st_Burn = 0
109  st_Enri = 0
110
111  curName = primName
112
113  ** Look up primary name
114  if (rptNum = 1)      && All assemblies, nothing to look up
115    curGroup = "All assemblies"
116  else
117    curGroup = name
118
119    if (debug)
120      @ 20,0 clear
121      @ 20,10 say "curGroup: " + curGroup
122      @ 21,10 say "curName: " + curName
123      wait "" to dummy
124    endif
125  endif
126
127  if (debug)
128    @ 21,0 clear
129    @ 21,10 say "found: "
130    @ 21,30 say found() picture 'y'
131    @ 22,10 say 'curGroup: ' + curGroup
132    wait "" to dummy
133    @ 10,0 say ''
134  endif
135
136  first = .t.
137  select 1           && psummary
138
139  do while (curName = primName .and. key = (psummary->case+psummary->class);
140          .and. .not. eof())
141  ** If report is all assemblies or assemblies by Reactor Type, we have
142  ** burnup bin data
```

APPENDIX A – PROGRAM LISTINGS
PGENRPT.PRG

33

```

143
144     if (rptNum = 1 .or. rptNum = 7)
145         do PGen1YrB
146     else
147         st_NumAssm = st_NumAssm + numAssm
148         st_Wt = st_Wt + weight
149         st_Burn = st_Burn + (numAssm * avgBurn)
150         st_Enri = st_Enri + (numAssm * avgEnri)
151
152    if (debug)
153        @ 21,0 clear
154        @ 21,10 say "In PGenRpt1.  st_Wt: "
155        @ 21,50 say st_Wt
156        wait "" to dummy
157    endif
158
159        if (by_Year)
160            if (first)          && Need to print the primary name?
161                line1 = curGroup
162                l = len(line1)
163                line1 = line1 + space(30-l)
164                first = .f.
165            else
166                line1 = space(30)
167            endif
168
169            line1 = line1 + str(dischYear,4) + space(6) + str(numAssm,5) + ;
170                space(8) + str(avgBurn,5) + space(4) + str(weight,8,1) + ;
171                space(4) + str(avgEnri,5,3)
172            ? line1
173        endif
174        skip
175    endif                      && If rptNum 1 or 6
176    enddo
177
178 ** Print out totals line or line for the group
179    if (by_Year)
180        ?
181        line1 = '--- SUB TOTALS '
182    else
183        line1 = curGroup
184        l = len(line1)
185        line1 = line1 + space(30 - 1)
186    endif
187
188    st_AvgBurn = round(st_Burn/st_NumAssm,0)
189    st_AvgEnri = round(st_Enri/st_NumAssm,3)
190
191    if (by_Bin)
192        line1 = '--- SUB TOTALS ' + space(29) + str(st_NumAssm,6) + space(5)
193        +
194            str(st_AvgBurn,5) + space(3) + str(st_Wt,8,1) + space(3) + ;
195            str(st_AvgEnri,5,3)
196    else
197        if (by_Year)
198            line1 = line1 + space(9) + str(st_NumAssm,6) + space(8) + ;
199            str(st_AvgBurn,5) + space(4) + str(st_Wt,8,1) + space(4) + ;
200            str(st_AvgEnri,5,3)
201    else
202        line1 = line1 + space(4) + str(st_NumAssm,6) + space(8) + ;
203        str(st_AvgBurn,5) + space(6) + str(st_Wt,8,1) + space(7) + ;
204        str(st_AvgEnri,5,3)
205    endif
206    endif
207    ? line1
208    ?
209 ** Add subtotals to grand totals
210    gt_NumAssm = gt_NumAssm + st_NumAssm
211    gt_Wt = gt_Wt + st_Wt
212    gt_Burn = gt_Burn + st_Burn

```

APPENDIX A -- PROGRAM LISTINGS
PGENRPT.PRG

34

```

213     gt_Enri = gt_Enri + st_Enri
214 enddo
215
216 gt_AvgBurn = round(gt_Burn/gt_NumAssm,3)
217 gt_AvgEnri = round(gt_Enri/gt_NumAssm,3)
218
219 if (by_Bin)
220     line1 = '--- GRAND TOTALS' + space(29) + str(gt_NumAssm,6) + space(5) +
221         str(gt_AvgBurn,5) + space(3) + str(gt_Wt,8,1) + space(3) +
222         str(gt_AvgEnri,5,3)
223 else
224     if (by_Year)
225         line1 = '--- GRAND TOTALS' + space(9) + str(gt_NumAssm,6)
226     +;
226         space(8) + str(gt_AvgBurn,5) + space(4) + str(gt_Wt,8,1) + space(4)
227     +;
228         str(gt_AvgEnri,5,3)
229     else
230         line1 = '--- GRAND TOTALS' + str(gt_NumAssm,6) +;
231         space(8) + str(gt_AvgBurn,5) + space(6) + str(gt_Wt,8,1) +;
232         space(7) + str(gt_AvgEnri,5,3)
233     endif
234 endif
235 ? line1
236 return
237 ****
238 * PGenHdg -- Headings for a general report for Projected data. Called *
239 * from PGenRpt.
240 ****
241 ****
242
243 Procedure PGenHdg
244 . parameter sortBy
245
246 set alternate to qty.hdg
247 set alternate on
248
249 if (debug)
250     @ 20,0 clear
251     @ 20,10 say "In PGenHdg. sortBy: " + sortBy
252     wait "" to dummy
253 endif
254
255 ?? title1
256 strNum = str(rptNum,1)
257 line1 = "Projected Data: "
258 if (histproj = 2)           && No New Orders Case
259     line1 = line1 + "No New Orders Case"
260 else
261     line1 = line1 + "Upper Reference Case"
262 endif
263 line1 = line1 + "with Extended Burnup"
264
265 startPos = (80 - len(line1))/2
266 line1 = space(startPos) + line1
267 ? line1
268
269 line1 = pRTTitle&strNum + " through 2037"
270 startPos = (80 - len(line1))/2
271 line1 = space(startPos) + line1
272 ? line1
273
274 if (by_Bin)
275     line1 = 'DISCHARGE      BURNUP      NUMBER      BURNUP      WEIGHT      INIT'
276     line2 = '      YEAR      BIN      ASSM      (MWD/MTIHM) (MTIHM) ENRICH'
277 else
278     if (by_Year)
279         line1 = 'DISCHARGE' + space(4) + "NUMBER      BURNUP      WEIGHT      INIT"
280         "
280         line2 = 'YEAR      ' + space(2) + "ASSEMBLIES (MWD/MTIHM) (MTIHM) ENRIC

```

APPENDIX A - PROGRAM LISTINGS
PGENRPT.PRG

35

```

H"
281    else
282        line1 = space(9) + space(4) + "NUMBER      BURNUP      WEIGHT      I
NIT "
283        line2 = space(2) + "ASSEMBLIES   (MWd/MTIEM)   (MTIEM)   ENRICH"
284    endif
285 endif
286
287 ** Column Headings
288 if (by_Bin)
289    ? space(55) + " AVG      TOTAL      AVG"
290 else
291    if (by_Year)
292        ? space(52) + " AVG      TOTAL      AVG"
293    else
294        ? space(45) + " AVG      TOTAL      AVG"
295    endif
296 endif
297
298 if (by_Bin)
299    line1 = space(23) + line1
300 else
301    if (by_Year)
302        line1 = space(28) + line1
303    else
304        line1 = space(21) + line1
305    endif
306 endif
307 ? line1
308
309 line1 = sortBy
310 l = len(line1)
311 if (by_Bin)
312    line1 = line1 + space(23-1)
313 else
314    line1 = line1 + space(30-1)
315 endif
316
317 line1 = line1 + line2
318 ? line1 + carbRet + lineFeed
319
320 close alternate
321 return
322
323 -----
324 * PGen1YrB -- Total up data for 1 year, all burnup bins. Called from
325 * procedure GenRpt1 when working on report for All Assemblies or All
326 * Assemblies by Reactor Type. These 2 reports are available with burnup
327 * bin breakdowns, so the data is handled a little differently from the
328 * other general reports, where burnup bin breakdown is not available.
329 -----
330
331 Procedure PGen1YrB
332
333 private curYear
334
335 curYear = dischYear
336 ** Zero accumulators for this year
337 st2_NumAssm = 0
338 st2_Burn = 0
339 st2_Wt = 0
340 st2_Enri = 0
341
342 do while (curName = primName .and. KEY = (psummary->case+psummary->class);
343     .and. dischYear = curYear .and. .not. eof())
344
345     st2_NumAssm = st2_NumAssm + numAssm
346     st2_Burn = st2_Burn + (numAssm * avgBurn)
347     st2_Wt = st2_Wt + weight
348     st2_Enri = st2_Enri + (numAssm * avgEnri)
349

```

APPENDIX A -- PROGRAM LISTINGS
PGENRPT.PRG

36

```

350    if (by_Bin)
351        if (first)          && Need to print the primary name?
352            line1 = curGroup
353            l = len(line1)
354            line1 = line1 + space(25-l)
355            first = .f.
356        else
357            line1 = space(25)
358        endif
359
360        line1 = line1 + str(curYear,4) + space(3)
361        if (burnBin < 10)
362            strNum = str(burnBin,1)
363        else
364            strNum = str(burnBin,2)
365        endif
366
367        line1 = line1 + bb&strNum + space(3) + str(numAssm,5) + space(4)
368
369        line1 = line1 + str(avgBurn,6) + space(3) + str(weight,8,1) + space(3);
370            + str(avgEnri,5,3)
371        ? line1
372    endif
373
374    skip
375 enddo
376
377 if (by_Bin)
378    ?
379 else
380    if (by_Year)           && Printing only 1 line for the year
381        if (first)          && Need to print the primary name?
382            line1 = curGroup
383            l = len(line1)
384            line1 = line1 + space(30-l)
385            first = .f.
386        else
387            line1 = space(30)
388        endif
389
390        st2_AvgBurn = round(st2_Burn/st2_NumAssm,0)
391        st2_AvgEnri = round(st2_Enri/st2_NumAssm,3)
392
393        line1 = line1 + str(curYear,4) + space(5) + str(st2_numAssm,5) +;
394            space(8) + str(st2_AvgBurn,5) + space(4) + str(st2_Wt,8,1) +;
395            space(4) + str(st2_AvgEnri,5,3)
396        ? line1
397    endif
398 endif
399
400 ** Add this year's accumulators to subtotals for this group
401 st_NumAssm = st_NumAssm + st2_numAssm
402 st_Burn = st_Burn + st2_Burn
403 st_Wt = st_Wt + st2_Wt
404 st_Enri = st_Enri + st2_Enri
405
406 return
407

```

APPENDIX A -- PROGRAM LISTINGS
QTYINTRO.PRG

37

```
1 -----
2 * Program: qtyintro.prg (formerly sncdb.prg)
3 * Purpose: Presents intial entry screen into CDB Program for the Serial
4 * Number Data Base system.
5 * Date: 03/89
6 * Author: B. Lewis
7 -----
8
9 * Modified 5/89 for Quantities database intro -- KEJ
10 * Changed name from sncdb.prg to qtyintro.prg
11 -----
12
13 CLEAR
14 *
15 @ 0,0 TO 23,79 DOUBLE
16 @ 2,12 SAY "      CCCCCC      DDDDDDDD      BBBBBBBB"
17 @ 3,12 SAY "      CC  CC      DD  DD      BB  BB"
18 @ 4,12 SAY "      CC      DD  DD      BB  BB"
19 @ 5,12 SAY "      CC      DD  DD      BBBBBBBB"
20 @ 6,12 SAY "      CC      DD  DD      BB  BB"
21 @ 7,12 SAY "      CC  CC      DD  DD      BB  BB"
22 @ 8,12 SAY "      CCCCCC      DDDDDDDD      BBBBBBBB"
23 *
24 @ 10,12 SAY "          (Characteristics Data Base)"
25 *
26 @ 12,1 TO 12,78 DOUBLE
27 @ 13,1 say "      LWR QUANTITIES DATA BASE (QDB): Provides data about dischar
ged"
28 @ 14,1 say "      nuclear fuel assemblies from LWR reactors. This data includ
es"
29 @ 15,1 say "      previously discharged assemblies and projections by EIA of"
30 @ 16,1 say "      assemblies to be discharged up to the year 2037."
31 *
32 @ 18,1 TO 18,78 DOUBLE
33 @ 19,1 SAY "      For Assistance      Karl J. Notz      or      R. Scott Mo
ore"
34 @ 20,1 SAY "      Contact:      (615) 574-6632      (615) 482-6
601"
35 @ 21,1 SAY "          FTS 624-6632"
36
37 @ 23,26 say "Strike any key to continue"
38 *
39 **num=23
40 **DO rtrn WITH num
41 *
42 RETURN
```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

38

```

1 ****
2 * QTYMENU.PRG-- library of procedures for menus for the LWR Quantites *
3 * Database. *
4 ****
5
6 ****
7 * Sequence -- Sequences through all the choices on the screen. *
8 ****
9
10 procedure SEQUENCE
11
12 do HpS1ct
13 do WriteIt with "H", hpTxt
14 if (histProj = 1)           && Historical data
15   do HRptS1ct
16 else                        && Projected data
17   do PRptS1ct
18 endif
19
20 do brkDS1ct
21 do WriteIt with "B",brkDTxt
22 do DevS1ct with 3
23 do WriteIt with "O",oDTxt
24
25 return
26
27 ****
28 * HPs1ct -- user selects historical or projected data. *
29 ****
30
31 procedure HPs1ct
32
33 set color to w/n
34 @ 11,0 clear
35
36 set color to n/w
37 @ 24,60 say "F1-->Help"
38 set color to w/n
39
40 @ 11, 1 say '(A.) Select the type of Data: '
41
42 @ 13,5 say ' 1. '
43 @ 14,5 say ' 2. '
44 *@ 15,5 say ' 3. '
45
46 @ 13, 9 say hptxt1
47 @ 14, 9 say hptxt2 + "with Extended Burnup"
48 *@ 15, 9 say hptxt3 + " with Extended Burnup"
49
50 choice = 0
51 @ 24, 1 say 'Enter your selection (1-2):'
52 set color to w/n
53 do while choice = 0
54   @ 24,30 get choice picture '9' range 1,2
55     read
56 enddo
57
58 choicechr = str(choice,1)
59 histProj = choice
60 hpTxt = hpTxt&choicechr
61 @11,1 clear
62
63 set color to n/w
64 @ 24,60 say "F1-->Help"
65 set color to w/n
66
67 return
68
69 ****
70 * HRptS1ct -- user selects Data Requested -- for historical data. *
71 ****

```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

39

```

72
73 procedure HRptSlect
74
75 set color to w/n
76 @ 11,0 clear
77
78 set color to n/w
79 @ 24,60 say "F1-->Help"
80 set color to w/n
81
82 @ 11, 1 say '(B.) Select the Data Requested: '
83
84 @ 13,1 say ' 1. '
85 @ 14,1 say ' 2. '
86 @ 15,1 say ' 3. '
87 @ 16,1 say ' 4. '
88 @ 17,1 say ' 5. '
89 @ 18,1 say ' 6. '
90 @ 19,1 say ' 7. '
91
92 @ 13, 5 say hRptTxt1
93 @ 14, 5 say hRptTxt2
94 @ 15, 5 say hRptTxt3
95 @ 16, 5 say hRptTxt4
96 @ 17, 5 say hRptTxt5
97 @ 18, 5 say hRptTxt6
98 @ 19, 5 say hRptTxt7
99
100 @ 14,44 say ' 8. '
101 @ 15,44 say ' 9. '
102 @ 16,44 say ' 10. '
103 @ 17,44 say ' 11. '
104 @ 18,44 say ' 12. '
105 @ 19,44 say ' 13. '
106
107 @ 14, 49 say trim(hRptTxt8)
108 @ 15, 49 say trim(hRptTxt9)
109 @ 16, 49 say trim(hRptTxt10)
110 @ 17, 49 say trim(hRptTxt11)
111 @ 18, 49 say trim(hRptTxt12)
112 @ 19, 49 say trim(hRptTxt13)
113
114 choice = 0
115 @ 24, 1 say 'Enter your selection (1-13):'
116 set color to w/n
117 do while choice = 0
118     @ 24,30 get choice picture '99' range 1,13
119     read
120 enddo
121
122 if (choice < 10)
123     choicechr = str(choice,1)
124 else
125     choicechr = str(choice,2)
126 endif
127
128 rptNum = choice
129 rptTxt = hRptTxt&choicechr
130 @11,1 clear
131
132 set color to n/w
133 @ 24,60 say "F1-->Help"
134 set color to w/n
135
136 do WriteIt with "R", rptTxt
137
138 if (rptNum >= 8)          && Specific report, need to get specific name
139     do SpecName
140 endif
141
142 *** If user chose a specific report, call SubTSlct to let him select

```

APPENDIX A -- PROGRAM LISTINGS QTYMENU.PRG

40

```
143 *** subtotal options.
144 if (rptNum >= 8)
145   do SubtSlct
146 else
147   subTTxt = subTBlink
148 endif
149 do WriteIt with "S", subTTxt
150
151 return
152 ****
153 * PRptSlct -- user selects Data Requested -- for projected data. *
154 ****
155 ****
156
157 procedure PRptSlct
158
159 set color to w/n
160 @ 11,0 clear
161
162 set color to n/w
163 @ 24,60 say "F1-->Help"
164 set color to w/n
165
166 @ 11, 1 say '(B.) Select the Data Requested: '
167
168 @ 13,1 say ' 1. '
169 @ 14,1 say ' 2. '
170 @ 15,1 say ' 3. '
171 @ 16,1 say ' 4. '
172 @ 17,1 say ' 5. '
173
174 @ 13, 5 say hRptTxt1
175 @ 14, 5 say hRptTxt2
176 @ 15, 5 say hRptTxt3
177 @ 16, 5 say hRptTxt6
178 @ 17, 5 say hRptTxt7
179
180 @ 14,44 say ' 6. '
181 @ 15,44 say ' 7. '
182 @ 16,44 say ' 8. '
183 @ 17,44 say ' 9. '
184
185 @ 14, 49 say trim(hRptTxt8)
186 @ 15, 49 say trim(hRptTxt9)
187 @ 16, 49 say trim(hRptTxt12)
188 @ 17, 49 say trim(hRptTxt13)
189
190 choice = 0
191 @ 24, 1 say 'Enter your selection (1-9):'
192 set color to w/n
193 do while choice = 0
194   @ 24,30 get choice picture '99' range 1,9
195   read
196 enddo
197
198 *** Adjust numbers to match historical reports
199 do case
200   case choice = 4      // General, assembly class
201     choice = 6
202
203   case choice = 5      // General, reactor type
204     choice = 7
205
206   case choice = 6      // Specific, utility
207     choice = 8
208
209   case choice = 7      // Specific, reactor
210     choice = 9
211
212   case choice = 8      // Specific, assembly class
213     choice = 12
```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

41

```

214
215     case choice = 9          && Specific, reactor type
216         choice = 13
217     endcase
218
219     if (choice < 10)
220         choicechr = str(choice,1)
221     else
222         choicechr = str(choice,2)
223     endif
224
225     rptNum = choice
226     rptTxt = hRptTxt&choicechr
227     @11,1 clear
228
229     set color to n/w
230     @ 24,60 say "F1-->Help"
231     set color to w/n
232
233     do WriteIt with "R", rptTxt
234
235     if (rptNum >= 8)          && Specific report, need to get specific name
236         do SpecName
237     endif
238
239     subTTxt = subTBlnk
240     do WriteIt with "S", subTTxt
241
242     return
243
244 ****
245 * SubTS1ct -- user selects subtotal option. Called only if he selects *
246 * a specific report, for historical data. *
247 ****
248
249 procedure SubTS1ct
250
251     set color to w/n
252     @ 11,0 clear
253
254     @ 11, 1 say '(B.) Select the subtotal option: '
255     @ 12,5 say 'For this report, you may choose to see the data subtotalled'
256     @ 13,5 say 'by these additional categories.'
257
258     endMag = ''
259     do case
260         case rptNum = 10          && Assemblies by storage pool
261             @ 15,5 say ' 1. ' + subTTxt2
262             @ 16,5 say ' 2. ' + subTTxt3
263             endMag = '2):'
264
265         case rptNum = 11          && Assemblies by assembly type
266             @ 15,5 say ' 1. ' + subTTxt1
267             @ 16,5 say ' 2. ' + subTTxt3
268             endMag = '2):'
269
270         otherwise
271             @ 15,5 say ' 1. ' + subTTxt1
272             @ 16,5 say ' 2. ' + subTTxt2
273             @ 17,5 say ' 3. ' + subTTxt3
274             endMag = '3):'
275
276     endcase
277
278     choice = 0
279     msg = 'Enter your selection (1-' + endMag
280
281     @ 24, 1 say msg
282     set color to w/n
283     do while choice = 0
284         if (rptNum = 10 .or. rptNum = 11)

```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

42

```

285      @ 24,30 get choice picture '9' range 1,2
286      else
287          @ 24,30 get choice picture '9' range 1,3
288      endif
289      read
290  enddo
291
292 do case
293     case rptNum = 10           ** Assemblies by Storage Pool
294         choice = choice + 1
295
296     case rptNum = 11           ** Assemblies by Assembly Type
297         if (choice = 2)
298             choice = 3
299         endif
300  endcase
301
302 choicechr = str(choice,1)
303 subTotal = choice
304 subTTxt = subTTxt&choicechr
305 @11,1 clear
306
307 return
308 ****
309 * BrkDS1ct -- user selects data broken down by. *
310 ****
311 ****
312
313 procedure BrkDS1ct
314
315 set color to w/n
316 @ 11,0 clear
317
318 set color to n/w
319 @ 24,60 say "F1-->Help"
320 set color to w/n
321
322 @ 11, 1 say '(C.) Select how you want the data broken down:'
323
324 @ 13,5 say ' 1. ' + brkDTxt1
325 if (rptNum >= 8 .or. rptNum = 1 .or. rptNum = 7)
326     @ 14,5 say ' 2. ' + brkDTxt2
327     if (histProj = 1)
328         @ 15,5 say ' 3. ' + brkDTxt3
329     else
330         @ 15,5 say ' 3. ' + brkPTxt3
331     endif
332 else
333     if (histProj = 1)
334         @ 14,5 say ' 2. ' + brkDTxt3
335     else
336         @ 14,5 say ' 2. ' + brkPTxt3
337     endif
338 endif
339
340 *** Special case. If specific report, or general report for All Assemblies
341 ** or all assemblies by Reactor Type, you can break down by
342 *** Burnup bin and discharge year.
343
344 msg = "Enter your selection (1-"
345 if (rptNum >= 8 .or. rptNum = 1 .or. rptNum = 7)
346     msg = msg + '3):'
347 else
348     msg = msg + '2):'
349 endif
350
351 choice = 0
352 @ 24, 1 say msg
353 set color to w/n
354 do while choice = 0
355     if (rptNum >= 8 .or. rptNum = 1 .or. rptNum = 7)

```

APPENDIX A – PROGRAM LISTINGS
QTYMENU.PRG

43

```

356      @ 24,30 get choice picture '9' range 1,3
357      else
358          @ 24,30 get choice picture '9' range 1,2
359      endif
360      read
361  enddo
362
363 if .not. (rptNum >= 8 .or. rptNum = 1 .or. rptNum = 7)
364     if (choice = 2)
365         choice = 3
366     endif
367 endif
368
369 choicechr = str(choice,1)
370 brkDown = choice
371 if (choice = 3)
372     if (histProj = 1)           ** Historical data
373         brkDTxt = brkDTxt3
374     else                       ** Projected data
375         brkDTxt = brkPTxt3
376     endif
377 else
378     brkDTxt = brkDTxt&choicechr
379 endif
380 @11,1 clear
381
382 set color to n/w
383 @ 24,60 say "F1-->Help"
384 set color to w/n
385
386 return
387
388 ****
389 *   DevSlt -- user selects the output device *
390 ****
391
392 Procedure DevSlt
393     parameter numDev
394
395 private skipFlag
396
397 set color to w/n
398 if (numDev = 3)
399     @ 11,0 clear
400 else
401     clear
402 endif
403
404 set color to n/w
405 @ 24,60 say "F1-->Help"
406 set color to w/n
407
408 @ 11, 1 say '(D.) Select the output device:'
409
410 @ 13,5 say ' 1. '
411 @ 14,5 say ' 2. '
412 @ 15,5 say ' 3. '
413
414 if (numDev = 3)
415     @ 13, 9 say odTxt1
416     @ 14, 9 say odTxt2
417     @ 15, 9 say odTxt3
418 else                      ** Not enough memory for screen, just file or print
419     @ 13, 9 say odTxt2
420     @ 14, 9 say odTxt3
421     @ 15, 9 say "Exit"
422 endif
423
424 @ 17,5 say "If you select Disk File, you will be prompted for the file name."
425
426 choice = 0

```

APPENDIX A – PROGRAM LISTINGS

QTYMENU.PRG

44

```

427 endMsg = "3) :"
428 @ 24, 1 say 'Enter your selection (1-' + endMsg
429 set color to w/n
430
431 do while choice = 0
432     @ 24,30 get choice picture '9' range 1,3
433     read
434 enddo
435
436 skipFlag = .f.
437 if (numDev < 3)
438     if (choice = 3)          && Exit
439         skipFlag = .t.
440     else
441         choice = choice + 1
442     endif
443 endif
444
445 if (.not. skipFlag)
446     choicechr = str(choice,1)
447     odTxt = odTxt&choicechr
448     oDevice = oDevice&choicechr
449
450 *** If user chose Disk File, ask him for file name.
451
452 if (choice = 3)
453     overWrFlag = .f.
454     set color to w/n,w/n
455     oldFName = fName
456     @ 22,1 say "Enter the file name to which you would like to send data:"
457     fileOK = .f.
458     do while (.not. fileOK)
459         @ 23,10 get fName
460         read
461         if (fName <> "        ")
462             do ChkFName with fileOK
463         endif
464         if (.not. fileOK)
465             @ 24,1 say 'That file name has a reserved extension. Please use
a different extension.'
466         endif
467     enddo
468
469     if (file(fName))
470         @ 24,1 say trim(fName) + " already exists. Overwrite or Append? (O/
A)"
471         fChoice = ''
472         do while .not.(fChoice $ "QA")
473             @ 24,58 get fChoice picture 'I'
474             read
475         enddo
476
477         if (fChoice = "O")           && Overwrite, destroy existing file
478             set alternate to &fName
479             close alternate
480             overWrFlag = .t.
481         endif
482     endif
483
484     odTxt = fName
485     endif
486 endif                                && If skipFlag
487
488 @11,0 clear
489
490 set color to n/w
491 @ 24,60 say "F1-->Help"
492 set color to w/n
493
494 return
495

```

APPENDIX A – PROGRAM LISTINGS
QTYMENU.PRG

45

```

496 ****
497 * ChkFName -- check that the user-specified filename does not have a *
498 * reserved extension of TAB, DBF, NTX. *
499 ****
500
501 procedure ChkFName
502     parameter fileOK
503
504     fName2 = trim(fName)
505     dotPos = AT('.',fName2)
506     l = len(fName2)
507     fileOK = .t.
508     if (dotPos > 0 .and. dotPos <= l)
509         extens = upper(substr(fName,dotPos+1,3))
510     do case
511         case extens = 'DBF'
512             fileOK = .f.
513         case extens = 'NTX'
514             fileOK = .f.
515         case extens = 'TAB'
516             fileOK = .f.
517     endcase
518     else
519         fName = fName2 + '.'
520     * Adjust length to 12
521         l = l + 1
522         if (l < 12)
523             fName = fName + space(12 - l)
524         endif
525     endif
526 return
527 ****
528 * Rtslct -- user selects reactor type (PWR or BWR). Called from routine*
529 * SpecName, when user has selected report #11, assemblies from 1 reactor *
530 * type. *
531 ****
532
533
534 procedure Rtslct
535
536 set color to w/n
537 @ 11,0 clear
538
539 @ 11, 1 say '(B.) Select the Reactor Type:'
540
541 @ 13,1 say '1. BWR reactors'
542 @ 14,1 say '2. PWR reactors'
543
544 choice = 0
545 msg = 'Enter your selection (1-2)'
546
547 @ 24, 1 say msg
548 set color to w+/n
549 do while choice = 0
550     @ 24,30 get choice picture '9' range 1,2
551     read
552 enddo
553
554 if (choice = 1)
555     rptTxt = "BWR reactors"
556     rTName = "BWR"
557 else
558     rptTxt = "PWR reactors"
559     rTName = "PWR"
560 endif
561
562 @11,0 clear
563
564 do WriteIt with "R", rptTxt
565
566 return

```

APPENDIX A - PROGRAM LISTINGS
QTYMENU.PRG

46

```

567
568
569 **** ChkRpt -- Checks that the report selected matches the selection of *
570 * historical or projected data. Called from start when user has just *
571 * changed his selection for option A, historical or projected data. *
572 ****
573 ****
574
575 procedure ChkRpt
576     parameter rptOK
577
578     rptOK = .f.
579     if (histProj = 1)           && Historical data
580         if (rptNum >= 8)        && Specific reports, check the specific name
581             do ChkSpec with rptOK
582         else
583             rptOK = .t.          && Every report ok for historical
584         endif
585     else                      && Just changed to projected data
586         if ((rptNum >= 1 .and. rptNum <= 3) .or. rptNum = 6 .or. rptNum = 7)
587             rptOK = .t.
588         else
589             if (rptNum = 8 .or. rptNum = 9 .or. rptNum = 12 .or. rptNum = 13)
590                 do ChkSpec with rptOK
591             else
592                 rptOK = .f.
593             endif
594         endif
595
596 *** If just changed to projected, no subtotals allowed
597     subTTxt = subTBlnk
598     do WriteIt with "S",subTTxt
599
600     endif
601
602     if (.not. rptOK)
603         rptTxt = "*** Must be re-defined"
604         do WriteIt with "R",rptTxt
605     endif
606
607     if (debug)
608         @ 21,0 clear
609         @ 21,10 say "Leaving ChkRpt-- rptOK:"
610         @ 21,45 say rptOK picture 'Y'
611         wait "" to dummy
612     endif
613     return
614
615 ****
616 * ChkBrkD -- check that the selection for data broken down by is OK for *
617 * currant selections of historical/projected and Data Requested. *
618 * If not, set parameter to false. *
619 ****
620
621 procedure ChkBrkD
622     parameter brkDOK
623
624     if (brkDown <> 2)
625         if (brkDown = 3)           && Totals only option has different text
626             if (histProj = 1)        && Historical data
627                 brkDTxt = brkDTxt3
628             else                     && Projected data
629                 brkDTxt = brkPTxt3
630             endif
631         endif
632
633         brkDOK = .t.            && Options 1 and 3 are always valid
634     else
635         if (rptNum < 8 .and. rptNum <> 1 .and. rptNum <> 7)      && General report
636             , not OK
636             brkDOK = .f.          && except for All and by Reactor Type

```

APPENDIX A - PROGRAM LISTINGS
QTYMENU.PRG

47

```

637     else
638         brkDOK = .t.          && Specific report, OK
639     endif
640 endif
641
642 if (.not. brkDOK)
643     brkDTxt = **** Must be re-defined
644     do WriteIt with "B",brkDTxt
645 endif
646 return
647
648 ****
649 *   Refresh -- Write variable values into the screen. *
650 ****
651
652 procedure REFRESH
653     set color to n/w
654     @ 3,33 say hpTxt
655     @ 4,33 say rptTxt
656     @ 5,36 say subTTxt
657     @ 6,33 say brkDTxt
658     @ 8,33 say odTxt
659     @ 14,26 say mChoice
660     return
661
662 ****
663 *   writeIt -- Write a specific variable value into the screen. The first*
664 * parameter, code, indicates which variable to write. *
665 *   The purpose of this procedure is to speed up the program by rewriting *
666 * only the variable that has changed in value, rather than re-writing all *
667 * values. *
668 ****
669
670 procedure WriteIt
671     parameters code, value
672     set color to n/w
673 *if (debug)
674 *     @ 21,0 clear
675 *     @ 21,10 say "In WriteIt. code: " + code
676 *     @ 22,10 say "value: " + value
677 *     wait "" to dummy
678 *endif
679     do case
680         case code = "B"
681             @ 3,33 say value
682         case code = "R"
683             @ 4,33 say value
684         case code = "S"
685             @ 5,36 say value
686         case code = "B"
687             @ 6,33 say value
688         case code = "O"
689             @ 8,33 say value
690     endcase
691 return
692
693 ****
694 * SpecName -- Get specific name (reactor, utility, etc) from user. First *
695 * routine just sets up look up tables, etc., then calls GetName to get the *
696 * name of the entity. *
697 ****
698
699 Procedure SpecName
700
701 select 5                      && Lookup area
702 do case
703     case rptNum = 8
704         if (histProj = 1)      && Historical data
705             use &dbfPath.utility index &dbfPath.utlName
706         else
707             use &dbfPath.utility index &dbfPath.putlName

```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

48

```

708      endif
709      thing = "utilities"
710      entity = "utilcode"
711      msg = "Utility Name"
712
713      case rptNum = 9
714          if (histProj = 1)      && Historical data
715              use &dbfPath.reactor index &dbfPath.recname
716          else
717              use &dbfPath.preactor index &dbfPath.precName
718          endif
719          thing = "reactors"
720          entity = "recode"
721          msg = "Reactor Name"
722
723      case rptNum = 10
724          use &dbfPath.pool index &dbfPath.poolName    && Only for historical dat
725
726          thing = "pools"
727          entity = "pool"
728          msg = "Pool Name"
729
730      case rptNum = 11           && Assembly type
731          use &dbfPath.assmtype index &dbfPath.assmName
732          thing = "assembly types"
733          entity = "assmtype"
734          msg = "Assembly Type"
735
736      case rptNum = 12           && Assembly class
737          if (histProj = 1)      && Historical data
738              use &dbfPath.assmclas index &dbfPath.assmClas
739          else
740              use &dbfPath.pAssmcls index &dbfPath.pAssmCls
741          endif
742          thing = "assembly class"
743          entity = "assmclas"
744          msg = "Assembly Class"
745
746      case rptNum = 13           && One reactor type
747          do RTSlct      && Simpler case, just 2 reactor types to choose from
748          return
749      endcase
750
751      eName = ' '
752      if (histProj > 1)          && Projected data
753          pCase = str((histProj - 1),1)
754      else
755          pCase = ''
756      endif
757      do GetName with msg, thing, eName, entity, pCase
758
759      do case
760          case rptNum = 8
761              utilName = eName
762
763          case rptNum = 9
764              reacName = eName
765
766          case rptNum = 10
767              poolName = eName
768
769          case rptNum = 11
770              assmName = eName
771
772          case rptNum = 12
773              assmCName = eName
774
775          case rptNum = 13
776              rTName = eName
777      endcase

```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

49

```

778
779 *** Close open database file to avoid any confusion
780 select 5
781 use
782 return
783 ****
784 * GetName -- Input particular name (of utility, reactor, etc.) from user. *
785 * Called from routine SpecName, when user choose a specific report. *
786 ****
787 ****
788
789 Procedure GetName
790   parameter msg, thing, eName, entity, pCase
791
792 private foundIt, isUnique, t, l3, b, r, l, flds, hdgs, mName, mName2, winBuf
793 private fldName
794
795 if (debug)
796   @ 20,0 clear
797   @ 20,10 say "In GetName. "
798   wait "" to dummy
799 endif
800
801 declare flds[1], hdgs[1]
802 set color to w/n,n/w
803 @ 11,0 clear
804
805 line1 = "(B). Data Requested: " + trim(rptTxt)
806 @ 12,1 say line1
807 @ 14,5 say msg + ":" 
808
809 @ 16,1 say "You can either: "
810 @ 17,5 say "Enter the entire name"
811 @ 18,5 say "Enter only the first character to see a list of " + thing + " tha
t"
812 @ 19,10 say "start with that letter"
813 @ 20,5 say "Enter a ? to see a list of all " + thing
814
815 mName = '
816 more = .t.
817 do while (more)
818
819   found = .f.
820   @ 14,35 get mName picture '!!!!!!'
821   read
822
823   mName2 = trim(mName)
824   l = len(mName2)
825
826   foundit = .t.
827   isUnique = .f.
828   if (mName2 = '?')
829     go top
830   else
831     seek mName2
832     do ChkUniq with mName2, foundIt, isUnique
833   endif
834
835   if (debug)
836     @ 20,0 clear
837     @ 20,10 say "After ChkUniq. foundIt: "
838     @ 20,40 say foundIt picture 'Y'
839     @ 21,10 say "isUnique: "
840     @ 21,30 say isUnique picture 'Y'
841     wait "" to dummy
842     AltD()
843   endif
844
845   if (.not. isUnique)
846     t = 10
847     l3 = 45

```

APPENDIX A -- PROGRAM LISTINGS
QTYMENU.PRG

50

```

848     b = 23
849     r = 73
850
851     winBuf = SaveScreen(t,13,b,r)
852
853 ** 12/1/89 -- Scott requested we don't show the codes, just the names
854     fldName = "name"
855 ** flds[1] = "&entity + space(5) + name"
856 ** hdgs[1] = "CODE      NAME"
857
858     flds[1] = "&fldName"
859     hdgs[1] = "NAME"
860
861     @ t, 13, b, r box doubleBox
862     @ t+1, 13+1 clear to b-1, r-1
863     if (debug)
864         @ 20,0 clear
865         @ 20,10 say "About to call dbedit.  t: " + str(t,3) + "    13: "
866         + str(13,3)
867         @ 21,10 say "b: " + str(b,3) + "    r: " + str(r,3)
868         @ 22,10 say "entity: " + &entity
869         wait "" to dummy
870     endif
871
872     DbEdit(t+1, 13+1, b-1, r-1, flds,"","","",hdgs,"-",', ')
873
874     RestScreen(t, 13, b, r, winBuf)
875     if (lastKey() = 27)
876         more = .t.
877     else
878         rptTxt = msg + ":" + name
879         eName = &entity
880         more = .f.
881     endif
882     else
883         rptTxt = msg + ":" + name
884         eName = &entity
885         more = .f.
886     endif
887
888     @ 11,0 clear
889     l = len(rptTxt)
890     if (l < 40)
891         rptTxt = rptTxt + space(40 - l)
892     else
893         if (l > 40)
894             rptTxt = substr(rptTxt,1,40)
895         endif
896     endif
897
898 do WriteIt with "R", rptTxt
899 return
900
901 -----
902 * ChkUniq -- checks that the name entered is a unique name for this table.
903 * Called from GetName procedure.
904
905 * Parameters
906 *   mName2      char      the name we're looking for
907 *   foundIt     logical   Is the name found?
908 *   isUnique    logical   Is this a unique match?
909 -----
910
911 Procedure ChkUniq
912     parameters mName2, foundIt, isUnique
913
914 foundIt = .f.
915 isUnique = .f.
916
917 if (histProj = 1)

```

APPENDIX A - PROGRAM LISTINGS
QTYMENU.PRG

51

```

918     seek mName2
919   else          && Projected data, need to check case
920     seek pCase + mName2
921   endif
922
923   if (debug)
924     @ 20,0 clear
925     @ 20,10 say "In ChkUniq. seeking: " + mName2
926     @ 21,10 say "found(): "
927     @ 21,30 say found() picture 'Y'
928     wait "" to dummy
929   endif
930
931   if (found())
932     foundIt = .t.
933     skip 1
934
935   if (upper(name) = mName2)
936     isUnique = .f.
937   else
938     isUnique = .t.
939   endif
940     skip -1
941   else
942     skip -1
943   endif
944
945   return
946
947 ****
948 * ChkSpec -- User has just changed option A, historical or projected data. *
949 * If he has specified a specific report, check that the name of the      *
950 * reactor or whatever is ok for the new data type.                      *
951 ****
952
953 Procedure ChkSpec
954   parameter rptOK
955
956   select 5
957   do case
958     case rptNum = 8
959       if (histProj = 1)      && Historical data
960         use &dbfPath.utility index &dbfPath.utility
961       else
962         use &dbfPath.putility index &dbfPath.putility
963       endif
964       thing = "utilName"
965
966     case rptNum = 9
967       if (histProj = 1)      && Historical data
968         use &dbfPath.reactor index &dbfPath.reactor
969       else
970         use &dbfPath.preactor index &dbfPath.preactor
971       endif
972       thing = "reacName"
973
974     case rptNum = 10
975       use &dbfPath.pool index &dbfPath.poolName    && Only for historical dat
976
977     case rptNum = 11
978       use &dbfPath.assmtype index &dbfPath.assmName
979       thing = "assmName"
980
981     case rptNum = 12      && Assembly class
982       use &dbfPath.assmtype index &dbfPath.assmName
983       thing = "assmCName"
984
985     case rptNum = 13
986       if (histProj = 1)      && Historical data

```

```

988         use &dbfPath.reactype index &dbfPath.reactype
989     else
990         use &dbfPath.preactyp index &dbfPath.preactyp
991     endif
992     thing = "rtName"
993
994 endcase
995
996 if (histProj > 1)                                && Projected data
997     pCase = str((histProj - 1),1)
998 else
999     pCase = ''
1,000 endif
1,001
1,002 var = pCase + &thing
1,003 seek var
1,004 rptOK = found()
1,005 use                                         && Close database
1,006 return

```

APPENDIX A - PROGRAM LISTINGS
RUNRPT.PRG

53

```

1 ****
2 * RUNRPT.PRG Sets up flags, etc., to run the reports. *
3 ****
4 * 05/02/89 -- We have changed the projected data to come from EIA rather *
5 * than EIA. We currently will have only 1 case (no new *
6 * orders with extended burnup). However, all the code for *
7 * multiple cases has been left in so we can go back to *
8 * multiple cases in future revisions easily. *
9 ****
10
11     parameter getData
12
13 if (newParms)
14     getData = .f.
15 general = .f.
16 specific = .f.
17 by_Year = .f.
18 by_Bin = .f.
19 by_Pool = .f.
20 by_Assmt = .f.
21
22 if (debug)
23     @ 20,0 clear
24     @ 20,10 say "In RunRpt. histProj: " + str(histProj,2)
25     wait "" to dummy
26 endif
27
28 set console off
29
30 if (histProj = 1)           && Historical data
31     if (rptNum < 8)          && General reports
32         if brkDown = 1
33             by_Year = .t.
34         else
35             if (brkDown = 2)
36                 by_Year = .t.
37                 by_Bin = .t.
38             endif
39         endif
40         general = .t.
41         set proc to gen rpt
42         do GenSetup with getData
43     else                      && Specific reports
44         specific = .t.
45 if (debug3)
46     @ 21,0 clear
47     @ 21,10 say "About to check brkDown. brkDown: "
48     @ 21,65 say brkDown
49     wait "" to dummy
50 endif
51     do case
52         case brkDown = 1
53             by_Year = .t.
54
55         case brkDown = 2
56             by_Bin = .t.
57     endcase
58 if (debug3)
59     @ 21,0 clear
60     @ 21,10 say "Just set by_Bin. By_Bin: "
61     @ 21,65 say by_Bin
62     wait "" to dummy
63 endif
64 if (debug3)
65     @ 21,0 clear
66     @ 21,10 say "About to check subtotal. subTotal: "
67     @ 21,65 say subTotal
68     wait "" to dummy
69 endif
70
71     do case

```

APPENDIX A - PROGRAM LISTINGS
RUNRPT.PRG

54

```

72      case subTotal = 1
73          by_Pool = .t.
74
75      case subTotal = 2
76          by_AsmT = .t.
77  if (dmbug3)
78      @ 21,0 clear
79      @ 21,10 say "by_AsmT is true."
80      wait "" to dummy
81  endif
82      endcase
83
84 *** Check for special cases -- where defective data is not available
85 ** Commented out 10/30/89 -- no problems with defective data this year
86      noDefc1 = .f.
87  *      if (rptNum = 8 .and. (utilName = "10"));
88  *          .or. (rptNum = 9 .and. (substr(reacName,1,2) = "10"));
89  *          .or. (rptNum = 10 .and. (substr(poolName,1,2) = "10"))
90  *          noDefc1 = .t.
91  *      endif
92
93  if (debug3)
94      @ 21,0 clear
95      @ 21,10 say "noDefc1: "
96      @ 21,40 say noDefc1 picture 'Y'
97      wait "" to dummy
98  endif
99      set proc to DetRpt
100     do DetSetup with getData
101    endif
102 else                      && Projected data
103   if (histproj = 2)        && No New Orders Case
104       pCase = "1"
105   else                     && Upper Reference case -- 5/2/89--we don't
106       .                         && have this case anymore, but left code in for
107       .                         && future revisions
108       pCase = "2"
109  endif
110
111
112  if (rptNum < 8)
113      general = .t.
114      if (brkDown = 1)
115          by_Year = .t.
116      else
117          if (brkDown = 2)
118              by_Bin = .t.
119              by_Year = .t.
120          endif
121      endif
122
123      set proc to PGenRpt
124      do PGSetup with pCase, getData
125  else
126      specific = .t.
127      do case
128          case brkDown = 1
129              by_Year = .t.
130
131          case brkDown = 2
132              by_Bin = .t.
133      endcase
134
135      set proc to PDetRpt
136      do PDSsetup with pCase, getData
137  endif
138
139 endif
140
141 if (getData)
142     ? carbRat + lineFeed

```

APPENDIX A – PROGRAM LISTINGS
RUNRPT.PRG

55

```

143     close alternate
144     set alternate off
145     set console on
146 endif
147 endif
148
149 if (gotData)
150   if oDevice = 'Screen'
151   **      run hdisplay qty.rpt
152   rc = Overlay("hdisplay qty.rpt",0,"","")
153   if (file("NOROOM.DAT"))           && Hdisplay didn't have enough memo
154   ry
155     clear
156     run del noroom.dat
157     do DevSlt with 2
158   endif
159 endif
160 *** Print or file
161 set color to n/gb
162 clear
163 @ 1,29 say "LWR QUANTITIES DATABASE"
164 if (oDevice = 'Print')
165   @ 10,20 say 'Your report is being sent to the Printer.'
166   run dbprint qty.rpt > nul
167   @ 12,20 say "Strike any key to continue."
168   wait "" to dummy
169 endif
170
171 if (oDevice = 'Text File')
172   @ 10,20 say 'Your report is being sent to ' + fName
173   run copy &fName + qty.hdg + qty.rpt &fName > nul
174   @ 12,20 say "Strike any key to continue."
175   wait "" to dummy
176 . endif
177 else
178   do NoData
179 endif
180
181 return
182
183 -----
184 * NoData -- Put up a message that there is no data for these conditions.
185 * This condition should not occur in the production database, only in the
186 * test system. But it may be useful if there is, by accident, a no data
187 * case.
188 -----
189
190 Procedure NoData
191
192 set color to w+/n, n/w
193 @ 16,20 say "-----"
194 @ 17,20 say "----- No data to fit these criteria. -----"
195 @ 18,20 say "----- Strike any key to continue. . ."
196 @ 19,20 say "-----"
197 wait "" to dummy
198 set color to gr+/b,n/w
199 @ 16,20 clear to 19,61
200 return

```

APPENDIX A -- PROGRAM LISTINGS
START.PRG

56

```
1 ****
2 * Start.prg -- start module for the Quantities Data Base. This data base *
3 * will provide information about historical and projected inventories of *
4 * LWR fuel assemblies. For historical data, the assembly data can be *
5 * broken down by assembly type, reactor, utility, storage pool, and reactor*
6 * type. It can be further broken down by discharge year, and burnup bin. *
7 * The projected data can only be broken down by reactor type, utility, *
8 * reactor. It can be further broken down by discharge year only. *
9 ****
10
11 set status off
12 set scoreboard off
13 set talk off
14 set heading off
15 set bell off
16 set confirm off
17 set exact off
18 set safety off
19 set softseek on
20
21 public debug,debug2,dummy,clipper, lineFeed, cargReg,choice, mChoice
22 public by_Bin, by_BinH, by_Year, by_Pool, by_AssemT
23 public histProj, rptNum, subTotal, brkDown, oDevice, odTxt
24 public hpTxt, rptTxt, subTTxt, brkDTxt, fName, dbfPath, overWrFlag
25
26 public key, key2, reacName, utilName, poolName, assemTName, reactName,;
27     assmCName
28 clear
29
30 select 1
31 use sysdat
32 dbfPath = trim(SymDat->dbfP)
33 lastDate = trim(SymDat->lastDate)
34 use
35 .
36 debug = .f.
37 debug2 = .t.
38 debug3 = .f.
39 formFeed = ""
40 lineFeed = chr(10)
41 cargRet = chr(13)
42 overWrFlag = .f.
43 doubleBox = chr(201) + chr(205) + chr(187) + chr(186) + chr(188) + chr(205)+;
44     chr(200) + chr(186)
45
46
47 *load twenty
48 *load ClipPop
49 *call twenty
50
51 *** Initialize some variables
52
53 fName = space(12)
54 ** Burnup Bin ranges
55 bb0 = " 0- 5000"
56 bb1 = " 5000-10000"
57 bb2 = "10000-15000"
58 bb3 = "15000-20000"
59 bb4 = "20000-25000"
60 bb5 = "25000-30000"
61 bb6 = "30000-35000"
62 bb7 = "35000-40000"
63 bb8 = "40000-45000"
64 bb9 = "45000-50000"
65 bb10 = "50000-55000"
66 bb11 = "55000-up   "
67
68 title1 = "LWR QUANTITIES DATABASE"
69 startPos = (80 - len(title1))/2
70 title1 = space(startPos) + title1
71
```

APPENDIX A - PROGRAM LISTINGS
START.PRG

57

```

72 ** Historical Report titles
73 hRTtitle1 = "All Discharged Assemblies"
74 hRTtitle2 = "Discharged Assemblies by Utility"
75 hRTtitle3 = "Discharged Assemblies by Reactor"
76 hRTtitle4 = "Discharged Assemblies by Storage Pool"
77 hRTtitle5 = "Discharged Assemblies by Assembly Type"
78 hRTtitle6 = "Discharged Assemblies by Assembly Class"
79 hRTtitle7 = "Discharged Assemblies by Reactor Type"
80
81 hRTtitle8 = "Discharged Assemblies for Utility: "
82 hRTtitle9 = "Discharged Assemblies for Reactor: "
83 hRTtitle10= "Discharged Assemblies in Storage Pool: "
84 hRTtitle11= "Discharged Assemblies of Assembly Type: "
85 hRTtitle12= "Discharged Assemblies by Assembly Class: "
86 hRTtitle13= "Discharged Assemblies for Reactor Type: "
87
88 ** Projected Report titles
89 pRTtitle1 = "All Projected Assemblies"
90 pRTtitle2 = "Projected Assemblies by Utility"
91 pRTtitle3 = "Projected Assemblies by Reactor"
92 pRTtitle6 = "Projected Assemblies by Assembly Class"
93 pRTtitle7 = "Projected Assemblies by Reactor Type"
94 pRTtitle8 = "Projected Assemblies for Utility: "
95 pRTtitle9 = "Projected Assemblies for Reactor: "
96 pRTtitle12= "Projected Assemblies for Assembly Class: "
97 pRTtitle13 = "Projected Assemblies for Reactor Type: "
98
99 histProj = 0
100 rptNum = 0
101 brkDown = 0
102 subtotal = 0
103 oDevice = '
104
105 reacName = " "
106 utilName = " "
107 poolName = " "
108 assmName = " "
109 rTName = " "
110
111 *** Initialize menu selection values
112
113 hptxt1 = "Historical "
114 hptxt2 = "Projected -- No New Orders Case "
115 **hptxt3 = "Projected -- Upper Reference Case"
116
117 hRptTxt1 = "All Assemblies "
118 hRptTxt2 = "Assemblies by Utility "
119 hRptTxt3 = "Assemblies by Reactor "
120 hRptTxt4 = "Assemblies by Storage Pool "
121 hRptTxt5 = "Assemblies by Assembly Type "
122 hRptTxt6 = "Assemblies by Assembly Class "
123 hRptTxt7 = "Assemblies by Reactor Type(BWR and PWR)"
124 hRptTxt8 = "Assemblies for 1 Utility "
125 hRptTxt9 = "Assemblies for 1 Reactor "
126 hRptTxt10 = "Assemblies for 1 Storage Pool "
127 hRptTxt11 = "Assemblies for 1 Assembly Type "
128 hRptTxt12 = "Assemblies for 1 Assembly Class "
129 hRptTxt13 = "Assemblies for 1 Reactor Type "
130
131 *pRptTxt1 = "All Assemblies "
132 *pRptTxt2 = "Assemblies by Utility "
133 *pRptTxt3 = "Assemblies by Reactor "
134 *pRptTxt4 = "Assemblies by Reactor Type(BWR and PWR)"
135 *pRptTxt5 = "Assemblies for 1 Utility "
136 *pRptTxt6 = "Assemblies for 1 Reactor "
137 *pRptTxt7 = "Assemblies for 1 Reactor Type "
138
139 subTBLnk = "
140 subTTxt1 = "Subtotaled by Storage Pool "
141 subTTxt2 = "Subtotaled by Assembly Type "
142 subTTxt3 = "Totals only through " + lastDate

```

APPENDIX A -- PROGRAM LISTINGS
START.PRG

58

```

143
144 brkDTxt1 = "Subtotaled by discharge year      "
145 brkDTxt2 = "Subtotaled by discharge year and burnup bin"
146 brkDTxt3 = "Totals only through " + lastDate
147 brkPTxt3 = "Totals only      "
148
149 odtxt1 = "Screen      "
150 odtxt2 = "Print      "
151 odtxt3 = "Text File"
152 *odTxt4 = "Worksheet File"
153 oDevice1 = "Screen"
154 oDevice2 = "Print"
155 oDevice3 = "Text File"
156 *oDevice4 = "Worksheet File"
157
158 set color to w/n, n/w
159 clear
160 do qtyintro
161 *call ClipPop with "QTYINTRO"
162 SET COLOR TO 7/0, 0/7
163 wait "" to dummy
164
165 ** do RunRpt
166 @ 0,0 CLEAR
167 call ClipPop with "SLCT"
168
169 rptOK = .t.
170 brkDOK = .t.
171 subTTxt = subTBlink
172 mChoice = 'R'
173 answer = ''
174 getData = .f.
175
176 do Sequence           // Get selections for first report
177
178 newParms = .t.          // Signals if we need to generate the report
                           // for new parameters
179
180 do while (.t.)
181
182     set color to n/w,n/w
183     call ClipPop with "Main"
184     set color to n/w,n/w
185     mChoice = 'R'
186
187     @ 14,26 get mChoice picture 'I'
188     read
189
190     do case
191         case mChoice = 'A'           // Historical or Projected
192             call ClipPop with "COVER"
193             set color to n/w,n/w
194             oldHp = histProj
195         if (debug)
196             @ 21,0 clear
197             @ 21,10 say "histProj: "
198             @ 21,40 say histProj
199             wait "" to dummy
200     endif
201
202     do HpSlt
203         do WriteIt with "H",hpTxt
204         newParms = .t.
205
206     if (debug)
207         @ 21,0 clear
208         @ 21,10 say "HistProj: "
209         @ 21,40 say histProj
210         @ 22,10 say "oldHP: "
211         @ 22,40 say oldHp
212         wait "" to dummy
213     endif

```

APPENDIX A - PROGRAM LISTINGS
START.PRG

59

```

214
215 *** If they changed from historical to projected (or vice versa) check
216 *** That the report and/or break down is OK. If not, call the appropriate
217 *** routine and get a new one.
218
219     hpChange = (oldHp = 1 .and. histProj > 1) .or. (oldHp > 1 .and. ;
220             histProj = 1)
221     if (oldHp <> histProj)
222         do ChkRpt with rptOK
223         if (.not. rptOK)
224             if (histProj = 1)
225                 do HRptS1ct
226             else
227                 do PRptS1ct
228             endif
229         endif
230
231     if (hpChange)
232         do ChkBrkD with brkDOK      // Check the break down
233         if (.not. brkDOK)
234             do BrkDS1ct
235         endif
236         do WriteIt with "B",brkDTxt
237     endif
238 endif
239
240 case mChoice = 'B'           // Data Requested
241     call ClipPop with "COVER"
242     set color to n/w,n/w
243     if (histProj = 1)           // Historical data
244         do HRptS1ct
245     else                         // Projected data
246         do PRptS1ct
247     endif
248
249 *** Check that the break down option is still OK for the new report selected
250
251     do ChkBrkD with brkDOK
252     if (.not. brkDOK)
253         do BrkDS1ct
254         do WriteIt with "B",brkDTxt
255     endif
256     newParms = .t.
257
258 case mChoice = 'C'           // Data broken down by
259     call ClipPop with "COVER"
260     set color to n/w,n/w
261     do brkDS1ct
262     do WriteIt with "B",brkDTxt
263     newParms = .t.
264
265 case mChoice = 'D'           // Output Device
266     call ClipPop with "COVER"
267     set color to n/w,n/w
268     do DevS1ct with 3
269     do WriteIt with "O",oDTxt
270
271 case mChoice = 'S'           // Sequence
272     call ClipPop with "COVER"
273     set color to n/w,n/w
274     do Sequence
275     newParms = .t.
276
277 case mChoice = 'R'           // Run the report
278     set color to w/n
279     if (newParms)
280         @ 18,15 say space(41)    // Blank out error msg
281         @ 18,30 say "Generating Data . . ."
282     endif
283     do RunRpt with gotData
284

```

APPENDIX A - PROGRAM LISTINGS
START.PRG

60

```
285      set color to w/n
286      clear
287      call ClipPop with "SLCT"
288      do Refresh
289      newParms = .f.           && Don't need to re-generate report
290
291      case mChoice = 'X'       && Exit
292          set color to w/n,w/n
293          @ 16,0 clear
294          @ 18,15 say "Exit LWR Quantities Database -- Are you Sure (Y/N)?"
295          @ 18,69 get answer picture '!'
296          read
297          if (answer = 'Y')
298              exit
299          endif
300
301      otherwise
302          set color to w/n,w/n
303          @ 16,0 clear
304          @ 18,15 say "Please choose one of the options listed.
305
306      endcase
307
308 enddo
309
310 return
```

INTERNAL DISTRIBUTION

- | | |
|-----------------------|-----------------------------------|
| 1. R. J. Andermann | 25. W. J. Reich |
| 2. R. C. Ashline | 26. R. Salmon |
| 3. J. M. Begovich | 27. S. N. Storch |
| 4. A. G. Croff | 28. M. G. Stewart |
| 5. R. M. Gove | 29. T. D. Welch |
| 6. V. T. Hinkel | 30. Central Research Library |
| 7. D. S. Joy | 31. Laboratory Records Department |
| 8. E. K. Johnson | 32. Laboratory Records, ORNL RC |
| 9. P. E. Johnson | 33. ORNL Patent Section |
| 10. J. A. Klein | 34. Y-12 Technical Library |
| 11. S. B. Ludwig | |
| 12. A. P. Malinauskas | |
| 13. W. C. McClain | |
| 14. J. W. Nehls | |
| 15 - 24. K. J. Notz | |

EXTERNAL DISTRIBUTION

DOE - Office of Civilian Radioactive Waste Management, Forrestal Building,
1000 Independence Avenue, S. W., Washington, DC 20585

35. A. B. Brownstein
36. W. J. Danker
37. H. J. Hale
38. M. L. Payton
39. E. Svenson

DOE - Energy Information Administration, Washington, DC 20585

40. H. Chou
41. J. A. Disbrow
42. K. Gibbard

DOE - Albuquerque Operations Office, P. O. Box 5400, Albuquerque, NM 87115

- 43. K. Golliher
- 44. D. M. Lund

DOE - Idaho Operations Office, 785 DOE Place, Idaho Falls, ID 83402

- 45. M. Fisher
- 46. M. W. Shupe

DOE - Nevada Operations Office, P. O. Box 98518, Las Vegas, NV 89113

- 47. Carl Gertz
- 48. Ed Wilmot

DOE - Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831

- 49. Office of Assistant Manager for Energy Research

DOE - Richland Operations Office, P. O. Box 550, Richland, WA 99352

- 50. D. C. Langstaff

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94550

- 51. Les Jardine
- 52. Ray Stout

Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352

- 53. G. H. Beeman
- 54. M. E. Cunningham
- 55. A. T. Luksic
- 56. R. W. Walling

Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185

- 57. J. Cashwell
- 58. A. W. Dennis
- 59. E. Ryder

ASG, Inc., 800 Oak Ridge Turnpike, Oak Ridge, TN 37830

- 60. E. A. Lewis
- 61. R. S. Moore

DataPhile, Inc., 9121 Garrison Rd., Knoxville, TN 37919

62. K. E. Jones

David Andress Associates, Inc., 11008 Harriett Lane, Kensington, MD 20895

63. D. Andress

EG&G Idaho, Inc., P. O. Box 1625 Idaho Falls, ID 83415

64. H. Worle

Electric Power Research Institute, P. O. Box 10412, Palo Alto, CA 94303

65. R. W. Lambert

66. O. Ozer

E. R. Johnson Associates, Inc., 10461 White Granite Drive., Suite 204, Oakton, VA 22124

67. N. B. McLeod

68. M. White

General Atomics, P. O. Box 85608, San Diego, CA 92138

69. S. D. Su

70. J. Boshoven

H&R Technical Associates, 575 Oak Ridge Turnpike, Oak Ridge, TN 37830

71. W. R. Rhyne

Science Applications Int. Corp., 800 Oak Ridge Turnpike, Oak Ridge, TN 37830

72. R. Best

73. R. W. Peterson

Science Applications Int. Corp., 101 Convention Center Dr., Las Vegas, NV 89109

74. R. Morrisette

Westinghouse Hanford Operations, P. O. Box 800, Richland, WA 99352

75. R. A. Watrous

Westinghouse Idaho Nuclear Company, P. O. Box 4000, Idaho Falls, ID 83404

76. D. A. Knecht

Roy F. Weston, Inc., 955 L'Enfant Plaza, SW., Eighth Floor, Washington, DC 20024

- 77. M. Conroy
- 78. J. DiNunno

R. F. Weston, 1 Weston Way, West Chester, PA 19380

- 79. R. R. Macdonald

80-89. Office of Scientific and Technical Communication,
P. O. Box 62, Oak Ridge, TN 37831