SWEPP Gamma-Ray Spectrometer System Software Design Description

**D. A. Femec
E. W. Killian**

Published August 1994

**Idaho National Engineering Laboratory
EG&G Idaho, Inc.
Idaho Falls, Idaho 83415-7111**

**Prepared for the
U.S. Department of Energy
Under DOE Idaho Field Office
Contract DE-AC07-761D01570**

MASTER

DISCLAIMER

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, make 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 service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

ABSTRACT

To assist in the characterization of the radiological contents of contact-handled waste containers at the Stored Waste Examination Pilot Plant (SWEPP), the SWEPP Gamma-Ray Spectrometer (SGRS) System has been developed by the Radiation Measurements and Development Unit of the Idaho National Engineering Laboratory. The SGRS system software controls turntable and detector system activities. In addition to determining the concentrations of gamma-ray-emitting radionuclides, this software also calculates attenuation-corrected isotopic mass ratios of specific interest. This document describes the software design for the data acquisition and analysis software associated with the SGRS system.

CONTENTS

1. INTRODUCTION	1
1.1 Purpose	1
1.2 Scope	1
1.3 Definitions, Acronyms, and Abbreviations	1
1.3.1 General	1
1.3.2 Insert Libraries	2
1.4 Overview	2
1.5 Registered Trademarks and Copyrights	2
2. HARDWARE AND SOFTWARE REQUIREMENTS	3
2.1 Hardware	3
2.2 Software	3
2.2.1 Language and Tools	3
2.2.2 Necessary Files for Execution	3
3. GC - DATA ACQUISITION	6
3.1 Overview	6
3.2 Overall Design	6
3.3 Main Routine-, Subroutine-, and Function-Related Specifics	6
3.3.1 GC Main Routine	6
3.3.2 GC Help Screens	8
3.3.3 SLOSYN Subroutine	8
3.3.4 QIOWL Subroutine	8
3.3.5 DIAGLN Subroutine	8

3.3.6	ADCAST Subroutine	9
3.3.7	READ_STATUS Subroutine	9
3.3.8	SET_ERR_MSG Subroutine	9
3.3.9	STOP_COUNTING Subroutine	10
3.3.10	READ_SPECTRUM Subroutine	10
3.3.11	START_COUNTING Subroutine	11
3.3.12	ASTIME Subroutine	11
3.3.13	DVALID Function	11
3.3.14	STOP_DISPLAY Subroutine	11
3.3.15	ENTER_PAR Subroutine	12
3.3.16	GCMENU Subroutine	12
3.3.17	START_DISPLAY Subroutine	13
3.3.18	TH_CALB Subroutine	13
3.3.19	EU_CALB Subroutine	13
3.3.20	PEQ_TH Subroutine	14
3.3.21	QUALITY_UPDATE Subroutine	14
3.3.22	QA_REPORT_SUB Subroutine	15
3.3.23	QA_REMOVE Subroutine	15
3.3.24	QUE_PRINT_DEL Subroutine	16
3.3.25	PEQ_EU Subroutine	16
3.3.26	TYPE_FILE Subroutine	16
3.3.27	SUBMIT Subroutine	17
3.3.28	QUE_PRINT Subroutine	17
4.	GIDXC - ISOTOPIC RATIO CALCULATIONS	18

4.1 Overview	18
4.2 Overall Design	18
4.3 Main Routine-, Subroutine-, and Function-Related Specifics	18
4.3.1 BATCH_GIDXC_ANALYSIS Main Routine	18
4.3.2 CBAT_PRIGIDXC Subroutine	20
4.3.3 DPY_CBAT_GAUSS Subroutine	20
4.3.4 DPY_CBAT_LINFIT Subroutine	20
4.3.5 DPY_CBAT_LINFIT_DP Subroutine	21
4.3.6 FIND_LINE Subroutine	21
4.3.7 MU_LOOK Subroutine	22
4.3.8 READ_ATABLE Subroutine	22
4.3.9 MU_SELF_LOOK Subroutine	22
4.3.10 LSTSQR Subroutine	22
4.3.11 SIGFIG Function	23
4.3.12 BR_UC Function	23
4.3.13 WAVG_O Subroutine	23
4.3.14 DPY_REGION Subroutine	24
4.3.15 BATCH_GIDXC_SUMMARY Main Routine	24
5. REFERENCES	25
APPENDIXES	A-1
Appendix A. User-Readable Isotope, Insert, Limit, and Interference Library Listings	A-1
A.1 Isotope Library GE\$ISOTOPE\$LIB:SWEPP.LIB, Sorted by Energy	A-1

A.2	Isotope Library GE\$ISOTOPE\$LIB:SWEPP.LIB, Sorted by Nuclide	A-7
A.3	Insert Library GE\$LIMIT\$LIB:INSERT.LIB	A-14
A.4	Limit Library GE\$LIMIT\$LIB:SWEPPREQ.LIB	A-14
A.5	Interference Library GE\$INTERF\$LIB:SWEPPIF.LIB	A-16
A.6	Isotope Library GE\$ISOTOPE\$LIB:MASTER.LIB, Sorted by Energy	A-16
A.7	Isotope Library GE\$ISOTOPE\$LIB:MASTER.LIB, Sorted by Nuclide	A-35
A.8	Isotope Library GE\$ISOTOPE\$LIB:ENVLIB.LIB, Sorted by Energy	A-59
A.9	Isotope Library GE\$ISOTOPE\$LIB:ENVLIB.LIB, Sorted by Nuclide	A-64
A.10	Limit Library GE\$LIMIT\$LIB:ENVLIM.LIB	A-71
A.11	Interference Library GE\$INTERF\$LIB:ENVINTLIB.LIB	A-76
A.12	Isotope Library GE\$ISOTOPE\$LIB:EU_CALIBRATION.LIB	A-77
A.13	Isotope Library GE\$ISOTOPE\$LIB:TH_CALIBRATION.LIB	A-77
Appendix B.	GC-Related Source Code and Other Listings	B-1
B.1	GC Main Routine	B-1
B.2	GC Help Screen #1	B-47
B.3	GC Help Screen #2	B-47
B.4	GC Help Screen #3	B-48
B.5	GC Help Screen #4	B-48
B.6	SLOSYN Subroutine	B-49
B.7	QIOWL Subroutine	B-54
B.8	DIAGLN Subroutine	B-55
B.9	ADCAST_GC Subroutine	B-56
B.10	READ_STATUS Subroutine	B-66
B.11	SET_ERR_MSG Subroutine	B-67

B.12	STOP_COUNTING Subroutine	B-68
B.13	READ_SPECTRUM Subroutine	B-70
B.14	START_COUNTING Subroutine	B-75
B.15	ASTIME Subroutine	B-77
B.16	DVALID Function	B-78
B.17	STOP_DISPLAY Subroutine	B-78
B.18	ENTER_PAR Subroutine	B-78
B.19	GCMENU Subroutine	B-81
B.20	START_DISPLAY Subroutine	B-83
B.21	TH_CALB Subroutine	B-84
B.22	EU_CALB Subroutine	B-90
B.23	PEQ_TH Subroutine	B-97
B.24	QUALITY_UPDATE Subroutine	B-101
B.25	QA_REPORT_SUB Subroutine	B-110
B.26	QA_REMOVE Subroutine	B-119
B.27	QUE_PRINT_DEL Subroutine	B-120
B.28	PEQ_EU Subroutine	B-121
B.29	TYPE_FILE Subroutine	B-126
B.30	SUBMIT Subroutine	B-126
B.31	QUE_PRINT Subroutine	B-129
B.32	DCL Command File for Linking GC	B-131
B.33	ADC_STATUS Include File	B-131
B.34	AIM_GSS Include File	B-132
B.35	ANALYSIS_PARAMETER Include File	B-133
B.36	ASTTRM Include File	B-134

B.37	CALIBRATION Include File	B-134
B.38	PULSER_CONFIG Include File	B-134
B.39	QC_QUALITY Include File	B-135
B.40	THOR_QUALITY Include File	B-136
B.41	GC_USER_INITIALIZATION DCL Command File	B-136
Appendix C.	GIDXC-Related Source Code and Other Listings	C-1
C.1	BATCH_GIDXC_ANALYSIS Main Routine	C-1
C.2	CBAT_PRGIDXC Subroutine	C-46
C.3	DPY_CBAT_GAUSS Subroutine	C-51
C.4	DPY_CBAT_LINFIT Subroutine	C-63
C.5	DPY_CBAT_LINFIT_DP Subroutine	C-75
C.6	FIND_LINE Subroutine	C-84
C.7	MU_LOOK Subroutine	C-85
C.8	READ_ATABLE Subroutine	C-87
C.9	MU_SELF_LOOK Subroutine	C-89
C.10	LSTSQR Subroutine	C-91
C.11	SIGFIG Function	C-96
C.12	BR_UC Function	C-100
C.13	WAVG_O Subroutine	C-101
C.14	DPY_REGION Subroutine	C-105
C.15	DCL Command File for Linking BATCH_GIDXC_ANALYSIS	C-124
C.16	BATCH_GIDXC_SUMMARY Main Routine	C-124
C.17	DCL Command File for Linking BATCH_GIDXC_SUMMARY	C-140
C.18	DCL Command File for Executing GIDXC within a Batch Queue	C-140

C.19 DCL Command File for Generating a GIDXC Container Summary	C-141
C.20 AT_TABLE Include File	C-142
C.21 BRANCH Include File	C-142
C.22 FIND_LINE_STRUCT Include File	C-143

FIGURES

Figure 1. Directory structures related to core VAXGAP routines and to GC and GIDXC.	5
Figure 2. Code interrelationships within GC.	7
Figure 3. Code interrelationships within BATCH_GIDXC_ANALYSIS.	19
Figure 4. Code interrelationships within BATCH_GIDXC_SUMMARY.	19

1. INTRODUCTION

Nondestructive analysis systems are in use at the Stored Waste Examination Pilot Plant (SWEPP) to characterize the radiological contents of contact-handled waste containers. Drums and boxes from various waste generators, including Rocky Flats and Argonne National Laboratory West, are currently stored at SWEPP. Before these containers can be shipped to the Waste Isolation Pilot Plant (WIPP), compliance to WIPP's storage, shipping, and disposal requirements must be verified.

This document describes the design of version 1.00 of the Gamma-Ray Data Acquisition (GC) and Isotopic Ratio Analysis (GIDXC) computer programs, the software components of the SWEPP Gamma-Ray Spectrometer (SGRS) System. These data acquisition and analysis programs make extensive use of VAX Gamma-Ray Analysis Program (VAXGAP) version 1 subroutines and functions, which are described in References 1 and 2, for controlling and monitoring the detector systems and for arriving at isotopic concentrations from acquired gamma-ray spectral data. A user's guide is also available [3]. The components to be described herein include the code for the user interface with the data acquisition system, GC, and the code that calculates the isotopic ratio calculations from the analysis results [4], GIDXC; again, the core VAXGAP routines are not described in this document. Further development of the code responsible for isotopic ratio calculations will occur as currently untreated drum and box content codes [5] are encountered. Software and hardware requirements not covered by this document are presented in References 4 and 5; the configuration management plan is to be developed by RWMC/SWEPP personnel. Much of the software testing will take place in conjunction with the SGRS system acceptance and systems operations (SO) tests, given that the data acquisition control functions cannot be adequately tested without the associated detector systems and turntable.

GC and GIDXC are being developed by the staff of the Software Development Section of the Radiation Measurements and Development Unit, with technical direction from L. O. Johnson, L. V. East, and E. B. Nieschmidt.

1.1 Purpose

This document describes the internal and external interfaces for GC and GIDXC, and follows the design description format and content guideline detailed in the IEEE software engineering design standard [6], as appropriate for non-object-oriented programs and the programming and operating system environment. Other purposes of this document include laying the groundwork for further development of GC and GIDXC and for verification and validation conformable to EG&G Idaho, Inc.'s quality requirements [7]. The intended audience includes the developers and validation and verification analysts; users of GC and GIDXC may also find this document of interest in its description of the programming details.

1.2 Scope

This document breaks down the GC and GIDXC programs into their component pieces (or design entities): main routines, subroutines, and functions. The overall structure of the programs is presented, along with the purpose and function of each entity and the entity's code.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 General

Typical and customary definitions, acronyms, and abbreviations for a software design description and software development [8] should be assumed in reading this document. Terminology related to the core VAXGAP routines, and somewhat to gamma-ray spectral analysis in general, follows that used in References 1 and 2, with the addition of insert libraries described below. Further, a general understanding of terminology related to the Digital Equipment Corporation's VMS operating system, DECnet communication protocol, DECwindows Motif graphical user interface, FORTRAN compiler, and GKS (Graphical Kernel System) plotting routines is desirable. Otherwise, this document explicitly defines technical terms as they are presented and avoids unnecessary technical jargon for the sake of the general reader.

1.3.2 Insert Libraries

The proper identification of multiplets in a gamma-ray spectrum is sometimes problematic. When there is insufficient resolution of the separate peaks making up a multiplet, peak-locating routines, such as VAXGAP's SEARCH, may "locate" only a single peak in the region of the multiplet. Subsequent fitting of a gaussian function to the "single" peak yields an unacceptably large peak width and radionuclide tags and calculated concentrations that are incorrect. The analysis of spectra from plutonium-containing samples would suffer from these problems if it were not for the introduction of insert libraries.

The standard flow of peak identification and fitting through VAXGAP is as follows [1]: (1) SEARCH attempts to locate peaks in the spectrum of interest, (2) GAUSS attempts a non-linear fit (i.e., one in which the peak position and width are allowed to vary along with the peak height and parameters related to the spectrum continuum adjacent to the peak) on all possible peaks found by SEARCH, (3) user-specified peak energies of radionuclides that must be analyzed for (from the limit, or required, library) are added to the list of peaks, and (4) LINFIT attempts a linear fit (i.e., one in which the peak position and width are not allowed to vary) on those peaks GAUSS was not able to fit and those peaks added in step (3).

With the advent of insert libraries, user-specified peak energies of known multiplets are inserted into the list of possible peaks generated by SEARCH. GAUSS now knows that a multiplet is to be fit within a certain spectral region, and proceeds accordingly. The results obtained in this way are much more satisfactory, particular in cases where the spectral resolution is not optimal.

1.4 Overview

Section 2 lists the hardware and software requirements for executing the programs GC and GIDXC. Section 3 describes GC, and its component pieces, in detail; Section 4 presents the same for GIDXC. Section 5 lists the appropriate references for this document. User-readable listings of the isotope, insert, limit, and interference libraries mentioned in Section 2.2.2 are presented in Appendix A. Source code listings for GC are presented in Appendix B; those for GIDXC are presented in Appendix C.

1.5 Registered Trademarks and Copyrights

DEC, DECnet, DECwindows, GKS, VAX, and VMS are registered trademarks of Digital Equipment Corporation (DEC). Motif is a registered trademark of Open Systems Foundation. Slo-Syn is a registered trademark of The Superior Electric Company.

2. HARDWARE AND SOFTWARE REQUIREMENTS.

2.1 Hardware

GC and GIDXC are designed to run on a DEC VAX computer (or workstation). At a minimum, a DEC VT330 terminal (or equivalent) is necessary for the user; the GC display routine, DAS [2, 3], requires an X-windowing terminal and a mouse. Hardcopy output and spectral plots are generated on an attached laser printer. Canberra acquisition interface modules (AIMs) are connected to the computer via thin-wire Ethernet and provide the communication link between the computer and the gamma-ray detector systems. Canberra's turntable, equipped with Superior Electric's Slo-Syn model SS2000D6 packaged drive [9] and model SS20001 programmable motion control [10], is connected to the computer's serial port via an RS-232 cable.

2.2 Software

2.2.1 Language and Tools

GC and GIDXC were developed using DEC FORTRAN and GKS and the DECwindows Motif graphical user interface under the VMS operating system. The VMS operating system, along with run-time versions of FORTRAN and GKS, are required for execution of the programs. The core VAXGAP code, along with Canberra's (formerly Nuclear Data's) AIM control library (composed of FORTRAN-callable NMC_ functions), must also be installed.

2.2.2 Necessary Files for Execution

The following isotope, insert, limit (or required), and interference libraries [1, 2, 3], respectively, are required for analyzing spectra: GE\$ISOTOPE\$LIB:SWEPP.LIB, GE\$LIMIT\$LIB:INSERT.LIB and SWEPPREQ.LIB, and GE\$INTERF\$LIB:SWEPPIF.LIB. User-readable listings of the contents of these libraries are found in Appendix A. The following user and batch program communication files are needed within the directory pointed to by the logical SYS\$LOGIN for any given user: A1.INPUT, A1.OUTPUT, A2.INPUT, A2.OUTPUT, A3.INPUT, A3.OUTPUT, A4.INPUT, A4.OUTPUT, GIDXC.INPUT, GIDXC.OUTPUT, SLOSYN.DAT, SLOSYN_SPAWN_IN.DAT, and SLOSYN_SPAWN_OT.DAT. These program communication files, which can be created by executing the DCL^a command program INEL\$EXE:GC_USER_INITIALIZATION [3], are generally empty, unless an error occurs during data processing [3]; in that case, the appropriate .OUTPUT file may contain some information useful for the user, developer, or system administrator in tracking down and correcting the cause of the error.

The installation of VAXGAP on the computer creates a specific directory structure on the disk pointed to by the logical GAP\$DISK. The directory structures for this disk and the system disk, including GC- and GIDXC-related additions, and the logicals^b that point to specific directories within the directory

a. DCL is an abbreviation for DEC's command language within the VMS operating system. The DCL prompt is the prompt the VMS operating system presents to the user after the user logs in.

b. Logicals, within VMS, point to directories or devices, or they contain values. The logical INEL\$EXE, e.g., points to a directory that contains the programs associated with SGS. Some commands entered at the DCL prompt, such as GC, are actually symbols. Symbols, as contrasted to logicals, are generally abbreviations for commands to run programs or for lengthy DCL commands. For example, GC is an

trees, are presented in Figure 1.

abbreviation for the command RUN INEL\$EXE:GC. Symbols are also used within VAXGAP to point to AIMS; A1\$AIM, e.g., contains the Ethernet address for the AIM associated with detectors A1 and A2.

<u>disk</u>	<u>root directories</u>	<u>subdirectories . . .</u>	<u>associated logicals</u>	<u>contents</u>
DKA100: (system-specific)			GAP\$DISK, \$USER	VAXGAP; data, results, and user directories
	DIFFERENCES	CONFIGURED NONCONFIGURED PRECERTIFICATION	DIF\$CON DIF\$NON DIF\$PRE	difference files from core VAXGAP changes from other post-certification changes from other pre-certification changes
	NETWORK	EXE	RML\$NET	network-related control command files
	OPERATOR		-	user files
	VAXGAPCODE	GAP GAP_V2 INEL LIBRARY WIRE	GAP\$EXE GAP\$SRC GAPV2\$EXE GAPV2\$SRC INEL\$EXE INEL\$SRC LIB\$SRC LIBV2\$SRC WIRE\$EXE	core VAXGAP executables source code for the above core VAXGAP version 2 executables source code for the above GC and GIDXC (and non-core VAXGAP) executables source code for the above VAXGAP function object library and source code same for VAXGAP version 2 historical location of some conversion routines
	VAXGAPDATA	LIBRARY RESULTS SLOT SPECTRUM TABLES	GE\$INTERF\$LIB GE\$ISOTOPE\$LIB GE\$LIMIT\$LIB GE\$PD\$LIB BATCH\$RESULTS EFFCRV\$RESULTS PANRATIOS SLOT\$BKG SLOT\$SPC BK\$SAVE EU\$SAVE GN\$SAVE, GE\$SAVE TH\$SAVE AT\$TABLE EFFICIENCY\$TABLES	interference libraries isotope libraries limit libraries parent-daughter correction libraries radionuclide activity results files by spectrum results from efficiency curve determinations isotopic mass ratio results file by container slot-specific spectrum files, QC/calibration data slot-specific background files background spectrum files europium calibration spectrum files sample (i.e., general) spectrum files thorium calibration files attenuation coefficient tables detector efficiency curves
DKA0: (system-specific)			\$SYSTEM, \$DATA	operating system, compilers, . . .
	NDAIMLIB		LIB_CAM	Canberra's AIM control library
	VAXGAPDATA	SPECTRUM BK EU GN TH	BK\$DATA EU\$DATA GN\$DATA, GE\$DATA TH\$DATA	background spectrum files archive europium calibration spectrum files archive sample (i.e., general) spectrum files archive thorium calibration files archive

Figure 1. Directory structures related to core VAXGAP routines and to GC and GIDXC.

3. GC - DATA ACQUISITION

3.1 Overview

GC provides the interface between the user and the detector system and turntable. It also allows the user to perform diagnostic measurements on the detector system and spectral analyses using GIDXC (see Section 4).

3.2 Overall Design

The interrelationships between the main routine, GC, and its subroutines and functions are depicted in Figure 2, with subroutines and functions listed in the order they are first encountered. Various VMS operating system subroutines and functions are used throughout GC and its developer-coded subroutines and functions to handle screen (SMG) and file (LIB) management, logical and symbol translations (LIB), string (STR) and error handling (LIB), batch and print queue submissions (LIB), and port-related queued I/O functions (QIO); these are described in the appropriate VMS manuals and are not included in the figure. Further, these system functions and routines, along with core VAXGAP subroutines and functions and Canberra (formerly Nuclear Data) NMC_ functions, are not explicitly listed in the figure. A double line is drawn to these to indicate both that this is only a general class listings and that members of the class may be accessed by other explicitly listed subroutines and functions.

3.3 Main Routine-, Subroutine-, and Function-Related Specifics

Each routine and function within GC will now be treated in detail, including a functional description (with a reference to the appropriate subsection of Appendix B for the source code listing), the parameters to be passed (expected inputs) and the parameters to be returned (intended outputs), excluding those parameters passed via common blocks, and the interfaces (screens, files, and ports) used. Other files necessary for execution (see Section 2.2.2) and for linking the main routine will also be included. Include files that are not part of the core VAXGAP routines will be listed in alphabetical order at the end of Appendix B.

From hereon the full wording "located in the directory pointed to by the logical ..." will be replaced with "located in ..." for the sake of brevity.

3.3.1 GC Main Routine

3.3.1.1 Functional Description. The main routine initializes screen management, including virtual screens and keyboard, displays the user interface screen, initializes communication with the AIMS and the turntable controller, and responds to commands entered by the user [3]. The source code listing can be found in Appendix B-1.

3.3.1.2 Parameters to be Passed. If started via the BS (bar code start) command (see Reference 4), the USER_MODE logical CONTAINER\$ID will contain the container identification code.

3.3.1.3 Parameters to be Returned. None.

3.3.1.4 Interfaces Used.

Figure 2. Code interrelationships within GC.

Screens: virtual displays on a VT330 (or equivalent) terminal.

Files: files used as channels to the AIMs (and hence to the ADCs), <ADC symbol>.DAT; calibration files, <ADC symbol>.CLB, (located in SLOT\$SPC); and spectrum files, <ADC symbol><date-time stamp>.DGE (located in GE\$SAVE, OLDSWEPP\$SAVE, EU\$SAVE, or TH\$SAVE).

Ports: none.

3.3.2 GC Help Screens

The help screens displayed by GC in response to the user command "HELP;" are listed in Appendixes B-2 through B-5.

3.3.3 SLOSYN Subroutine

3.3.3.1 Functional Description. This subroutine sends commands to the turntable controller via calls to QIOWL (see Section 3.3.4). The source code listing can be found in Appendix B-6.

3.3.3.2 Parameters to be Passed. A logical flag indicating whether communications with the turntable controller are to be initialized during the current call to SLOSYN, and a character string describing the turntable command to be executed.

3.3.3.3 Parameters to be Returned. A logical flag indicating whether communications with the turntable controller are to be initialized during the next call to SLOSYN.

3.3.3.4 Interfaces Used. None (here and hereafter, a sole "None" should be read as "None" to screens, files, and ports).

3.3.4 QIOWL Subroutine

3.3.4.1 Functional Description. This subroutine sends commands to the turntable controller via system routine QIO calls. The source code listing can be found in Appendix B-7.

3.3.4.2 Parameters to be Passed. The logical unit number for the terminal port to which the turntable controller is connected, and a character string containing the command to be sent to the turntable controller.

3.3.4.3 Parameters to be Returned. None.

3.3.4.4 Interfaces Used.

Screens: none.

Files: none.

Ports: Terminal port connected to the turntable controller.

3.3.5 DIAGLN Subroutine

3.3.5.1 Functional Description. This subroutine draws pseudo-diagonal (i.e., staircase-like) lines via screen management calls. The source code listing can be found in Appendix B-8.

3.3.5.2 Parameters to be Passed. A virtual display identifier, and the (row,column) data pairs for the starting point and ending point of the desired diagonal line.

3.3.5.3 Parameters to be Returned. None.

3.3.5.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.6 ADCAST Subroutine

3.3.6.1 Functional Description. This subroutine initializes and maintains, via system service calls, asynchronous system traps (ASTs) for monitoring the status of the AIMs (and their attached ADCs), starting, monitoring, stopping, and processing acquisitions, and appropriately updating virtual displays. The source code listing can be found in Appendix B-9.

3.3.6.2 Parameters to be Passed. None.

3.3.6.3 Parameters to be Returned. None.

3.3.6.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.7 READ_STATUS Subroutine

3.3.7.1 Functional Description. This subroutine uses NMC_ function calls to determine the status of an ADC being monitored by GC. The source code listing can be found in Appendix B-10.

3.3.7.2 Parameters to be Passed. The ordinal number of the ADC of interest within the structure found in the ADC_STATUS include file.

3.3.7.3 Parameters to be Returned. A logical which is set to true if an error occurred while obtaining the status of the specified ADC.

3.3.7.4 Interfaces Used. None.

3.3.8 SET_ERR_MSG Subroutine

3.3.8.1 Functional Description. This subroutine displays new messages, principally error messages, in the appropriate virtual display. The source code listing can be found in Appendix B-11.

3.3.8.2 Parameters to be Passed. A character string containing the message to be displayed.

3.3.8.3 Parameters to be Returned. None.

3.3.8.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.9 STOP_COUNTING Subroutine

3.3.9.1 Functional Description. This subroutine stops an acquisition on an ADC using the appropriate NMC_ function. The source code listing can be found in Appendix B-12.

3.3.9.2 Parameters to be Passed. The ordinal number of the ADC of interest within the structure found in the ADC_STATUS include file.

3.3.9.3 Parameters to be Returned. A logical which is set to true if an error occurred while obtaining the status of the specified ADC.

3.3.9.4 Interfaces Used. None.

3.3.10 READ_SPECTRUM Subroutine

3.3.10.1 Functional Description. This subroutine reads, via the appropriate AIM and NMC_ function, the contents of the ADC into a spectrum file, including identifying header information. The source code listing can be found in Appendix B-13.

3.3.10.2 Parameters to be Passed. The two-character symbol of the ADC of interest, the appropriate sample weight and type, and any comments to be included in the spectrum file.

3.3.10.3 Parameters to be Returned. A logical which is set to true if an error occurred while obtaining the status of the ADC of interest.

3.3.10.4 Interfaces Used.

Screens: none.

Files: file used as channel to the appropriate AIM, <ADC symbol>.DAT (located in SLOT\$SPC); and spectrum file to contain the acquired data and appropriate header information, <ADC symbol><date-time stamp>.DGE (located in GE\$SAVE, OLDSWEPP\$SAVE, EU\$SAVE, or TH\$SAVE, as appropriate).

Ports: none.

3.3.11 START_COUNTING Subroutine

3.3.11.1 Functional Description. This subroutine, after clearing the appropriate AIM, starts an acquisition on an ADC using the appropriate NMC_ function. The source code listing can be found in Appendix B-14.

3.3.11.2 Parameters to be Passed. The ordinal number of the ADC of interest within the structure found in the ADC_STATUS include file.

3.3.11.3 Parameters to be Returned. A logical which is set to true if an error occurred while obtaining the status of the ADC of interest.

3.3.11.4 Interfaces Used. None.

3.3.12 ASTIME Subroutine

3.3.12.1 Functional Description. This subroutine refreshes the time interval for an AST, i.e. how long until ADCAST executes again. The source code listing can be found in Appendix B-15.

3.3.12.2 Parameters to be Passed. A one-character-long character string containing the single-digit number of seconds to which the time interval is to be reset.

3.3.12.3 Parameters to be Returned. None.

3.3.12.4 Interfaces Used. None.

3.3.13 DVALID Function

3.3.13.1 Functional Description. This function returns a value of true if the passed symbol for an ADC is valid. The source code listing can be found at the end of Appendix B-1.

3.3.13.2 Parameters to be Passed. A two-character-long character string containing the ADC symbol to be tested, the identifier for the virtual display where error messages are to be displayed, and the number of valid ADCs.

3.3.13.3 Parameters to be Returned. None.

3.3.13.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.14 STOP_DISPLAY Subroutine

3.3.14.1 Functional Description. This subroutine temporarily suspends screen management

activity (and hence virtual screen and keyboard updating), and then clears the terminal screen, so that non-screen management output can be displayed. The source code listing can be found in Appendix B-17.

3.3.14.2 Parameters to be Passed. None.

3.3.14.3 Parameters to be Returned. None.

3.3.14.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.15 ENTER_PAR Subroutine

3.3.15.1 Functional Description. This subroutine displays the current analysis parameters (library names and analysis option choices) and allows the user to modify them. The source code listing can be found in Appendix B-18.

3.3.15.2 Parameters to be Passed. A two-character-long character string containing the symbol of the ADC for which the analysis parameters are to be changed.

3.3.15.3 Parameters to be Returned. None.

3.3.15.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: analysis parameters file, <ADC symbol>.ANP (located in SLOT\$SPC).

Ports: none.

3.3.16 GCMENU Subroutine

3.3.16.1 Functional Description. This subroutine displays the analysis parameter menu for ENTER_PAR. The source code listing can be found in Appendix B-19.

3.3.16.2 Parameters to be Passed. None.

3.3.16.3 Parameters to be Returned. None.

3.3.16.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.17 START_DISPLAY Subroutine

3.3.17.1 Functional Description. This subroutine restarts screen management activity (and hence virtual screen and keyboard updating) after non-screen management output has been displayed. The source code listing can be found in Appendix B-20.

3.3.17.2 Parameters to be Passed. None.

3.3.17.3 Parameters to be Returned. None.

3.3.17.4 Interfaces Used.

Screens: virtual display on a VT330 (or equivalent) terminal.

Files: none.

Ports: none.

3.3.18 TH_CALB Subroutine

3.3.18.1 Functional Description. This is a subroutine version of the core VAXGAP thorium-228 calibration routine THOR [1, 2], and it performs energy- and peak width-calibrations for detector systems not equipped with pulsers [12, 13]. The source code listing can be found in Appendix B-21.

3.3.18.2 Parameters to be Passed. A character string containing either the spectrum file name or the ADC symbol referring to the spectrum to be analyzed.

3.3.18.3 Parameters to be Returned. A logical which is set to true if an error occurred while the calibration was being attempted.

3.3.18.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: file used as channel to the appropriate AIM, <ADC symbol>.DAT (located in SLOT\$SPC), or spectrum file, <ADC symbol><date-time stamp>.DGE (typically located in TH\$SAVE); calibration results storage file, <ADC symbol>.CLB, (located in SLOT\$SPC); and calibration results output file, TH_CALB.DAT, (located in SLOT\$SPC until printing is completed).

Ports: none.

3.3.19 EU_CALB Subroutine

3.3.19.1 Functional Description. This subroutine is derived from TH_CALB (Section 3.3.18), and it performs europium-152 calibrations for detector systems not equipped with pulsers [12, 13]. The source code listing can be found in Appendix B-22.

3.3.19.2 Parameters to be Passed. A character string containing either the spectrum file

name or the ADC symbol referring to the spectrum to be analyzed.

3.3.19.3 Parameters to be Returned. A logical which is set to true if an error occurred while the calibration was being attempted.

3.3.19.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: file used as channel to the appropriate AIM, <ADC symbol>.DAT (located in SLOT\$SPC), or spectrum file, <ADC symbol><date-time stamp>.DGE (typically located in EU\$SAVE); calibration results storage file, <ADC symbol>.CLB (located in SLOT\$SPC); and calibration results output file, EU_CALB.DAT, (located in SLOT\$SPC until printing is completed).

Ports: none.

3.3.20 PEQ_TH Subroutine

3.3.20.1 Functional Description. This is a subroutine version of the core VAXGAP thorium-228 calibration routine PEQ [1, 2], and it performs energy- and peak width-calibrations for detector systems equipped with pulsers [12, 13]. The source code listing can be found in Appendix B-23.

3.3.20.2 Parameters to be Passed. A character string containing either the spectrum file name or the ADC symbol referring to the spectrum to be analyzed.

3.3.20.3 Parameters to be Returned. A logical which is set to true if an error occurred while the calibration was being attempted.

3.3.20.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: file used as channel to the appropriate AIM, <ADC symbol>.DAT (located in SLOT\$SPC), or spectrum file, <ADC symbol><date-time stamp>.DGE (typically located in TH\$SAVE); temporary spectrum file PEQ.DAT (located in SLOT\$SPC); calibration results storage file, <ADC symbol>.CLB (located in SLOT\$SPC); and a temporary output "spool" file, <process id stamp>.SPL, for printing calibration results.

Ports: none.

3.3.21 QUALITY_UPDATE Subroutine

3.3.21.1 Functional Description. This subroutine performs a spectrum analysis on a calibration spectrum, compares the results to any prior results, and then, if the results are acceptably close to the prior results, stores them in the appropriate quality control data file and calls QA_REPORT_SUB to print a summary of all the results. The source code listing can be found in Appendix B-24.

3.3.21.2 Parameters to be Passed. A character string containing the full path name of a calibration spectrum, a character string containing the full path name of the isotope library to be used in analyzing the spectrum, and the number of the efficiency table to be used.

3.3.21.3 Parameters to be Returned. None.

3.3.21.4 Interfaces Used.

Screens: none.

Files: spectrum file, <ADC symbol><date-time stamp>.DGE (typically located in EU\$SAVE, or TH\$SAVE), isotope library file, <EU or TH>_CALIBRATION.LIB (located in GE\$ISOTOPE\$LIB); and the appropriate quality control data file, <ADC symbol><EU or TH>QC.DAT (located in SLOT\$SPC).

Ports: none.

3.3.22 QA_REPORT_SUB Subroutine

3.3.22.1 Functional Description. This subroutine generates the summary report referred to in the functional description of QUALITY_UPDATE (Section 3.3.21.1). The source code listing can be found in Appendix B-25.

3.3.22.2 Parameters to be Passed. A character string containing the symbol of the ADC of interest, and a character string containing either EU (for a europium-152 summary report) or TH (for a thorium-228 summary report).

3.3.22.3 Parameters to be Returned. None.

3.3.22.4 Interfaces Used.

Screens: none.

Files: appropriate quality control data file, <ADC symbol><EU or TH>QC.DAT (located in SLOT\$SPC); and a temporary output "spool" file, <process id stamp>.SPL, for printing calibration results.

Ports: none.

3.3.23 QA_REMOVE Subroutine

3.3.23.1 Functional Description. This subroutine returns a QA flag value [2] after having calculated a new average and standard deviation for a list of numbers from which a specified number has been removed. The source code listing can be found in Appendix B-26.

3.3.23.2 Parameters to be Passed. the ordinal number of the entry to be removed from the list, the number of entries in the list, and the list of entries as an array.

3.3.23.3 Parameters to be Returned. The new QA flag value [2].

3.3.23.4 Interfaces Used. None.

3.3.24 QUE_PRINT_DEL Subroutine

3.3.24.1 Functional Description. Using the SYS\$PRINT system service, this subroutine submits an output file to a print queue for printing, the file being deleted after the print job is complete. (This subroutine was written to circumvent a resource-consuming error in the handling of LIB\$SPAWN commands in VMS versions 6.0 and 6.1.) The source code listing can be found in Appendix B-27.

3.3.24.2 Parameters to be Passed. A character string containing the full path name of the file to be printed.

3.3.24.3 Parameters to be Returned. None.

3.3.24.4 Interfaces Used. None.

3.3.25 PEQ_EU Subroutine

3.3.25.1 Functional Description. This subroutine is derived from PEQ_TH (Section 3.3.20), and it performs europium-152 energy- and peak width-calibrations for detector systems equipped with pulsers [12, 13]. The source code listing can be found in Appendix B-28.

3.3.25.2 Parameters to be Passed. A character string containing either the spectrum file name or the ADC symbol referring to the spectrum to be analyzed.

3.3.25.3 Parameters to be Returned. A logical which is set to true if an error occurred while the calibration was being attempted.

3.3.25.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: file used as channel to the appropriate AIM, <ADC symbol>.DAT (located in SLOT\$SPC), or spectrum file, <ADC symbol><date-time stamp>.DGE (typically located in EU\$SAVE); temporary spectrum file PEQ.DAT (located in SLOT\$SPC); calibration results storage file, <ADC symbol>.CLB (located in SLOT\$SPC); and a temporary output "spool" file, <process id stamp>.SPL, for printing calibration results.

Ports: none.

3.3.26 TYPE_FILE Subroutine

3.3.26.1 Functional Description. This subroutine reads the contents of a file, one line at a time, and displays the lines on a VT330 (or equivalent) terminal. (This subroutine was written to circumvent a resource-consuming error in the handling of LIB\$SPAWN commands in VMS versions 6.0 and 6.1.) The source code listing can be found in Appendix B-29.

3.3.26.2 Parameters to be Passed. A character string containing the full path name of the file to be typed.

3.3.26.3 Parameters to be Returned. None.

3.3.26.4 Interfaces Used.

Screens: VT330 (or equivalent) terminal.

Files: file to be printed.

Ports: none.

3.3.27 SUBMIT Subroutine

3.3.27.1 Functional Description. Using the SYSSNDJBCW system service, this subroutine submits a BATCH_GIDXC_ANALYSIS job to a batch queue for execution. (This subroutine was written to circumvent a resource-consuming error in the handling of LIB\$SPAWN commands in VMS versions 6.0 and 6.1.) The source code listing can be found in Appendix B-30.

3.3.27.2 Parameters to be Passed. Six character strings, the first two containing either Y or N, the last four containing either the file name of a spectrum to be analyzed or a space.

3.3.27.3 Parameters to be Returned. None.

3.3.27.4 Interfaces Used. None.

3.3.28 QUE_PRINT Subroutine

3.3.28.1 Functional Description. Using the SYS\$PRINT system service, this subroutine submits an output file to a print queue for printing, the file being retained after the print job is complete. (This subroutine was written to circumvent a resource-consuming error in the handling of LIB\$SPAWN commands in VMS versions 6.0 and 6.1.) The source code listing can be found in Appendix B-31.

3.3.28.2 Parameters to be Passed. A character string containing the full path name of the file to be printed.

3.3.28.3 Parameters to be Returned. None.

3.3.28.4 Interfaces Used. None.

4. GIDXC - ISOTOPIC RATIO CALCULATIONS

4.1 Overview

GIDXC performs the isotopic ratio calculations on a single gamma-ray spectrum or on a set of spectra from a single container measurement. In the latter case, a summary report is generated, along with a file containing certain isotopic ratios for use by the SWEPP Assay System (SAS) [11]. GIDXC is implemented here as a batch routine, and is composed of two main routines: BATCH_GIDXC_ANALYSIS performs the spectrum analysis, and BATCH_GIDXC_SUMMARY generates a summary report if appropriate.

4.2 Overall Design

The interrelationships between the main routines, BATCH_GIDXC_ANALYSIS and BATCH_GIDXC_SUMMARY, and their subroutines and functions are depicted in Figures 3 and 4, respectively, with subroutines and functions listed in the order they are first encountered. Various VMS operating system subroutines and functions are used throughout GIDXC and its developer-coded subroutines and functions to handle screen (SMG) and file (LIB) management, logical and symbol translations (LIB), string (STR) and error handling (LIB), and batch and print queue submissions (LIB); these are described in the appropriate VMS manuals and are not included in the figures. Further, these system functions and routines, along with core VAXGAP subroutines and functions, are not explicitly listed in the figures. A double line is drawn to these to indicate both that this is only a general class listings and that members of the class may be accessed by other explicitly listed subroutines and functions.

4.3 Main Routine-, Subroutine-, and Function-Related Specifics

Each routine and function within GIDXC will now be treated in detail, including a functional description (with a reference to the appropriate subsection of Appendix B for the source code listing), the parameters to be passed (expected inputs) and the parameters to be returned (intended outputs), excluding those parameters passed via common blocks, and the interfaces (screens, files, and ports) used. Other files necessary for execution (see Section 2.2.2) and for linking the main routine will also be included. Include files that are not part of the core VAXGAP routines, and that are not found in Appendix B, will be listed in alphabetical order at the end of Appendix C.

4.3.1 BATCH_GIDXC_ANALYSIS Main Routine

4.3.1.1 Functional Description. This main routine performs a VAXGAP analysis on the specified spectrum [1], calculates and applies the appropriate matrix corrections and evaluates the calculable isotopic ratios [4, 5], and generates appropriate output (see Reference 4 for a sample output). The source code listing can be found in Appendix C-1.

4.3.1.2 Parameters to be Passed. The name of the file containing the spectrum to be analyzed (passed via USER_MODE logical A1\$SPECTRUM\$ID), and a flag indicating whether printed output should be generated (passed as IF\$PRINT).

4.3.1.3 Parameters to be Returned. None.

4.3.1.4 Interfaces Used.

Figure 3. Code interrelationships within BATCH_GIDXC_ANALYSIS.

Figure 4. Code interrelationships within BATCH_GIDXC_SUMMARY.

Screens: none.

Files: analysis parameters file, <ADC symbol>.ANP (located in SLOT\$SPC); isotope, insert, limit, interference, and branching ratio uncertainty library files (located in GE\$ISOTOPE\$LIB, GE\$LIMIT\$LIB, GE\$LIMIT\$LIB, GE\$INTERF\$LIB, and GE\$ISOTOPE\$LIB, respectively); sample and background spectrum files, <ADC symbol><date-time stamp>.DGE (typically located in GN\$SAVE and BK\$SAVE, respectively); temporary output file GIDXC_TEMP.OUTPUT; and a results file, <ADC symbol><date-time stamp>.RES (located in BATCH\$RESULTS).

Ports: none.

4.3.2 CBAT_PRIGIDXC Subroutine

4.3.2.1 Functional Description. This subroutine generates the standard VAXGAP analysis results report. The source code listing can be found in Appendix C-2.

4.3.2.2 Parameters to be Passed. None.

4.3.2.3 Parameters to be Returned. None.

4.3.2.4 Interfaces Used.

Screens: none.

Files: temporary output file GIDXC_TEMP.OUTPUT.

Ports: none.

4.3.3 DPY_CBAT_GAUSS Subroutine

4.3.3.1 Functional Description. This subroutine is derived from the VAXGAP subroutine CBAT_GAUSS [1, 2], which performs fits to peaks found in a spectrum; it has been modified to retain all parameters related to fitting peaks found in the appropriate spectrum for purposes of generating a truly representative XTP plot. The source code listing can be found in Appendix C-3.

4.3.3.2 Parameters to be Passed. None.

4.3.3.3 Parameters to be Returned. None.

4.3.3.4 Interfaces Used.

Screens: none.

Files: temporary output file GIDXC_TEMP.OUTPUT.

Ports: none.

4.3.4 DPY_CBAT_LINFIT Subroutine

4.3.4.1 Functional Description. This subroutine is derived from the VAXGAP subroutine

CBAT_LINFIT [1, 2], which performs a linear fit to a single peak, modified as noted in Section 4.3.3.1. The source code listing can be found in Appendix C-4.

4.3.4.2 Parameters to be Passed. None.

4.3.4.3 Parameters to be Returned. None.

4.3.4.4 Interfaces Used.

Screens: none.

Files: temporary output file GIDXC_TEMP.OUTPUT.

Ports: none.

4.3.5 DPY_CBAT_LINFIT_DP Subroutine

4.3.5.1 Functional Description. This subroutine is derived from the VAXGAP subroutine CBAT_LINFIT [1, 2], which performs a linear fit to a doublet (or "interfered") peak, modified as noted in Section 4.3.3.1. The source code listing can be found in Appendix C-5.

4.3.5.2 Parameters to be Passed. the positions and peaks of the two members of the doublet, the spectral range over which the fit is to be performed, and an index pointing to the "limit" peak (i.e., not the "interfered" peak) within the peak list array.

4.3.5.3 Parameters to be Returned. None.

4.3.5.4 Interfaces Used.

Screens: none.

Files: temporary output file GIDXC_TEMP.OUTPUT.

Ports: none.

4.3.6 FIND_LINE Subroutine

4.3.6.1 Functional Description. This subroutine searches for the specified radionuclide's spectral line within the peak list array, and, if it finds it, loads the structure found in the FIND_LINE_STRUCT include file (see Appendix C.22). The source code listing can be found in Appendix C-6.

4.3.6.2 Parameters to be Passed. The atomic number, atomic mass number, and elemental symbol for the radionuclide of interest, and the energy of the particular spectral line of interest for that radionuclide.

4.3.6.3 Parameters to be Returned. None.

4.3.6.4 Interfaces Used. None.

4.3.7 MU_LOOK Subroutine

4.3.7.1 Functional Description. This subroutine performs a "look up" operation, obtaining the attenuation factor at a specified energy for the sample's matrix (i.e., content code [4, 5]) from the table read in by READ_ATABLE (see Section 4.3.8). The source code listing can be found in Appendix C-7.

4.3.7.2 Parameters to be Passed. The energy of interest.

4.3.7.3 Parameters to be Returned. The matrix-attenuation factor at the energy of interest.

4.3.7.4 Interfaces Used. None.

4.3.8 READ_ATABLE Subroutine

4.3.8.1 Functional Description. This subroutine reads the attenuation factor table (see Section 4.3.7) for the specified content code. The source code listing can be found in Appendix C-8.

4.3.8.2 Parameters to be Passed. None.

4.3.8.3 Parameters to be Returned. None.

4.3.8.4 Interfaces Used.

Screens: none.

Files: attenuation factor table, <table identifier>.TBL (located in AT\$TABLE).

Ports: none.

4.3.9 MU_SELF_LOOK Subroutine

4.3.9.1 Functional Description. This subroutine performs a "look up" operation, obtaining the self-attenuation factor at a specified energy for the radionuclide itself, akin to MU_LOOK (see Section 4.3.7). The source code listing can be found in Appendix C-9.

4.3.9.2 Parameters to be Passed. The energy of interest.

4.3.9.3 Parameters to be Returned. The self-attenuation factor at the energy of interest.

4.3.9.4 Interfaces Used. None.

4.3.10 LSTSQR Subroutine

4.3.10.1 Functional Description. This subroutine performs a weighted, linear-least-squares regression analysis of the supplied data [14], and is derived from the application-specific VAXGAP routine POLYFIT [1, 2]. The source code listing can be found in Appendix C-10.

4.3.10.2 Parameters to be Passed. Arrays containing independent and dependent data values and the one-sigma uncertainties (i.e., standard deviation) in the respective dependent data values, the number of data sets, and the number of linear coefficients to be returned (and hence the number of

independent values associated with each dependent value).

4.3.10.3 Parameters to be Returned. Arrays containing the calculated linear coefficients and their one-sigma uncertainties, and the uncertainty between a observed dependent value and its predicted value.

4.3.10.4 Interfaces Used. None.

4.3.11 SIGFIG Function

4.3.11.1 Functional Description. This function returns a character string of the form (<scaled value> +/- <scaled one-sigma uncertainty>)E<scaling exponential>, with the proper number of significant figures. The source code listing can be found in Appendix C-11.

4.3.11.2 Parameters to be Passed. An unscaled value and its uncertainty.

4.3.11.3 Parameters to be Returned. A character string containing the corresponding representation described above.

4.3.11.4 Interfaces Used. None.

4.3.12 BR_UC Function

4.3.12.1 Functional Description. This function returns the uncertainty in the branching ratio for the specified radionuclide's spectral line. The source code listing can be found in Appendix C-12.

4.3.12.2 Parameters to be Passed. The atomic number, atomic mass number, and elemental symbol for the radionuclide of interest, and the energy of the particular spectral line of interest for that radionuclide.

4.3.12.3 Parameters to be Returned. The uncertainty in the appropriate branching ratio.

4.3.12.4 Interfaces Used.

Screens: none.

Files: branching ratio uncertainty library, GE\$ISOTOPE\$LIB:SWEPP.BR_UC.

Ports: none.

4.3.13 WAVG_O Subroutine

4.3.13.1 Functional Description. This subroutine calculates a unweighted and weighted means for a set of values, having successively applied outlier testing, as described elsewhere, as needed and if requested [1]. The source code listing can be found in Appendix C-13.

4.3.13.2 Parameters to be Passed. Arrays containing the values and their associated uncertainties, the number of values, and a flag indicating whether outlier testing should be performed.

4.3.13.3 Parameters to be Returned. The unweighted mean and its associated uncertainty,

the unweighted mean and its associated uncertainty, the reduced chi-squared value, and an array flagging which values from the input list were used in the final mean and reduced chi-squared value calculations.

4.3.13.4 Interfaces Used. None.

4.3.14 DPY_REGION Subroutine

4.3.14.1 Functional Description. This subroutine is derived from the VAXGAP subroutine XTP_REGION [1, 2], modified to make use of those peak fit parameters retained via prior DPY_ subroutines. It generates a plot of the specified spectral energy range, displaying the fits used for each peak displayed. The source code listing can be found in Appendix C-14.

4.3.14.2 Parameters to be Passed. The energy range to be plotted, and, via USER_MODE logical XTP\$QUE, the laser printer queue at which the plot is to be generated.

4.3.14.3 Parameters to be Returned. None.

4.3.14.4 Interfaces Used.

Screens: none.

Files: GKS plot device file (which is submitted to the specified laser printer queue upon the plot being completed).

Ports: none.

4.3.15 BATCH_GIDXC_SUMMARY Main Routine

4.3.15.1 Functional Description. This routine generates a summary report for a set of measurements performed on a single container (see Reference 4 for a sample output). The source code listing can be found in Appendix C-16.

4.3.15.2 Parameters to be Passed. The names of the files containing the spectra to be included in the summary report (passed via USER_MODE logicals A<n>\$SPECTRUM\$ID, n ranging from one to a maximum of six).

4.3.15.3 Parameters to be Returned. None.

4.3.15.4 Interfaces Used.

Screens: none.

Files: results files, <ADC symbol><date-time stamp>.RES, for the specified spectra (located in BATCH\$RESULTS); a temporary output "spool" file, <process id stamp>.SPL, for printing the summary; and a composite ratio results file, <container identifier>.GAM (located in PAN\$RATIOS), for use by SAS [11].

Ports: none.

5. REFERENCES

1. E. W. Killian and J. K. Hartwell, *VAXGAP: A Code for the Routine Analysis of Gamma-Ray Pulse-Height Spectra on a VAX Computer*, EGG-2533, May 1988.
2. E. W. Killian and D. A. Femec, *Operator's Guide for VAXGAP, a Gamma-Ray Spectrum Analysis Package*, EGG-2672, August 1992.
3. D. A. Femec, *SWEPP Gamma-Ray Spectrometer System Software User's Guide*, EGG-NRE-11452, August 1994.
4. W. Serrano, *Relative Isotopic Mass Ratio Gamma Measurements*, RWMC-EDF-658, June 28, 1994.
5. W. Serrano, *Drum Matrix Mass Attenuation Coefficients*, RWMC-EDF-722, March 14, 1994.
6. ANSI/IEEE Std. 1016-1987, "IEEE Recommended Practice for Software Design Descriptions," Institute of Electrical and Electronic Engineers, Inc.
7. EG&G Idaho, Inc., *Quality Manual*, December 7, 1992.
8. ANSI/IEEE Std. 729-1983, "IEEE Standard Glossary of Software Engineering Terminology," Institute of Electrical and Electronic Engineers, Inc.
9. "Installation Instructions for Slo-Syn SS2000D6 Packaged Drive," The Superior Electric Company, 1992.
10. "Installation Instructions for Slo-Syn SS20001 Programmable Motion Control," The Superior Electric Company, Revision B.
11. S. D. Matthews and E. S. Marwil, *SWEPP Assay System Software Design Description*, EGG-RAAM-10424, September 1993 (and documents referenced therein).
12. J. K. Hartwell and S. G. Goodwin, *IEEE Trans. Nucl. Sci.*, NS-36, February 1989, p. 801.
13. L. O. Johnson, E. W. Killian, R. G. Helmer, and R. A. Coates, *IEEE Trans. Nucl. Sci.*, NS-28, February 1981, p. 638.
14. M. G. Natrella, "Experimental Statistics," National Bureau of Standards Handbook 91, August 1, 1963.

APPENDIXES

Appendix A. User-Readable Isotope, Insert, Limit, and Interference Library Listings

The following are user-readable listings of the SWEPP isotope, insert, limit (or required), and interference libraries used by GIDXC, along with additional isotope (MASTER and ENVLIB), limit (ENVLIM), and interference libraries (ENVINTLIB) available through both the DAS option within GC and other core VAXGAP analysis routines and those isotope libraries (EU_- and TH_CALIBRATION) used in detector calibration.

A.1 Isotope Library GE\$ISOTOPE\$LIB:SWEPP.LIB, Sorted by Energy

This listing was generated using the VAXGAP PRLIB command [2].

SWEPP

PAGE 1

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
42.8167	96	244	CM	18.1000 Y	0.25E-01	1
57.8137	92	232	U	69.8000 Y	0.20E+00	1
99.8530	94	238	PU	87.7000 Y	0.74E-02	2
100.1065	73	182	TA	115.0000 D	0.14E+02	2
102.9516	95	241	AM	432.7000 Y	0.20E-01	5
103.6529	94	241	PU	14.4000 Y	0.10E-03	2
104.2340	94	240	PU	6.56E+03 Y	0.71E-02	2
105.3129	63	155	EU	4.8500 Y	0.22E+02	1
109.1784	92	235	U	7.04E+08 Y	0.15E+01	3
113.6723	73	182	TA	115.0000 D	0.19E+01	3
118.9680	92	233	U	1.59E+05 Y	0.41E-02	3
120.8160	92	233	U	1.59E+05 Y	0.33E-02	3
121.7824	63	152	EU	13.5100 Y	0.28E+02	2
122.3106	91	231	PADA	3.28E+04 Y	0.12E+01	3
122.3107	89	227	ACDA	21.7700 Y	0.12E+01	3
123.1001	63	154	EU	8.5880 Y	0.40E+02	2
125.2100	94	239	PU	2.41E+04 Y	0.71E-04	4
125.2880	95	241	AM	432.7000 Y	0.41E-02	5
129.0348	92	232	U	69.8000 Y	0.68E-01	2
129.2960	94	239	PU	2.41E+04 Y	0.63E-02	2
133.5386	58	144	CE	285.0000 D	0.11E+02	1
133.5387	58	144	CEPR	285.0000 D	0.11E+02	2
135.3284	92	233	U	1.59E+05 Y	0.23E-02	3
143.7680	92	235	U	7.04E+08 Y	0.11E+02	2
144.1835	91	231	PADA	3.28E+04 Y	0.32E+01	2
144.2010	94	239	PU	2.41E+04 Y	0.30E-03	3
146.0940	94	239	PU	2.41E+04 Y	0.12E-03	4
146.3450	92	233	U	1.59E+05 Y	0.66E-02	2
146.5495	95	241	AM	432.7000 Y	0.46E-03	1
148.5396	94	241	PU	14.4000 Y	0.19E-03	1
152.4308	73	182	TA	115.0000 D	0.72E+01	2
152.7200	94	238	PU	87.7000 Y	0.94E-03	1
154.1766	91	231	PADA	3.28E+04 Y	0.55E+01	2
154.1767	89	227	ACDA	21.7700 Y	0.55E+01	2

156.3874	73 182 TA	115.0000 D	0.27E+01	3
159.9276	94 241 PU	14.4000 Y	0.66E-05	3
160.0674	94 239 PU	2.41E+04 Y	0.50E-05	5
160.3080	94 240 PU	6.56E+03 Y	0.40E-03	1
161.4820	94 239 PU	2.41E+04 Y	0.12E-03	4
161.6994	95 241 AM	432.7000 Y	0.90E-05	6
163.3570	92 235 U	7.04E+08 Y	0.51E+01	3
164.5220	92 233 U	1.59E+05 Y	0.62E-02	2
164.5925	94 241 PUDA	14.4000 Y	0.19E+01	3
165.9215	95 241 AM	432.7000 Y	0.24E-04	6
169.5593	95 241 AM	432.7000 Y	0.17E-03	6
176.3161	51 125 SB	2.7590 Y	0.69E+01	2
179.3948	73 182 TA	115.0000 D	0.31E+01	3
185.7220	92 235 U	7.04E+08 Y	0.57E+02	1
186.1109	90 230 THDA	7.54E+04 Y	0.34E+01	3
198.3530	73 182 TA	115.0000 D	0.15E+01	3
200.9572	94 238 PU	87.7000 Y	0.41E-05	5
202.1060	92 235 U	7.04E+08 Y	0.11E+01	3

PAGE 2

ENERGY	Z A NUCLIDE	HALF LIFE	BR	I-CODE
203.5500	94 239 PU	2.41E+04 Y	0.57E-03	3
203.8694	95 241 AM	432.7000 Y	0.29E-05	6
205.3180	92 235 U	7.04E+08 Y	0.50E+01	3
208.0077	95 241 AM	432.7000 Y	0.79E-03	6
208.0078	94 241 PUDA	14.4000 Y	0.22E+02	2
208.1710	92 233 U	1.59E+05 Y	0.23E-02	3
209.3852	90 232 THDA	1.40E+10 Y	0.41E+01	2
209.3853	88 228 RAAC	5.7500 Y	0.41E+01	3
210.5809	91 231 PADA	3.28E+04 Y	0.11E+01	3
210.5810	89 227 ACDA	21.7700 Y	0.11E+01	3
222.1099	73 182 TA	115.0000 D	0.76E+01	2
229.3220	73 182 TA	115.0000 D	0.36E+01	3
235.9695	91 231 PADA	3.28E+04 Y	0.11E+02	2
235.9696	89 227 ACDA	21.7700 Y	0.11E+02	2
238.6320	90 228 THDA	1.9130 Y	0.44E+02	2
240.9870	90 228 THDA	1.9130 Y	0.41E+01	3
241.9192	88 226 RADA	1.60E+03 Y	0.74E+01	2
241.9193	92 234 UDA	2.46E+05 Y	0.74E+01	2
241.9194	90 230 THDA	7.54E+04 Y	0.74E+01	2
244.6989	63 152 EU	13.5100 Y	0.75E+01	2
245.3450	92 233 U	1.59E+05 Y	0.36E-02	3
247.9678	63 154 EU	8.5880 Y	0.66E+01	2
256.2410	91 231 PADA	3.28E+04 Y	0.67E+01	2
256.2411	89 227 ACDA	21.7700 Y	0.67E+01	2
258.1807	92 238 UPA	4.47E+09 Y	0.81E-01	2
264.0755	73 182 TA	115.0000 D	0.36E+01	3
267.5441	94 241 PUDA	14.4000 Y	0.73E+00	1
269.3929	91 231 PADA	3.28E+04 Y	0.13E+02	1
269.3930	89 227 ACDA	21.7700 Y	0.13E+02	1
270.2561	90 232 THDA	1.40E+10 Y	0.38E+01	2
270.2562	88 228 RAAC	5.7500 Y	0.38E+01	3
277.2830	90 228 THDA	1.9130 Y	0.23E+01	3
283.6729	91 231 PADA	3.28E+04 Y	0.16E+01	2
286.1099	91 231 PADA	3.28E+04 Y	0.16E+01	2
286.1100	89 227 ACDA	21.7700 Y	0.16E+01	2
291.3540	92 233 U	1.59E+05 Y	0.54E-02	3
295.0908	88 226 RADA	1.60E+03 Y	0.19E+02	2
295.0909	92 234 UDA	2.46E+05 Y	0.19E+02	2
295.0910	90 230 THDA	7.54E+04 Y	0.19E+02	2
299.9681	89 227 ACDA	21.7700 Y	0.21E+01	2

300.0229	91	231	PADA	3.28E+04	Y	0.45E+01	2
300.0336	90	228	THDA	1.9130	Y	0.33E+01	3
300.1750	93	237	NPPA	2.14E+06	Y	0.66E+01	2
302.6690	91	231	PADA	3.28E+04	Y	0.25E+01	2
304.5108	91	231	PADA	3.28E+04	Y	0.11E+01	3
304.5109	89	227	ACDA	21.7700	Y	0.11E+01	3
312.0117	93	237	NPPA	2.14E+06	Y	0.39E+02	1
317.1327	92	233	U	1.59E+05	Y	0.78E-02	1
320.5410	92	233	U	1.59E+05	Y	0.29E-02	3
322.5410	95	241	AM	432.7000	Y	0.15E-03	6
323.8752	91	231	PADA	3.28E+04	Y	0.39E+01	2
323.8753	89	227	ACDA	21.7700	Y	0.39E+01	2

PAGE 3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
328.0698	90	232	THDA	1.40E+10	Y	0.35E+01	2
328.0699	88	228	RAAC	5.7500	Y	0.35E+01	3
329.8370	89	227	ACDA	21.7700	Y	0.27E+01	2
329.9137	91	231	PADA	3.28E+04	Y	0.41E+01	2
332.3608	95	241	AM	432.7000	Y	0.15E-03	6
332.3609	94	241	PUDA	14.4000	Y	0.12E+01	3
332.8450	94	239	PU	2.41E+04	Y	0.49E-03	3
335.4005	95	241	AM	432.7000	Y	0.49E-03	6
338.2850	91	231	PADA	3.28E+04	Y	0.27E+01	2
338.2851	89	227	ACDA	21.7700	Y	0.28E+01	2
338.4199	90	232	THDA	1.40E+10	Y	0.12E+02	2
338.4200	88	228	RAAC	5.7500	Y	0.12E+02	2
340.5896	93	237	NPPA	2.14E+06	Y	0.45E+01	2
344.2811	63	152	EU	13.5100	Y	0.27E+02	2
345.0130	94	239	PU	2.41E+04	Y	0.56E-03	3
350.0907	91	231	PADA	3.28E+04	Y	0.13E+02	2
350.0908	89	227	ACDA	21.7700	Y	0.13E+02	2
351.8664	88	226	RADA	1.60E+03	Y	0.36E+02	2
351.8665	92	234	UDA	2.46E+05	Y	0.36E+02	2
351.8666	90	230	THDA	7.54E+04	Y	0.36E+02	2
368.5083	94	239	PU	2.41E+04	Y	0.90E-04	4
368.5925	95	241	AM	432.7000	Y	0.22E-03	6
375.0540	94	239	PU	2.41E+04	Y	0.16E-02	3
376.5762	95	241	AM	432.7000	Y	0.14E-03	6
380.1910	94	239	PU	2.41E+04	Y	0.30E-03	3
380.4155	51	125	SB	2.7590	Y	0.15E+01	3
382.6980	94	239	PU	2.41E+04	Y	0.26E-03	3
392.9140	94	239	PU	2.41E+04	Y	0.55E-03	4
398.6635	93	237	NPPA	2.14E+06	Y	0.14E+01	3
409.6246	90	232	THDA	1.40E+10	Y	0.22E+01	2
409.6247	88	228	RAAC	5.7500	Y	0.22E+01	3
411.1260	63	152	EU	13.5100	Y	0.22E+01	3
413.7130	94	239	PU	2.41E+04	Y	0.15E-02	1
415.9324	93	237	NPPA	2.14E+06	Y	0.17E+01	3
422.5980	94	239	PU	2.41E+04	Y	0.12E-03	4
427.8752	51	125	SB	2.7590	Y	0.29E+02	1
433.9360	47	108	AGM	127.0000	Y	0.90E+02	2
443.9034	63	152	EU	13.5100	Y	0.31E+01	3
444.9428	91	231	PADA	3.28E+04	Y	0.13E+01	3
444.9429	89	227	ACDA	21.7700	Y	0.13E+01	3
446.8110	47	110	AGM	249.7600	D	0.37E+01	3
451.4810	94	239	PU	2.41E+04	Y	0.19E-03	4
463.1049	90	232	THDA	1.40E+10	Y	0.46E+01	2
463.1050	88	228	RAAC	5.7500	Y	0.46E+01	3
463.3670	51	125	SB	2.7590	Y	0.10E+02	2
475.3573	55	134	CS	2.0652	Y	0.15E+01	3

511.8647	45	106	RH	29.80000	S	0.21E+02	2
562.6522	90	232	THDA	1.40E+10	Y	0.10E+01	3
562.6523	88	228	RAAC	5.7500	Y	0.10E+01	3
563.2370	55	134	CS	2.0652	Y	0.84E+01	3
569.3207	55	134	CS	2.0652	Y	0.15E+02	2
581.5247	90	232	THDA	1.40E+10	Y	0.15E+01	3

PAGE 4

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
581.5248	88	228	RAAC	5.7500 Y	0.15E+01	3
583.1910	90	228	THDA	1.9130 Y	0.31E+02	2
591.8112	63	154	EU	8.5880 Y	0.48E+01	2
600.5002	51	125	SB	2.7590 Y	0.18E+02	2
604.7101	55	134	CS	2.0652 Y	0.98E+02	2
606.6332	51	125	SB	2.7590 Y	0.50E+01	2
609.3111	88	226	RADA	1.60E+03 Y	0.46E+02	1
609.3112	92	234	UDA	2.46E+05 Y	0.46E+02	1
609.3113	90	230	THDA	7.54E+04 Y	0.46E+02	1
614.2810	47	108	AGM	127.0000 Y	0.90E+02	1
616.1807	45	106	RH	29.80000 S	0.74E+00	2
620.3600	47	110	AGM	249.7600 D	0.28E+01	3
621.9170	45	106	RH	29.80000 S	0.98E+01	1
635.8897	51	125	SB	2.7590 Y	0.11E+02	2
657.7620	47	110	AGM	249.7600 D	0.95E+02	2
661.6600	55	137	CS	30.2500 Y	0.85E+02	1
662.4258	95	241	AM	432.7000 Y	0.36E-03	6
665.4420	88	226	RADA	1.60E+03 Y	0.16E+01	3
665.4421	92	234	UDA	2.46E+05 Y	0.16E+01	3
665.4422	90	230	THDA	7.54E+04 Y	0.16E+01	3
671.3815	51	125	SB	2.7590 Y	0.18E+01	3
677.6230	47	110	AGM	249.7600 D	0.10E+02	2
687.0150	47	110	AGM	249.7600 D	0.64E+01	3
692.4855	63	154	EU	8.5880 Y	0.17E+01	3
696.5100	58	144	CEPR	285.0000 D	0.13E+01	2
702.6450	41	94	NB	2.00E+04 Y	0.10E+03	2
706.6820	47	110	AGM	249.7600 D	0.16E+02	2
722.6994	95	241	AM	432.7000 Y	0.13E-03	6
722.9290	47	108	AGM	127.0000 Y	0.91E+02	2
723.3563	63	154	EU	8.5880 Y	0.20E+02	2
724.1990	40	95	ZR	63.9800 D	0.44E+02	2
724.1991	40	95	ZRNB	63.9800 D	0.44E+02	2
727.2466	90	228	THDA	1.9130 Y	0.67E+01	2
742.8172	94	238	PU	87.7000 Y	0.78E-05	5
744.2770	47	110	AGM	249.7600 D	0.47E+01	3
755.2773	90	232	THDA	1.40E+10 Y	0.13E+01	3
755.2774	88	228	RAAC	5.7500 Y	0.13E+01	3
756.7290	40	95	ZR	63.9800 D	0.55E+02	1
756.7291	40	95	ZRNB	63.9800 D	0.55E+02	2
756.8083	63	154	EU	8.5880 Y	0.43E+01	2
763.9440	47	110	AGM	249.7600 D	0.22E+02	2
765.8070	40	95	ZRNB	63.9800 D	0.22E+03	1
766.4124	94	238	PU	87.7000 Y	0.33E-04	5
766.4125	92	238	UPA	4.47E+09 Y	0.30E+00	2
768.3499	88	226	RADA	1.60E+03 Y	0.49E+01	2
768.3500	92	234	UDA	2.46E+05 Y	0.49E+01	2
768.3500	90	230	THDA	7.54E+04 Y	0.49E+01	2
772.2775	90	232	THDA	1.40E+10 Y	0.11E+01	3
772.2776	88	228	RAAC	5.7500 Y	0.11E+01	3
778.9030	63	152	EU	13.5100 Y	0.13E+02	1
785.5095	90	228	THDA	1.9130 Y	0.11E+01	3
785.8274	88	226	RADA	1.60E+03 Y	0.10E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
785.8275	92	234	UDA	2.46E+05 Y	0.10E+01	3
785.8276	90	230	THDA	7.54E+04 Y	0.10E+01	3
786.2867	94	238	PU	87.7000 Y	0.47E-05	5
794.7914	90	232	THDA	1.40E+10 Y	0.46E+01	2
794.7916	88	228	RAAC	5.7500 Y	0.46E+01	3
795.8674	55	134	CS	2.0652 Y	0.86E+02	1
801.9511	55	134	CS	2.0652 Y	0.87E+01	3
806.1550	88	226	RADA	1.60E+03 Y	0.12E+01	3
806.1552	92	234	UDA	2.46E+05 Y	0.12E+01	3
806.1552	90	230	THDA	7.54E+04 Y	0.12E+01	3
810.7915	27	58	CO	70.8600 D	0.99E+02	1
818.0310	47	110	AGM	249.7600 D	0.73E+01	3
834.8430	25	54	MN	312.3000 D	0.10E+03	1
835.6032	90	232	THDA	1.40E+10 Y	0.17E+01	3
835.6033	88	228	RAAC	5.7500 Y	0.17E+01	3
860.5640	90	228	THDA	1.9130 Y	0.45E+01	2
863.9741	27	58	CO	70.8600 D	0.69E+00	3
867.3900	63	152	EU	13.5100 Y	0.42E+01	2
871.1190	41	94	NB	2.00E+04 Y	0.10E+03	1
873.2299	63	154	EU	8.5880 Y	0.11E+02	2
873.4679	45	106	RH	29.80000 S	0.43E+00	3
884.6850	47	110	AGM	249.7600 D	0.73E+02	1
911.1572	90	232	THDA	1.40E+10 Y	0.29E+02	1
911.1573	88	228	RAAC	5.7500 Y	0.29E+02	1
934.0389	88	226	RADA	1.60E+03 Y	0.32E+01	3
934.0390	92	234	UDA	2.46E+05 Y	0.32E+01	3
934.0391	90	230	THDA	7.54E+04 Y	0.32E+01	3
937.4930	47	110	AGM	249.7600 D	0.34E+02	2
964.1102	63	152	EU	13.5100 Y	0.14E+02	2
964.6375	90	232	THDA	1.40E+10 Y	0.58E+01	2
964.6376	88	228	RAAC	5.7500 Y	0.58E+01	3
968.9707	90	232	THDA	1.40E+10 Y	0.17E+02	2
968.9708	88	228	RAAC	5.7500 Y	0.17E+02	2
996.3293	63	154	EU	8.5880 Y	0.10E+02	2
1000.9970	92	238	UPA	4.47E+09 Y	0.84E+00	1
1001.7018	73	182	TA	115.0000 D	0.20E+01	3
1004.7747	63	154	EU	8.5880 Y	0.18E+02	2
1038.5920	55	134	CS	2.0652 Y	0.99E+00	3
1050.3599	45	106	RH	29.80000 S	0.15E+01	2
1085.8420	63	152	EU	13.5100 Y	0.10E+02	2
1089.7670	63	152	EU	13.5100 Y	0.17E+01	3
1112.0870	63	152	EU	13.5100 Y	0.14E+02	2
1115.5460	30	65	ZN	244.2600 D	0.51E+02	1
1120.2727	88	226	RADA	1.60E+03 Y	0.15E+02	2
1120.2737	92	234	UDA	2.46E+05 Y	0.15E+02	2
1120.2738	90	230	THDA	7.54E+04 Y	0.15E+02	2
1121.3010	73	182	TA	115.0000 D	0.35E+02	2
1128.0422	45	106	RH	29.80000 S	0.40E+00	3
1129.6609	13	26	AL	7.20E+05 Y	0.25E+01	3
1155.1825	88	226	RADA	1.60E+03 Y	0.17E+01	3
1155.1835	92	234	UDA	2.46E+05 Y	0.17E+01	3
1155.1836	90	230	THDA	7.54E+04 Y	0.17E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
1167.9443	55	134	CS	2.0652 Y	0.18E+01	3
1173.2380	27	60	CO	5.2719 Y	0.10E+03	2
1189.0500	73	182	TA	115.0000 D	0.17E+02	2
1212.9700	63	152	EU	13.5100 Y	0.14E+01	3

1221.4080	73	182	TA	115.0000	D	0.27E+02	1
1231.0160	73	182	TA	115.0000	D	0.12E+02	2
1238.1073	88	226	RADA	1.60E+03	Y	0.59E+01	2
1238.1083	92	234	UDA	2.46E+05	Y	0.59E+01	2
1238.1084	90	230	THDA	7.54E+04	Y	0.59E+01	2
1257.4180	73	182	TA	115.0000	D	0.15E+01	3
1274.5420	11	22	NA	2.6032	Y	0.10E+03	1
1274.5435	63	154	EU	8.5880	Y	0.35E+02	1
1280.9519	88	226	RADA	1.60E+03	Y	0.15E+01	3
1280.9529	92	234	UDA	2.46E+05	Y	0.15E+01	3
1280.9530	90	230	THDA	7.54E+04	Y	0.15E+01	3
1289.1560	73	182	TA	115.0000	D	0.14E+01	3
1299.1520	63	152	EU	13.5100	Y	0.16E+01	3
1332.5020	27	60	CO	5.2719	Y	0.10E+03	1
1365.1844	55	134	CS	2.0652	Y	0.30E+01	3
1377.6586	88	226	RADA	1.60E+03	Y	0.40E+01	3
1377.6595	92	234	UDA	2.46E+05	Y	0.40E+01	3
1377.6597	90	230	THDA	7.54E+04	Y	0.40E+01	3
1384.3000	47	110	AGM	249.7600	D	0.24E+02	2
1401.4832	88	226	RADA	1.60E+03	Y	0.14E+01	3
1401.4841	92	234	UDA	2.46E+05	Y	0.14E+01	3
1401.4843	90	230	THDA	7.54E+04	Y	0.14E+01	3
1407.9697	88	226	RADA	1.60E+03	Y	0.25E+01	3
1407.9707	92	234	UDA	2.46E+05	Y	0.25E+01	3
1407.9708	90	230	THDA	7.54E+04	Y	0.25E+01	3
1408.0220	63	152	EU	13.5100	Y	0.21E+02	2
1459.1932	90	232	THDA	1.40E+10	Y	0.11E+01	3
1459.1942	88	228	RAAC	5.7500	Y	0.11E+01	3
1460.8300	19	40	K	1.26E+09	Y	0.11E+02	1
1475.7880	47	110	AGM	249.7600	D	0.40E+01	3
1489.1600	58	144	CEPR	285.0000	D	0.28E+00	3
1496.0311	90	232	THDA	1.40E+10	Y	0.10E+01	3
1496.0321	88	228	RAAC	5.7500	Y	0.11E+01	3
1505.0400	47	110	AGM	249.7600	D	0.13E+02	2
1509.2189	88	226	RADA	1.60E+03	Y	0.22E+01	3
1509.2198	92	234	UDA	2.46E+05	Y	0.22E+01	3
1509.2200	90	230	THDA	7.54E+04	Y	0.22E+01	3
1562.2191	45	106	RH	29.80000	S	0.16E+00	3
1562.3030	47	110	AGM	249.7600	D	0.10E+01	3
1588.2271	90	232	THDA	1.40E+10	Y	0.36E+01	2
1588.2280	88	228	RAAC	5.7500	Y	0.36E+01	3
1593.0000	63	154	EU	8.5880	Y	0.10E+01	3
1596.5818	63	154	EU	8.5880	Y	0.18E+01	3
1620.7350	90	228	THDA	1.9130	Y	0.15E+01	3
1630.4746	90	232	THDA	1.40E+10	Y	0.19E+01	3
1630.4756	88	228	RAAC	5.7500	Y	0.19E+01	3
1661.2577	88	226	RADA	1.60E+03	Y	0.11E+01	3
1661.2587	92	234	UDA	2.46E+05	Y	0.11E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
1661.2588	90	230	THDA	7.54E+04	Y	0.11E+01
1674.7526	27	58	CO	70.8600	D	0.52E+00
1729.5803	88	226	RADA	1.60E+03	Y	0.30E+01
1729.5813	92	234	UDA	2.46E+05	Y	0.30E+01
1729.5814	90	230	THDA	7.54E+04	Y	0.30E+01
1764.4901	88	226	RADA	1.60E+03	Y	0.16E+02
1764.4911	92	234	UDA	2.46E+05	Y	0.16E+02
1764.4912	90	230	THDA	7.54E+04	Y	0.16E+02
1808.6484	13	26	AL	7.20E+05	Y	0.10E+03
1847.4147	88	226	RADA	1.60E+03	Y	0.21E+01

1847.4156	92	234	UDA	2.46E+05	Y	0.21E+01	3
1847.4158	90	230	THDA	7.54E+04	Y	0.21E+01	3
2118.5254	88	226	RADA	1.60E+03	Y	0.12E+01	3
2118.5264	92	234	UDA	2.46E+05	Y	0.12E+01	3
2118.5266	90	230	THDA	7.54E+04	Y	0.12E+01	3
2185.6621	58	144	CEPR	285.0000	D	0.70E+00	1
2204.0906	88	226	RADA	1.60E+03	Y	0.50E+01	2
2204.0916	92	234	UDA	2.46E+05	Y	0.50E+01	2
2204.0918	90	230	THDA	7.54E+04	Y	0.50E+01	2
2447.6750	88	226	RADA	1.60E+03	Y	0.16E+01	3
2447.6760	92	234	UDA	2.46E+05	Y	0.16E+01	3
2447.6763	90	230	THDA	7.54E+04	Y	0.16E+01	3
2614.5330	90	228	THDA	1.9130	Y	0.36E+02	1

A.2 Isotope Library GE\$ISOTOPE\$LIB:SWEPP.LIB, Sorted by Nuclide

This listing was generated using the VAXGAP PRLIB command [2].

SWEPP

PAGE 1

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
11	22	NA	2.6032 Y	1274.5420	9.99E+01	1
13	26	AL	7.20E+05 Y	1129.6609	2.47E+00	3
				1808.6484	9.98E+01	1
19	40	K	1.26E+09 Y	1460.8300	1.07E+01	1
25	54	MN	312.3000 D	834.8430	1.00E+02	1
27	58	CO	70.8600 D	810.7915	9.94E+01	1
				863.9741	6.90E-01	3
				1674.7526	5.19E-01	3
27	60	CO	5.2719 Y	1173.2380	9.99E+01	2
				1332.5020	1.00E+02	1
30	65	ZN	244.2600 D	1115.5460	5.08E+01	1
40	95	ZR	63.9800 D	724.1990	4.42E+01	2
				756.7290	5.45E+01	1
40	95	ZRNB	63.9800 D	724.1991	4.42E+01	2
				756.7291	5.45E+01	2
				765.8070	2.20E+02	1
41	94	NB	2.00E+04 Y	702.6450	9.98E+01	2
				871.1190	9.99E+01	1
45	106	RH	29.80000 S	511.8647	2.07E+01	2
				616.1807	7.38E-01	2

				621.9170	9.81E+00	1
				873.4679	4.29E-01	3
				1050.3599	1.53E+00	2
				1128.0422	3.98E-01	3
				1562.2191	1.58E-01	3
47	108 AGM	127.0000 Y		433.9360	9.05E+01	2
				614.2810	8.98E+01	1
				722.9290	9.08E+01	2
47	110 AGM	249.7600 D		446.8110	3.75E+00	3
				620.3600	2.81E+00	3
				657.7620	9.46E+01	2
				677.6230	1.04E+01	2
				687.0150	6.43E+00	3
				706.6820	1.64E+01	2
				744.2770	4.71E+00	3
				763.9440	2.23E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
47	110 AGM	249.7600 D		818.0310	7.31E+00	3
				884.6850	7.27E+01	1
				937.4930	3.44E+01	2
				1384.3000	2.43E+01	2
				1475.7880	4.02E+00	3
				1505.0400	1.30E+01	2
				1562.3030	1.03E+00	3
51	125 SB	2.7590 Y		176.3161	6.90E+00	2
				380.4155	1.51E+00	3
				427.8752	2.94E+01	1
				463.3670	1.04E+01	2
				600.5002	1.78E+01	2
				606.6332	5.00E+00	2
				635.8897	1.13E+01	2
				671.3815	1.81E+00	3
55	134 CS	2.0652 Y		475.3573	1.50E+00	3
				563.2370	8.38E+00	3
				569.3207	1.54E+01	2
				604.7101	9.76E+01	2
				795.8674	8.55E+01	1
				801.9511	8.70E+00	3
				1038.5920	9.91E-01	3
				1167.9443	1.79E+00	3
				1365.1844	3.02E+00	3
55	137 CS	30.2500 Y		661.6600	8.52E+01	1
58	144 CE	285.0000 D		133.5386	1.12E+01	1
58	144 CEPR	285.0000 D		133.5387	1.12E+01	2
				696.5100	1.34E+00	2
				1489.1600	2.79E-01	3
				2185.6621	7.00E-01	1
63	155 EU	4.8500 Y		105.3129	2.23E+01	1
63	152 EU	13.5100 Y		121.7824	2.84E+01	2

PAGE 2

				244.6989	7.53E+00	2
				344.2811	2.66E+01	2
				411.1260	2.24E+00	3
				443.9034	3.12E+00	3
				778.9030	1.30E+01	1
				867.3900	4.21E+00	2
				964.1102	1.45E+01	2
				1085.8420	1.01E+01	2
				1089.7670	1.73E+00	3
				1112.0870	1.35E+01	2
				1212.9700	1.41E+00	3
				1299.1520	1.63E+00	3
				1408.0220	2.09E+01	2
63	154	EU	8.5880 Y	123.1001	4.05E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
63	154	EU	8.5880 Y	247.9678	6.60E+00	2
				591.8112	4.84E+00	2
				692.4855	1.69E+00	3
				723.3563	1.97E+01	2
				756.8083	4.34E+00	2
				873.2299	1.14E+01	2
				996.3293	1.03E+01	2
				1004.7747	1.79E+01	2
				1274.5435	3.55E+01	1
				1593.0000	1.03E+00	3
				1596.5818	1.83E+00	3
73	182	TA	115.0000 D	100.1065	1.40E+01	2
				113.6723	1.92E+00	3
				152.4308	7.18E+00	2
				156.3874	2.73E+00	3
				179.3948	3.14E+00	3
				198.3530	1.52E+00	3
				222.1099	7.59E+00	2
				229.3220	3.63E+00	3
				264.0755	3.63E+00	3
				1001.7018	2.00E+00	3
				1121.3010	3.48E+01	2
				1189.0500	1.65E+01	2
				1221.4080	2.73E+01	1
				1231.0160	1.16E+01	2
				1257.4180	1.52E+00	3
				1289.1560	1.40E+00	3
88	228	RAAC	5.7500 Y	209.3853	4.12E+00	3
				270.2562	3.77E+00	3
				328.0699	3.46E+00	3
				338.4200	1.24E+01	2
				409.6247	2.20E+00	3
				463.1050	4.56E+00	3
				562.6523	1.01E+00	3
				581.5248	1.50E+00	3
				755.2774	1.32E+00	3
				772.2776	1.09E+00	3
				794.7916	4.59E+00	3
				835.6033	1.71E+00	3
				911.1573	2.87E+01	1

PAGE 3

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
88	226	RADA	1.60E+03 Y	241.9192	7.41E+00	2
				964.6376	5.77E+00	3
				968.9708	1.74E+01	2
				1459.1942	1.06E+00	3
				1496.0321	1.05E+00	3
				1588.2280	3.59E+00	3
				1630.4756	1.95E+00	3
88	226	RADA	1.60E+03 Y	295.0908	1.87E+01	2
				351.8664	3.58E+01	2
				609.3111	4.61E+01	1
				665.4420	1.56E+00	3
				768.3499	4.89E+00	2
				785.8274	1.05E+00	3
				806.1550	1.23E+00	3
				934.0389	3.16E+00	3
				1120.2727	1.50E+01	2
				1155.1825	1.69E+00	3
				1238.1073	5.92E+00	2
				1280.9519	1.47E+00	3
				1377.6586	4.02E+00	3
				1401.4832	1.39E+00	3
				1407.9697	2.48E+00	3
				1509.2189	2.19E+00	3
				1661.2577	1.15E+00	3
				1729.5803	3.05E+00	3
				1764.4901	1.59E+01	2
				1847.4147	2.12E+00	3
				2118.5254	1.21E+00	3
				2204.0906	4.99E+00	2
				2447.6750	1.55E+00	3
89	227	ACDA	21.7700 Y	122.3107	1.18E+00	3
				154.1767	5.53E+00	2
				210.5810	1.12E+00	3
				235.9696	1.11E+01	2
				256.2411	6.67E+00	2
				269.3930	1.35E+01	1
				286.1100	1.57E+00	2
				299.9681	2.08E+00	2
				304.5109	1.08E+00	3
				323.8753	3.86E+00	2
				329.8370	2.71E+00	2
				338.2851	2.75E+00	2
				350.0908	1.26E+01	2
				444.9429	1.26E+00	3
90	230	THDA	7.54E+04 Y	186.1109	3.36E+00	3
				241.9194	7.41E+00	2
				295.0910	1.87E+01	2
				351.8666	3.58E+01	2
				609.3113	4.61E+01	1
				665.4422	1.56E+00	3
				768.3500	4.89E+00	2
				785.8276	1.05E+00	3
				806.1552	1.23E+00	3
				934.0391	3.16E+00	3

1120.2738	1.50E+01	2
1155.1836	1.69E+00	3
1238.1084	5.92E+00	2

PAGE 5

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
90	230	THDA	7.54E+04 Y	1280.9530	1.47E+00	3
				1377.6597	4.02E+00	3
				1401.4843	1.39E+00	3
				1407.9708	2.48E+00	3
				1509.2200	2.19E+00	3
				1661.2588	1.15E+00	3
				1729.5814	3.05E+00	3
				1764.4912	1.59E+01	2
				1847.4158	2.12E+00	3
				2118.5266	1.21E+00	3
				2204.0918	4.99E+00	2
				2447.6763	1.55E+00	3
90	232	THDA	1.40E+10 Y	209.3852	4.12E+00	2
				270.2561	3.77E+00	2
				328.0698	3.46E+00	2
				338.4199	1.24E+01	2
				409.6246	2.20E+00	2
				463.1049	4.56E+00	2
				562.6522	1.01E+00	3
				581.5247	1.50E+00	3
				755.2773	1.32E+00	3
				772.2775	1.09E+00	3
				794.7914	4.59E+00	2
				835.6032	1.71E+00	3
				911.1572	2.87E+01	1
				964.6375	5.77E+00	2
				968.9707	1.74E+01	2
				1459.1932	1.06E+00	3
				1496.0311	1.05E+00	3
				1588.2271	3.59E+00	2
				1630.4746	1.95E+00	3
90	228	THDA	1.9130 Y	238.6320	4.35E+01	2
				240.9870	4.07E+00	3
				277.2830	2.30E+00	3
				300.0336	3.25E+00	3
				583.1910	3.06E+01	2
				727.2466	6.66E+00	2
				785.5095	1.10E+00	3
				860.5640	4.50E+00	2
				1620.7350	1.50E+00	3
				2614.5330	3.59E+01	1
91	231	PADA	3.28E+04 Y	122.3106	1.18E+00	3
				144.1835	3.22E+00	2
				154.1766	5.52E+00	2
				210.5809	1.12E+00	3
				235.9695	1.11E+01	2
				256.2410	6.66E+00	2
				269.3929	1.34E+01	1
				283.6729	1.65E+00	2

PAGE 6

91	231	PADA	3.28E+04 Y	286.1099	1.57E+00	2
				300.0229	4.48E+00	2
				302.6690	2.47E+00	2
				304.5108	1.07E+00	3
				323.8752	3.85E+00	2
				329.9137	4.06E+00	2
				338.2850	2.75E+00	2
				350.0907	1.26E+01	2
				444.9428	1.25E+00	3
92	232	U	69.8000 Y	57.8137	2.00E-01	1
				129.0348	6.82E-02	2
92	235	U	7.04E+08 Y	109.1784	1.54E+00	3
				143.7680	1.10E+01	2
				163.3570	5.08E+00	3
				185.7220	5.72E+01	1
				202.1060	1.08E+00	3
				205.3180	5.01E+00	3
92	233	U	1.59E+05 Y	118.9680	4.06E-03	3
				120.8160	3.32E-03	3
				135.3284	2.32E-03	3
				146.3450	6.57E-03	2
				164.5220	6.23E-03	2
				208.1710	2.29E-03	3
				245.3450	3.62E-03	3
				291.3540	5.37E-03	3
				317.1327	7.76E-03	1
				320.5410	2.90E-03	3
92	234	UDA	2.46E+05 Y	241.9193	7.41E+00	2
				295.0909	1.87E+01	2
				351.8665	3.58E+01	2
				609.3112	4.61E+01	1
				665.4421	1.56E+00	3
				768.3500	4.89E+00	2
				785.8275	1.05E+00	3
				806.1552	1.23E+00	3
				934.0390	3.16E+00	3
				1120.2737	1.50E+01	2
				1155.1835	1.69E+00	3
				1238.1083	5.92E+00	2
				1280.9529	1.47E+00	3
				1377.6595	4.02E+00	3
				1401.4841	1.39E+00	3
				1407.9707	2.48E+00	3
				1509.2198	2.19E+00	3
				1661.2587	1.15E+00	3
				1729.5813	3.05E+00	3
				1764.4911	1.59E+01	2
				1847.4156	2.12E+00	3
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
92	234	UDA	2.46E+05 Y	2118.5264	1.21E+00	3
				2204.0916	4.99E+00	2
				2447.6760	1.55E+00	3
92	238	UPA	4.47E+09 Y	258.1807	8.10E-02	2

			766.4125	2.96E-01	2	
			1000.9970	8.39E-01	1	
93	237 NPPA	2.14E+06 Y	300.1750	6.63E+00	2	
			312.0117	3.86E+01	1	
			340.5896	4.50E+00	2	
			398.6635	1.41E+00	3	
			415.9324	1.74E+00	3	
94	238 PU	87.7000 Y	99.8530	7.35E-03	2	
			152.7200	9.37E-04	1	
			200.9572	4.10E-06	5	
			742.8172	7.78E-06	5	
			766.4124	3.30E-05	5	
			786.2867	4.71E-06	5	
94	241 PU	14.4000 Y	103.6529	1.02E-04	2	
			148.5396	1.86E-04	1	
			159.9276	6.58E-06	3	
94	240 PU	6.56E+03 Y	104.2340	7.14E-03	2	
			160.3080	4.02E-04	1	
94	239 PU	2.41E+04 Y	125.2100	7.11E-05	4	
			129.2960	6.31E-03	2	
			144.2010	3.04E-04	3	
			146.0940	1.19E-04	4	
			160.0674	5.00E-06	5	
			161.4820	1.23E-04	4	
			203.5500	5.69E-04	3	
			332.8450	4.94E-04	3	
			345.0130	5.56E-04	3	
			368.5083	8.96E-05	4	
			375.0540	1.55E-03	3	
			380.1910	3.05E-04	3	
			382.6980	2.59E-04	3	
			392.9140	5.53E-04	4	
			413.7130	1.47E-03	1	
			422.5980	1.22E-04	4	
			451.4810	1.89E-04	4	
94	241 PUDA	14.4000 Y	164.5925	1.87E+00	3	
			208.0078	2.16E+01	2	
			267.5441	7.31E-01	1	
			332.3609	1.21E+00	3	
95	241 AM	432.7000 Y	102.9516	1.96E-02	5	
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
95	241 AM	432.7000 Y	125.2880	4.10E-03	5	
			146.5495	4.60E-04	1	
			161.6994	9.00E-06	6	
			165.9215	2.40E-05	6	
			169.5593	1.72E-04	6	
			203.8694	2.90E-06	6	
			208.0077	7.90E-04	6	
			322.5410	1.50E-04	6	
			332.3608	1.50E-04	6	
			335.4005	4.95E-04	6	

		368.5925	2.20E-04	6
		376.5762	1.38E-04	6
		662.4258	3.60E-04	6
		722.6994	1.30E-04	6
96	244 CM	18.1000 Y	42.8167	2.48E-02
				1

A.3 Insert Library GE\$LIMIT\$LIB:INSERT.LIB

This listing was generated using the "P" (for print) option of MAKE_LIM [11].

LISTING FILE FOR THE LIMIT LIBRARY INSERT
GENERATED ON 6-JUL-94 AT 13:55:41

The current limit library documentation file is
ge\$limit\$lib:INSERT.doc, revision date/time = 6-JUL-1994 13:55:40.64

The associated limit library file is
ge\$limit\$lib:INSERT.lib, revision date/time = 26-JUL-1993 10:07:42.06

The associated isotope library file is
ge\$isotope\$lib:SWEPP.lib, revision date/time = 31-JAN-1994 15:51:52.95

Entry	Nuclide	Energy	ICODE	Half-life	Branching ratio	Isomer?
1	94-PU -239	144.2010	3	2.41E+04 Y	0.000	N
2	94-PU -239	161.4820	4	2.41E+04 Y	0.000	N
3	94-PU -240	160.3080	1	6.56E+03 Y	0.000	N
4	94-PU -241	148.5396	1	14.4000 Y	0.000	N
5	95-AM -241	146.5495	1	432.7000 Y	0.000	N

A.4 Limit Library GE\$LIMIT\$LIB:SWEPPREQ.LIB

This listing was generated using the "P" (for print) option of MAKE_LIM [11].

LISTING FILE FOR THE LIMIT LIBRARY SWEPPREQ
GENERATED ON 6-JUL-94 AT 13:55:54

The current limit library documentation file is
ge\$limit\$lib:SWEPPREQ.doc, revision date/time = 6-JUL-1994 13:55:54.34

The associated limit library file is
ge\$limit\$lib:SWEPPREQ.lib, revision date/time = 31-JAN-1994 14:42:09.75

The associated isotope library file is
ge\$isotope\$lib:SWEPP.lib, revision date/time = 31-JAN-1994 15:51:52.95

Entry	Nuclide	Energy	ICODE	Half-life	Branching ratio	Isomer?
1	63-EU -152	121.7824	2	13.5100 Y	28.370	N
2	63-EU -152	778.9030	1	13.5100 Y	12.970	N
3	63-EU -152	1408.0220	2	13.5100 Y	20.850	N
4	92- U -235	143.7680	2	7.04E+08 Y	10.960	N
5	92- U -235	163.3570	3	7.04E+08 Y	5.080	N
6	92- U -235	185.7220	1	7.04E+08 Y	57.200	N
7	92- U -235	205.3180	3	7.04E+08 Y	5.010	N
8	94-PU -238	152.7200	1	87.7000 Y	0.001	N
9	94-PU -239	129.2960	2	2.41E+04 Y	0.006	N
10	94-PU -239	203.5500	3	2.41E+04 Y	0.001	N
11	94-PU -239	332.8450	3	2.41E+04 Y	0.000	N
12	94-PU -239	345.0130	3	2.41E+04 Y	0.001	N
13	94-PU -239	375.0540	3	2.41E+04 Y	0.002	N
14	94-PU -239	413.7130	1	2.41E+04 Y	0.001	N
15	95-AM -241	125.2880	5	432.7000 Y	0.004	N
16	95-AM -241	165.9215	6	432.7000 Y	0.000	N
17	95-AM -241	169.5593	6	432.7000 Y	0.000	N

A.5 Interference Library GE\$INTERF\$LIB:SWEPPIF.LIB

This listing was generated using the VAXGAP EDINTFER command [2].

INTERFERENCE LIBRARY				SWEPPIF			
CONTAMINATED LINES:				CLEAN LINES:			
ENERGY	Z	ELE	A ISOMER	ENERGY	Z	ELE	A ISOMER
152.72	94	PU	238 NO	156.39	73	TA	182 NO
661.66	55	CS	137 NO	722.69	95	AM	241 NO
756.73	40	ZR	95 NO	996.33	63	EU	154 NO
1000.99	92	UPA	238 NO	1189.05	73	TA	182 NO
1274.54	11	NA	22 NO	1004.77	63	EU	154 NO
144.20	94	PU	239 NO	163.36	92	U	235 NO
146.09	94	PU	239 NO	144.20	94	PU	239 NO
146.55	95	AM	241 NO	144.20	94	PU	239 NO
160.31	94	PU	240 NO	148.54	94	PU	241 NO
125.29	95	AM	241 NO	129.30	94	PU	239 NO
345.01	94	PU	239 NO	778.90	63	EU	152 NO
160.31	94	PU	240 NO	129.30	94	PU	239 NO
161.48	94	PU	239 NO	165.92	95	AM	241 NO
203.55	94	PU	239 NO	165.92	95	AM	241 NO

A.6 Isotope Library GE\$ISOTOPE\$LIB:MASTER.LIB, Sorted by Energy

This listing was generated using the VAXGAP PRLIB command [2].

MASTER

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
27.7908	52	129	TE	1.16000 H	0.16E+02	2
30.4000	41	93	NBM	16.1300 Y	0.52E-03	1
39.5710	53	129	I	1.57E+07 Y	0.75E+01	1
46.5200	82	210	PB	22.3000 Y	0.41E+01	1
53.1480	56	133	BA	10.5700 Y	0.22E+01	3
53.1716	82	214	PB	26.80000 M	0.11E+01	3
53.1717	88	226	RAPB	1.60E+03 Y	0.11E+01	3
53.2306	92	234	U	2.46E+05 Y	0.12E+00	1
57.3817	58	143	CE	1.3750 D	0.12E+02	2
57.6091	52	127	TEM	109.0000 D	0.51E+00	1
57.8137	92	232	U	69.8000 Y	0.20E+00	1
58.9921	90	232	TH	1.40E+10 Y	0.19E+00	1
58.9922	90	232	THPB	1.40E+10 Y	0.19E+00	4
58.9923	90	232	THBI	1.40E+10 Y	0.19E+00	3
58.9924	90	232	THAC	1.40E+10 Y	0.19E+00	3
58.9925	90	232	THTL	1.40E+10 Y	0.19E+00	3

PAGE 1

58.9926	90	232	THRA	1.40E+10	Y	0.19E+00	3
58.9927	90	232	THTH	1.40E+10	Y	0.19E+00	2
59.0315	59	144	PRM	7.20000	M	0.79E-01	1
59.5364	95	241	AM	432.7000	Y	0.36E+02	1
60.0153	63	155	EU	4.8500	Y	0.11E+01	3
61.4268	51	127	SB	3.8500	D	0.14E+01	2
61.4618	93	239	NP	2.3550	D	0.13E+01	3
63.1208	70	169	YB	32.0220	D	0.44E+02	2
63.2878	90	234	TH	4.47E+09	Y	0.38E+01	1
65.7207	73	182	TA	115.0000	D	0.28E+01	3
66.0599	34	75	SE	119.6400	D	0.11E+01	3
66.9373	55	136	CS	13.1600	D	0.12E+02	2
67.7500	73	182	TA	115.0000	D	0.41E+02	2
69.6734	64	153	GD	241.6000	D	0.23E+01	3
69.6734	62	153	SM	1.9458	D	0.53E+01	2
71.9834	74	187	W	23.90000	H	0.11E+02	2
72.8035	90	232	THTL	1.40E+10	Y	0.77E+00	9
74.8137	90	232	THPB	1.40E+10	Y	0.11E+02	8
74.9693	90	232	THTL	1.40E+10	Y	0.13E+01	8
75.3429	91	233	PA	27.0000	D	0.13E+01	3
77.1073	90	232	THPB	1.40E+10	Y	0.18E+02	8
77.3432	78	197	PT	18.30000	H	0.17E+02	2
77.3433	80	197	HG	2.6710	D	0.19E+02	1
77.5714	56	142	BA	10.60000	M	0.11E+02	2
79.1310	47	108	AGM	127.0000	Y	0.76E+01	2
79.6117	56	133	BA	10.5700	Y	0.26E+01	3
80.1034	58	144	CE	285.0000	D	0.15E+01	2
80.1845	53	131	I	8.0210	D	0.26E+01	3
80.9601	42	101	MO	14.60000	M	0.38E+01	2
80.9894	54	133	XE	5.2440	D	0.38E+02	1
80.9895	56	133	BA	10.5700	Y	0.34E+02	2
81.1304	52	131	TEM	1.2500	D	0.41E+01	2
84.2551	69	170	TM	128.6000	D	0.33E+01	1
84.2570	90	228	TH	1.9130	Y	0.12E+01	1
84.2571	90	232	THTH	1.40E+10	Y	0.12E+01	1
84.6808	73	182	TA	115.0000	D	0.27E+01	3

PAGE 2

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
84.7132	73	183	TA	5.1000	D	0.13E+01	3
84.7893	90	232	THTL	1.40E+10	Y	0.46E+00	9
86.3575	55	136	CS	13.1600	D	0.63E+01	2
86.5423	63	155	EU	4.8500	Y	0.33E+02	2
86.6519	91	233	PA	27.0000	D	0.20E+01	3
86.7879	65	160	TB	72.3000	D	0.13E+02	2
87.1902	90	232	THPB	1.40E+10	Y	0.63E+01	8
88.0341	46	109	PD	13.70000	H	0.36E+01	1
88.0342	48	109	CD	1.2666	Y	0.37E+01	1
88.9854	63	156	EU	15.1900	D	0.89E+01	2
89.3561	72	175	HF	70.0000	D	0.23E+01	3
89.9547	90	232	THAC	1.40E+10	Y	0.34E+01	8
91.1040	60	147	ND	10.9800	D	0.28E+02	2
92.5640	90	234	TH	4.47E+09	Y	0.54E+01	2
93.3500	90	232	THAC	1.40E+10	Y	0.56E+01	8
93.6151	70	169	YB	32.0220	D	0.27E+01	3
94.7035	66	165	DY	2.33400	H	0.36E+01	1
96.7337	34	75	SE	119.6400	D	0.35E+01	3
97.4316	64	153	GD	241.6000	D	0.28E+02	1
99.0808	73	183	TA	5.1000	D	0.65E+01	2
99.5475	89	228	AC	6.13000	H	0.13E+01	2
99.5476	90	232	THAC	1.40E+10	Y	0.13E+01	2

100.1065	73 182 TA	115.0000 D	0.14E+02	2
102.0643	52 131 TEM	1.2500 D	0.79E+01	2
103.1806	62 153 SM	1.9458 D	0.28E+02	1
103.1807	64 153 GD	241.6000 D	0.20E+02	2
105.3129	63 155 EU	4.8500 Y	0.22E+02	1
105.3616	90 232 THAC	1.40E+10 Y	0.20E+01	8
106.1272	93 239 NP	2.3550 D	0.27E+02	2
107.9332	73 183 TA	5.1000 D	0.11E+02	2
109.1784	92 235 U	7.04E+08 Y	0.15E+01	3
109.7802	70 169 YB	32.0220 D	0.17E+02	2
109.8945	8 19 O	26.91000 S	0.25E+01	3
111.6609	68 171 ER	7.52000 H	0.21E+02	2
111.8601	52 132 TE	3.2583 D	0.19E+01	3
112.9489	71 177 LU	6.7100 D	0.64E+01	2
113.6723	73 182 TA	115.0000 D	0.19E+01	3
113.8060	70 175 YB	4.1900 D	0.19E+01	2
116.4001	52 132 TE	3.2583 D	0.19E+01	3
116.6894	68 171 ER	7.52000 H	0.23E+01	3
118.1901	70 169 YB	32.0220 D	0.19E+01	3
121.1192	34 75 SE	119.6400 D	0.17E+02	2
121.2579	61 147 PM	2.6234 Y	0.29E-02	1
121.7190	36 90 KR	32.32000 S	0.36E+02	2
121.7758	63 152 EUM	9.32000 H	0.72E+01	2
121.7824	63 152 EU	13.5100 Y	0.28E+02	2
122.0614	27 57 CO	271.7900 D	0.86E+02	1
123.1001	63 154 EU	8.5880 Y	0.40E+02	2
123.7774	56 131 BA	11.8000 D	0.29E+02	2
124.0523	68 171 ER	7.52000 H	0.91E+01	2
127.2200	43 101 TC	14.20000 M	0.29E+01	3
129.0348	89 228 AC	6.13000 H	0.29E+01	2

ENERGY	Z A NUCLIDE	HALF LIFE	BR	I-CODE
129.0349	90 232 THAC	1.40E+10 Y	0.29E+01	2
129.0350	92 232 U	69.8000 Y	0.68E-01	2
129.2960	94 239 PU	2.41E+04 Y	0.63E-02	2
129.4341	76 191 OS	15.4000 D	0.26E+02	1
129.5966	44 105 RU	4.44000 H	0.56E+01	2
129.5967	45 105 RHM	45.00000 S	0.20E+02	1
130.5239	70 169 YB	32.0220 D	0.11E+02	2
131.4968	90 228 TH	1.9130 Y	0.13E+00	2
131.4969	90 232 THTH	1.40E+10 Y	0.13E+00	2
132.9411	72 181 HF	42.3900 D	0.43E+02	2
133.5386	58 144 CE	285.0000 D	0.11E+02	1
133.5819	56 131 BA	11.8000 D	0.22E+01	3
134.2277	74 187 W	23.90000 H	0.86E+01	2
135.4063	53 134 I	52.60000 M	0.41E+01	2
136.0017	34 75 SE	119.6400 D	0.59E+02	2
136.1687	72 181 HF	42.3900 D	0.58E+01	3
136.4743	27 57 CO	271.7900 D	0.11E+02	2
138.0396	55 138 CS	32.20000 M	0.15E+01	3
138.3717	49 116 INM	54.15000 M	0.33E+01	2
140.4740	42 99 MO	2.7470 D	0.91E+02	1
140.5110	43 99 TCM	6.00700 H	0.89E+02	1
142.6520	26 59 FE	44.5100 D	0.98E+00	3
143.7680	92 235 U	7.04E+08 Y	0.11E+02	1
144.1246	73 183 TA	5.1000 D	0.25E+01	3
144.2010	94 239 PU	2.41E+04 Y	0.30E-03	3
145.4404	58 141 CE	32.5000 D	0.49E+02	1
149.1725	44 105 RU	4.44000 H	0.17E+01	2
149.7166	52 131 TE	25.00000 M	0.69E+02	1

149.7167	52	131	TEM	1.2500	D	0.51E+01	2
151.1809	36	85	KRM	4.48000	H	0.75E+02	1
152.4308	73	182	TA	115.0000	D	0.72E+01	2
153.2948	55	136	CS	13.1600	D	0.75E+01	2
153.9039	54	138	XE	14.08000	M	0.60E+01	2
155.0641	75	188	RE	16.98000	H	0.15E+02	1
156.0217	50	117	SNM	13.6100	D	0.21E+01	3
156.3874	73	182	TA	115.0000	D	0.27E+01	3
158.5603	50	117	SNM	13.6100	D	0.86E+02	1
158.9887	52	123	TE	119.7000	D	0.84E+02	1
159.3815	21	47	SC	3.3510	D	0.68E+02	1
161.1445	73	183	TA	5.1000	D	0.12E+02	2
162.3250	73	183	TA	5.1000	D	0.48E+01	2
162.7047	56	140	BA	12.7650	D	0.62E+01	2
163.3570	92	235	U	7.04E+08	Y	0.51E+01	3
163.9316	54	131	XEM	11.9000	D	0.19E+01	1
163.9794	55	136	CS	13.1600	D	0.46E+01	2
165.8530	56	139	BA	1.41000	H	0.22E+02	1
165.8531	58	139	CE	137.6400	D	0.80E+02	1
165.9245	36	88	KR	2.84000	H	0.31E+01	2
166.4293	90	228	TH	1.9130	Y	0.11E+00	3
166.4294	90	232	THTH	1.40E+10	Y	0.11E+00	3
170.6772	12	27	MG	9.46200	M	0.79E+00	3
172.9045	41	98	NBM	51.30000	M	0.31E+01	2

PAGE 4

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
174.9815	54	139	XE	39.68000	S	0.19E+02	2
175.3571	21	48	SC	1.8208	D	0.75E+01	2
176.3161	51	125	SB	2.7590	Y	0.69E+01	2
176.6428	55	136	CS	13.1600	D	0.14E+02	2
177.2144	70	169	YB	32.0220	D	0.21E+02	2
179.3948	73	182	TA	115.0000	D	0.31E+01	3
181.0688	42	99	MO	2.7470	D	0.61E+01	3
184.0948	43	101	TC	14.20000	M	0.17E+01	3
185.7220	92	235	U	7.04E+08	Y	0.57E+02	2
186.1109	88	226	RA	1.60E+03	Y	0.33E+01	1
186.1111	88	226	RAPB	1.60E+03	Y	0.33E+01	3
186.1111	88	226	RABI	1.60E+03	Y	0.33E+01	3
190.3143	56	141	BA	18.27000	M	0.46E+02	1
191.3619	78	197	PT	18.30000	H	0.37E+01	1
191.3620	80	197	HG	2.6710	D	0.58E+00	3
191.9430	42	101	MO	14.60000	M	0.19E+02	2
192.3490	26	59	FE	44.5100	D	0.30E+01	2
195.9997	42	101	MO	14.60000	M	0.29E+01	2
196.3389	36	88	KR	2.84000	H	0.26E+02	2
197.0338	65	160	TB	72.3000	D	0.51E+01	2
197.1420	8	19	O	26.91000	S	0.96E+02	2
197.2817	36	89	KR	3.16000	M	0.18E+01	3
197.9581	70	169	YB	32.0220	D	0.35E+02	1
198.3530	73	182	TA	115.0000	D	0.15E+01	3
198.5959	32	75	GE	1.37967	H	0.12E+01	2
198.5960	34	75	SE	119.6400	D	0.15E+01	3
200.6369	52	131	TEM	1.2500	D	0.75E+01	2
202.1060	92	235	U	7.04E+08	Y	0.11E+01	3
202.4712	39	90	YM	3.19000	H	0.97E+02	1
203.5500	94	239	PU	2.41E+04	Y	0.57E-03	3
204.1136	41	95	NBM	3.6100	D	0.23E+01	3
205.3180	92	235	U	7.04E+08	Y	0.50E+01	3
205.7955	77	192	IR	73.8300	D	0.33E+01	3
208.3629	71	177	LU	6.7100	D	0.11E+02	1

209.3852	89	228	AC	6.13000	H	0.41E+01	2
209.3853	90	232	THAC	1.40E+10	Y	0.41E+01	2
209.7554	93	239	NP	2.3550	D	0.34E+01	2
209.8695	73	183	TA	5.1000	D	0.45E+01	2
215.6448	65	160	TB	72.3000	D	0.40E+01	2
215.7538	90	228	TH	1.9130	Y	0.26E+00	2
215.7539	90	232	THTH	1.40E+10	Y	0.26E+00	2
216.0480	56	131	BA	11.8000	D	0.20E+02	2
218.6504	54	139	XE	39.68000	S	0.52E+02	1
220.5121	53	135	I	6.55000	H	0.17E+01	2
220.9741	36	89	KR	3.16000	M	0.20E+02	2
221.4590	35	82	BR	1.4708	D	0.23E+01	3
222.1099	73	182	TA	115.0000	D	0.76E+01	2
225.4392	54	139	XE	39.68000	S	0.27E+01	3
227.7152	55	138	CS	32.20000	M	0.15E+01	3
228.1865	93	239	NP	2.3550	D	0.11E+02	2
228.2601	52	132	TE	3.2583	D	0.88E+02	1
229.3220	73	182	TA	115.0000	D	0.36E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
231.5658	56	142	BA	10.60000	M	0.12E+02
231.5753	58	143	CE	1.3750	D	0.20E+01
233.2071	54	133	XEM	2.1880	D	0.10E+02
234.4322	36	90	KR	32.32000	S	0.25E+01
235.4974	53	134	I	52.60000	M	0.22E+01
235.6800	41	95	NBM	3.6100	D	0.25E+02
238.6320	82	212	PB	10.64000	H	0.44E+02
238.6321	90	232	THPB	1.40E+10	Y	0.44E+02
239.6122	56	131	BA	11.8000	D	0.24E+01
240.9340	52	131	TEM	1.2500	D	0.76E+01
240.9870	88	224	RA	3.6600	D	0.41E+01
240.9871	90	232	THRA	1.40E+10	Y	0.41E+01
241.6350	38	92	SR	2.71000	H	0.30E+01
241.9192	82	214	PB	26.80000	M	0.74E+01
241.9193	88	226	RAPB	1.60E+03	Y	0.74E+01
242.1779	36	90	KR	32.32000	S	0.92E+01
242.6559	54	138	XE	14.08000	M	0.35E+01
244.2652	73	183	TA	5.1000	D	0.87E+01
244.6989	63	152	EU	13.5100	Y	0.75E+01
246.0609	73	183	TA	5.1000	D	0.28E+02
247.9678	63	154	EU	8.5880	Y	0.66E+01
249.3230	36	90	KR	32.32000	S	0.13E+01
249.4167	56	131	BA	11.8000	D	0.28E+01
249.7924	54	135	XE	9.10400	H	0.90E+02
252.7457	51	127	SB	3.8500	D	0.82E+01
254.2669	40	97	ZR	16.90000	H	0.13E+01
255.0702	50	113	SN	115.0900	D	0.19E+01
255.2269	56	142	BA	10.60000	M	0.21E+02
258.1807	91	234	PAM	4.47E+09	Y	0.81E-01
258.4458	54	138	XE	14.08000	M	0.32E+02
260.9080	48	115	CD	2.2275	D	0.19E+01
261.0788	70	169	YB	32.0220	D	0.19E+01
262.7488	53	132	I	2.28400	H	0.14E+01
262.8453	44	105	RU	4.44000	H	0.65E+01
264.0755	73	182	TA	115.0000	D	0.36E+01
264.6556	32	75	GE	1.37967	H	0.11E+02
264.6557	34	75	SE	119.6400	D	0.59E+02
266.9072	39	93	Y	10.25000	H	0.68E+01
270.2561	89	228	AC	6.13000	H	0.38E+01
270.2562	90	232	THAC	1.40E+10	Y	0.38E+01

PAGE 5

272.6467	37	89	RB	15.20000	M	0.14E+01	3
273.6848	55	136	CS	13.1600	D	0.13E+02	2
274.5627	38	91	SR	9.52000	H	0.10E+01	3
276.3877	56	133	BA	10.5700	Y	0.72E+01	2
276.9071	56	141	BA	18.27000	M	0.23E+02	2
277.2830	81	208	TL	3.05300	M	0.64E+01	2
277.2831	90	232	THTL	1.40E+10	Y	0.23E+01	2
277.6017	93	239	NP	2.3550	D	0.14E+02	1
278.5582	51	126	SB	12.4000	D	0.22E+01	3
278.5703	52	131	TEM	1.2500	D	0.18E+01	3
279.1967	80	203	HG	46.5950	D	0.81E+02	1
279.5381	34	75	SE	119.6400	D	0.25E+02	2

PAGE 6

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
282.5220	70	175	YB	4.1900	D	0.31E+01	2
284.2989	53	131	I	8.0210	D	0.62E+01	3
288.4619	53	135	I	6.55000	H	0.31E+01	2
289.8163	54	139	XE	39.68000	S	0.85E+01	2
290.9805	51	127	SB	3.8500	D	0.19E+01	2
291.7270	73	183	TA	5.1000	D	0.38E+01	2
293.2735	58	143	CE	1.3750	D	0.42E+02	1
293.5511	77	194	IR	19.15000	H	0.25E+01	2
295.0908	82	214	PB	26.80000	M	0.19E+02	2
295.0909	88	226	RAPB	1.60E+03	Y	0.19E+02	2
295.9393	68	171	ER	7.52000	H	0.29E+02	2
295.9582	77	192	IR	73.8300	D	0.29E+02	2
296.6051	54	139	XE	39.68000	S	0.20E+02	2
296.8904	51	126	SB	12.4000	D	0.49E+01	2
298.5756	65	160	TB	72.3000	D	0.27E+02	2
300.0336	82	212	PB	10.64000	H	0.33E+01	3
300.0337	90	232	THPB	1.40E+10	Y	0.33E+01	3
300.1750	91	233	PA	27.0000	D	0.66E+01	2
302.8513	56	133	BA	10.5700	Y	0.18E+02	2
303.9236	34	75	SE	119.6400	D	0.13E+01	3
304.1778	56	141	BA	18.27000	M	0.25E+02	2
304.8568	36	85	KRM	4.48000	H	0.14E+02	2
304.8671	56	140	BA	12.7650	D	0.43E+01	2
306.2778	45	105	RH	1.4733	D	0.51E+01	2
306.8311	43	101	TC	14.20000	M	0.88E+02	1
307.7382	70	169	YB	32.0220	D	0.11E+02	2
308.3306	68	171	ER	7.52000	H	0.64E+02	1
308.4568	77	192	IR	73.8300	D	0.30E+02	2
309.1371	56	142	BA	10.60000	M	0.26E+01	2
312.0117	91	233	PA	27.0000	D	0.39E+02	1
312.7460	19	42	K	12.35900	H	0.32E+00	3
313.1553	73	183	TA	5.1000	D	0.69E+01	2
315.8825	93	239	NP	2.3550	D	0.16E+01	3
316.4894	44	105	RU	4.44000	H	0.11E+02	2
316.5080	77	192	IR	73.8300	D	0.83E+02	1
319.2204	45	105	RH	1.4733	D	0.19E+02	1
319.4140	60	147	ND	10.9800	D	0.19E+01	3
320.0841	22	51	TI	5.76000	M	0.93E+02	1
320.0842	24	51	CR	27.7060	D	0.99E+01	1
328.0698	89	228	AC	6.13000	H	0.35E+01	2
328.0699	90	232	THAC	1.40E+10	Y	0.35E+01	2
328.4583	77	194	IR	19.15000	H	0.13E+02	1
328.7576	57	140	LA	1.6782	D	0.21E+02	2
332.8450	94	239	PU	2.41E+04	Y	0.49E-03	3
334.2768	52	131	TEM	1.2500	D	0.96E+01	2
334.3136	93	239	NP	2.3550	D	0.21E+01	3

335.3655	41	98	NBM	51.30000	M	0.11E+02	2
336.2585	48	115	CD	2.2275	D	0.50E+02	2
336.2585	49	115	INM	4.48600	H	0.46E+02	1
338.4199	89	228	AC	6.13000	H	0.12E+02	2
338.4200	90	232	THAC	1.40E+10	Y	0.12E+02	2
340.5896	91	233	PA	27.0000	D	0.45E+01	2

PAGE 7

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
340.6219	55	136	CS	13.1600	D	0.49E+02	2
343.3964	72	175	HF	70.0000	D	0.87E+02	1
343.6760	56	141	BA	18.27000	M	0.14E+02	2
344.2811	63	152	EU	13.5100	Y	0.27E+02	1
344.2859	63	152	EUM	9.32000	H	0.24E+01	2
345.0130	94	239	PU	2.41E+04	Y	0.56E-03	3
345.1641	36	89	KR	3.16000	M	0.12E+01	3
345.8287	72	181	HF	42.3900	D	0.15E+02	2
350.6550	58	143	CE	1.3750	D	0.34E+01	3
351.8664	82	214	PB	26.80000	M	0.36E+02	1
351.8665	88	226	RAPB	1.60E+03	Y	0.36E+02	1
353.9939	73	183	TA	5.1000	D	0.11E+02	2
355.4156	49	116	INM	54.15000	M	0.82E+00	3
355.4615	40	97	ZR	16.90000	H	0.23E+01	3
355.9992	56	133	BA	10.5700	Y	0.62E+02	1
356.1947	36	89	KR	3.16000	M	0.41E+01	2
360.3179	52	127	TE	9.35000	H	0.14E+00	2
362.2630	36	88	KR	2.84000	H	0.22E+01	2
363.8492	56	142	BA	10.60000	M	0.46E+01	2
364.4832	53	131	I	8.0210	D	0.82E+02	1
365.0518	52	131	TEM	1.2500	D	0.12E+01	3
366.3223	28	65	NI	2.52000	H	0.46E+01	2
366.4236	42	99	MO	2.7470	D	0.12E+01	3
369.4584	36	89	KR	3.16000	M	0.14E+01	3
373.1938	56	131	BA	11.8000	D	0.13E+02	2
375.0540	94	239	PU	2.41E+04	Y	0.16E-02	3
380.1910	94	239	PU	2.41E+04	Y	0.30E-03	3
380.4155	51	125	SB	2.7590	Y	0.15E+01	3
381.6000	39	94	Y	18.70000	M	0.20E+01	2
382.6980	94	239	PU	2.41E+04	Y	0.26E-03	3
383.8406	56	133	BA	10.5700	Y	0.89E+01	2
389.8273	56	141	BA	18.27000	M	0.13E+01	3
391.6898	50	113	SN	115.0900	D	0.65E+02	1
392.4948	65	160	TB	72.3000	D	0.13E+01	2
393.4020	44	105	RU	4.44000	H	0.37E+01	2
393.6316	54	139	XE	39.68000	S	0.62E+01	2
396.3278	70	175	YB	4.1900	D	0.65E+01	1
396.5595	54	138	XE	14.08000	M	0.63E+01	2
398.6635	91	233	PA	27.0000	D	0.14E+01	3
400.6568	34	75	SE	119.6400	D	0.12E+02	2
401.4551	54	138	XE	14.08000	M	0.22E+01	3
402.6372	36	87	KR	1.27183	H	0.50E+02	1
403.9879	56	131	BA	11.8000	D	0.13E+01	3
405.4623	53	134	I	52.60000	M	0.73E+01	2
408.9480	55	138	CS	32.20000	M	0.47E+01	2
409.6246	89	228	AC	6.13000	H	0.22E+01	2
409.6247	90	232	THAC	1.40E+10	Y	0.22E+01	2
411.1260	63	152	EU	13.5100	Y	0.22E+01	3
411.5917	36	89	KR	3.16000	M	0.25E+01	2
411.8044	79	198	AU	2.6943	D	0.96E+02	1
412.1878	51	127	SB	3.8500	D	0.37E+01	2
413.4927	44	105	RU	4.44000	H	0.22E+01	2

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
413.7130	94	239	PU	2.41E+04 Y	0.15E-02	1
414.6096	51	126	SB	12.4000 D	0.85E+02	2
415.9324	91	233	PA	27.0000 D	0.17E+01	3
416.9160	49	116	INM	54.15000 M	0.29E+02	2
417.6665	53	135	I	6.55000 H	0.35E+01	2
417.9268	52	127	TE	9.35000 H	0.10E+01	1
418.0133	53	130	I	12.36000 H	0.34E+02	2
423.7198	56	140	BA	12.7650 D	0.31E+01	2
425.0786	56	142	BA	10.60000 M	0.57E+01	2
427.8752	51	125	SB	2.7590 Y	0.29E+02	1
430.6625	38	92	SR	2.71000 H	0.33E+01	2
432.3586	39	95	Y	10.30000 M	0.20E+01	2
432.5231	57	140	LA	1.6782 D	0.29E+01	3
432.7523	72	175	HF	70.0000 D	0.16E+01	3
433.3716	53	134	I	52.60000 M	0.42E+01	2
433.4497	36	90	KR	32.32000 S	0.12E+01	2
433.9360	47	108	AGM	127.0000 Y	0.90E+02	2
434.3734	41	98	NBM	51.30000 M	0.12E+01	3
434.6144	54	138	XE	14.08000 M	0.20E+02	2
437.5849	56	140	BA	12.7650 D	0.19E+01	2
438.6340	30	69	ZNM	13.76000 H	0.95E+02	1
439.9097	60	147	ND	10.9800 D	0.12E+01	3
442.9172	53	128	I	24.99000 M	0.16E+02	1
443.9034	63	152	EU	13.5100 Y	0.31E+01	3
445.7084	51	127	SB	3.8500 D	0.42E+01	2
446.8110	47	110	AGM	249.7600 D	0.37E+01	3
448.5198	39	92	Y	3.54000 H	0.23E+01	2
452.3299	52	131	TE	25.00000 M	0.18E+02	2
452.3300	52	131	TEM	1.2500 D	0.15E+01	3
455.4591	54	137	XE	3.81800 M	0.31E+02	1
457.5394	56	141	BA	18.27000 M	0.48E+01	2
458.8935	53	134	I	52.60000 M	0.13E+01	3
459.5202	52	129	TE	1.16000 H	0.77E+01	1
462.1221	56	141	BA	18.27000 M	0.48E+01	2
462.7822	55	138	CS	32.20000 M	0.31E+02	2
462.9409	52	131	TEM	1.2500 D	0.18E+01	3
463.1049	89	228	AC	6.13000 H	0.46E+01	2
463.1050	90	232	THAC	1.40E+10 Y	0.46E+01	2
463.3292	49	116	INM	54.15000 M	0.83E+00	3
463.3670	51	125	SB	2.7590 Y	0.10E+02	2
467.2209	56	141	BA	18.27000 M	0.55E+01	2
468.0715	77	192	IR	73.8300 D	0.48E+02	2
469.3541	44	105	RU	4.44000 H	0.17E+02	2
473.6144	51	127	SB	3.8500 D	0.25E+02	2
475.3573	55	134	CS	2.0652 Y	0.15E+01	3
477.6050	4	7	BE	53.2000 D	0.10E+02	1
478.0130	75	188	RE	16.98000 H	0.10E+01	3
479.4911	39	90	YM	3.19000 H	0.91E+02	2
479.5337	74	187	W	23.90000 H	0.21E+02	1
481.9971	72	181	HF	42.3900 D	0.81E+02	1
484.5779	77	192	IR	73.8300 D	0.32E+01	3
486.4539	56	131	BA	11.8000 D	0.19E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
487.0262	57	140	LA	1.6782 D	0.44E+02	2
487.3109	52	129	TE	1.16000 H	0.14E+01	3
488.9096	53	134	I	52.60000 M	0.14E+01	3
489.2168	20	47	CA	4.5360 D	0.67E+01	2

490.3829	58 143 CE	1.3750 D	0.20E+01	3
491.4924	54 139 XE	39.68000 S	0.13E+01	3
492.3620	48 115 CD	2.2275 D	0.80E+01	2
492.6179	36 90 KR	32.32000 S	0.11E+01	3
492.6671	52 131 TE	25.00000 M	0.48E+01	3
496.2583	56 131 BA	11.8000 D	0.44E+02	1
497.0539	44 103 RU	39.2550 D	0.90E+02	1
497.7242	36 89 KR	3.16000 M	0.77E+01	2
499.2531	44 105 RU	4.44000 H	0.20E+01	2
505.9070	53 132 I	2.28400 H	0.50E+01	2
505.9655	42 101 MO	14.60000 M	0.12E+02	2
507.6987	40 97 ZR	16.90000 H	0.51E+01	2
514.0090	36 85 KR	10.7200 Y	0.43E+00	1
514.0091	38 85 SR	64.8490 D	0.99E+02	1
514.4314	53 134 I	52.60000 M	0.22E+01	3
520.4613	35 83 BR	2.39000 H	0.62E-01	2
522.6762	53 132 I	2.28400 H	0.16E+02	2
526.5628	53 135 I	6.55000 H	0.13E+02	2
526.5629	54 135 XEM	15.65000 M	0.81E+02	1
526.5750	53 128 I	24.99000 M	0.15E+01	3
527.9100	48 115 CD	2.2275 D	0.27E+02	1
529.6505	35 83 BR	2.39000 H	0.13E+01	1
529.8521	53 133 I	20.80000 H	0.86E+02	1
531.0134	60 147 ND	10.9800 D	0.13E+02	1
531.4449	43 101 TC	14.20000 M	0.10E+01	3
536.1009	53 130 I	12.36000 H	0.99E+02	1
537.3096	56 140 BA	12.7650 D	0.24E+02	1
539.1068	53 130 I	12.36000 H	0.14E+01	3
539.4855	36 90 KR	32.32000 S	0.29E+02	2
540.8681	53 134 I	52.60000 M	0.76E+01	2
543.7256	51 127 SB	3.8500 D	0.28E+01	2
545.0455	43 101 TC	14.20000 M	0.60E+01	3
546.5641	53 135 I	6.55000 H	0.71E+01	2
546.9872	55 138 CS	32.20000 M	0.11E+02	2
547.0200	53 132 I	2.28400 H	0.13E+01	3
550.9000	39 94 Y	18.70000 M	0.49E+01	2
551.5169	74 187 W	23.90000 H	0.49E+01	2
554.3500	35 82 BR	1.4708 D	0.71E+02	2
554.3762	36 90 KR	32.32000 S	0.47E+01	2
554.7978	51 126 SB	12.4000 D	0.18E+01	3
555.5648	38 91 SR	9.52000 H	0.61E+02	2
555.5649	39 91 YM	49.71000 M	0.95E+02	1
559.0797	33 76 AS	1.0967 D	0.45E+02	1
561.1139	39 92 Y	3.54000 H	0.24E+01	2
562.6522	89 228 AC	6.13000 H	0.10E+01	2
562.6523	90 232 THAC	1.40E+10 Y	0.10E+01	3
563.2370	55 134 CS	2.0652 Y	0.84E+01	3
563.2683	33 76 AS	1.0967 D	0.12E+01	3

ENERGY	Z	A NUCLIDE	HALF LIFE	BR	I-CODE
564.0900	51	122 SB	2.7000 D	0.71E+02	1
569.3207	55	134 CS	2.0652 Y	0.15E+02	2
573.1299	51	126 SB	12.4000 D	0.67E+01	2
577.1678	36	89 KR	3.16000 M	0.56E+01	2
581.5247	89	228 AC	6.13000 H	0.15E+01	2
581.5248	90	232 THAC	1.40E+10 Y	0.15E+01	3
583.1910	81	208 TL	3.05300 M	0.85E+02	2
583.1911	90	232 THTL	1.40E+10 Y	0.31E+02	2
584.9670	56	131 BA	11.8000 D	0.12E+01	3
585.9959	36	89 KR	3.16000 M	0.16E+02	1

586.1083	53	130	I	12.36000	H	0.17E+01	3
586.2961	52	131	TEM	1.2500	D	0.20E+01	3
588.5851	77	192	IR	73.8300	D	0.45E+01	3
589.6543	42	101	MO	14.60000	M	0.56E+01	2
590.9282	42	101	MO	14.60000	M	0.16E+02	2
591.8112	63	154	EU	8.5880	Y	0.48E+01	2
592.0648	76	185	OS	93.6000	D	0.13E+01	3
592.9862	51	126	SB	12.4000	D	0.75E+01	2
595.3820	53	134	I	52.60000	M	0.11E+02	2
599.5115	63	156	EU	15.1900	D	0.23E+01	2
600.5002	51	125	SB	2.7590	Y	0.18E+02	2
602.0459	52	131	TE	25.00000	M	0.42E+01	3
602.5672	40	97	ZR	16.90000	H	0.14E+01	3
602.7300	51	124	SB	60.2000	D	0.98E+02	1
604.0521	51	127	SB	3.8500	D	0.43E+01	2
604.4146	77	192	IR	73.8300	D	0.81E+01	3
604.7101	55	134	CS	2.0652	Y	0.98E+02	2
605.0319	35	84	BR	31.80000	M	0.17E+01	2
606.3300	35	82	BR	1.4708	D	0.13E+01	3
606.6332	51	125	SB	2.7590	Y	0.50E+01	2
608.1845	54	135	XE	9.10400	H	0.29E+01	3
608.5581	22	51	TI	5.76000	M	0.12E+01	3
609.3111	83	214	BI	19.90000	M	0.46E+02	1
609.3112	88	226	RABI	1.60E+03	Y	0.46E+02	1
610.2985	44	103	RU	39.2550	D	0.56E+01	3
612.4657	77	192	IR	73.8300	D	0.53E+01	3
613.1157	54	139	XE	39.68000	S	0.51E+01	3
614.2810	47	108	AGM	127.0000	Y	0.90E+02	2
618.3354	74	187	W	23.90000	H	0.61E+01	2
619.0703	36	90	KR	32.32000	S	0.10E+01	3
619.1100	35	82	BR	1.4708	D	0.43E+02	2
620.0352	56	131	BA	11.8000	D	0.16E+01	3
620.0663	38	91	SR	9.52000	H	0.18E+01	3
620.3600	47	110	AGM	249.7600	D	0.28E+01	3
621.1804	53	132	I	2.28400	H	0.16E+01	3
621.8187	53	134	I	52.60000	M	0.11E+02	2
621.9170	45	106	RH	1.0200	Y	0.98E+01	1
625.1650	56	141	BA	18.27000	M	0.33E+01	3
625.4727	74	187	W	23.90000	H	0.11E+01	2
627.3832	55	139	CS	9.27000	M	0.16E+01	2
627.9756	53	134	I	52.60000	M	0.22E+01	3
630.2711	53	132	I	2.28400	H	0.14E+02	2

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
633.0767	75	188	RE	16.98000	H	0.13E+01 3
635.8897	51	125	SB	2.7590	Y	0.11E+02 2
636.9747	53	131	I	8.0210	D	0.71E+01 3
641.2542	57	142	LA	1.54167	H	0.47E+02 2
644.9802	41	98	NBM	51.30000	M	0.56E+01 2
645.1631	77	194	IR	19.15000	H	0.12E+01 3
645.8550	51	124	SB	60.2000	D	0.74E+01 2
646.1100	76	185	OS	93.6000	D	0.81E+02 1
646.2847	63	156	EU	15.1900	D	0.70E+01 1
647.8530	56	141	BA	18.27000	M	0.56E+01 2
650.5872	53	132	I	2.28400	H	0.27E+01 3
652.8076	38	91	SR	9.52000	H	0.11E+02 2
654.2997	52	131	TE	25.00000	M	0.15E+01 3
656.2463	44	105	RU	4.44000	H	0.20E+01 2
656.3478	51	126	SB	12.4000	D	0.22E+01 3
657.0624	33	76	AS	1.0967	D	0.62E+01 2

PAGE 11

657.7620	47	110	AGM	249.7600	D	0.95E+02	1
657.8970	37	89	RB	15.20000	M	0.10E+02	2
658.2213	41	97	NB	1.20167	H	0.98E+02	1
658.8791	52	127	TEM	109.0000	D	0.13E-01	3
661.6600	55	137	CS	30.2500	Y	0.85E+02	1
664.5760	58	143	CE	1.3750	D	0.53E+01	2
665.1075	52	131	TEM	1.2500	D	0.43E+01	2
665.4420	83	214	BI	19.90000	M	0.16E+01	2
665.4421	88	226	RABI	1.60E+03	Y	0.16E+01	3
666.3698	51	126	SB	12.4000	D	0.10E+03	1
667.7339	53	132	I	2.28400	H	0.99E+02	2
668.5634	53	130	I	12.36000	H	0.96E+02	2
669.8646	53	132	I	2.28400	H	0.49E+01	2
671.3815	51	125	SB	2.7590	Y	0.18E+01	3
671.6200	53	132	I	2.28400	H	0.52E+01	2
673.9084	36	87	KR	1.27183	H	0.19E+01	3
674.6799	51	126	SB	12.4000	D	0.37E+01	2
676.3369	44	105	RU	4.44000	H	0.15E+02	2
677.3566	53	134	I	52.60000	M	0.78E+01	2
677.6230	47	110	AGM	249.7600	D	0.10E+02	2
685.7275	51	127	SB	3.8500	D	0.35E+02	1
685.7441	74	187	W	23.90000	H	0.26E+02	2
686.0191	53	130	I	12.36000	H	0.11E+01	3
687.0150	47	110	AGM	249.7600	D	0.64E+01	3
692.4855	63	154	EU	8.5880	Y	0.17E+01	3
692.7900	51	122	SB	2.7000	D	0.37E+01	3
694.9048	51	126	SB	12.4000	D	0.10E+03	2
695.6788	42	101	MO	14.60000	M	0.72E+01	2
695.8412	52	129	TEM	33.6000	D	0.30E+01	1
696.3801	36	89	KR	3.16000	M	0.18E+01	3
696.5100	59	144	PR	17.29000	M	0.13E+01	2
696.8638	51	126	SB	12.4000	D	0.32E+02	2
698.3680	35	82	BR	1.4708	D	0.28E+02	2
698.4531	51	127	SB	3.8500	D	0.35E+01	2
702.6450	41	94	NB	2.00E+04	Y	0.10E+03	2
703.7614	40	97	ZR	16.90000	H	0.93E+00	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
706.5551	53	133	I	20.80000	H	0.15E+01	3
706.6820	47	110	AGM	249.7600	D	0.16E+02	2
709.3248	51	124	SB	60.2000	D	0.13E+01	3
713.1110	42	101	MO	14.60000	M	0.34E+01	2
713.1304	52	131	TEM	1.2500	D	0.14E+01	3
713.7810	51	124	SB	60.2000	D	0.23E+01	3
713.9437	41	98	NBM	51.30000	M	0.86E+01	2
717.4179	76	185	OS	93.6000	D	0.41E+01	2
720.4111	51	126	SB	12.4000	D	0.54E+02	2
721.9574	58	143	CE	1.3750	D	0.51E+01	2
722.5526	51	127	SB	3.8500	D	0.18E+01	2
722.6986	41	98	NBM	51.30000	M	0.71E+02	2
722.7860	51	124	SB	60.2000	D	0.11E+02	2
722.8918	53	131	I	8.0210	D	0.18E+01	3
722.9290	47	108	AGM	127.0000	Y	0.91E+02	1
723.3563	63	154	EU	8.5880	Y	0.20E+02	2
723.4995	63	156	EU	15.1900	D	0.59E+01	2
723.9975	54	139	XE	39.68000	S	0.17E+01	3
724.1990	40	95	ZR	63.9800	D	0.44E+02	2
724.2661	44	105	RU	4.44000	H	0.47E+02	1
726.8862	53	132	I	2.28400	H	0.53E+01	3
727.2466	83	212	BI	2.5229	D	0.66E+01	1

727.2467	90	232	THBI	1.40E+10	Y	0.66E+01	1
728.7039	53	132	I	2.28400	H	0.11E+01	3
729.5334	52	129	TEM	33.6000	D	0.70E+00	2
730.7876	53	134	I	52.60000	M	0.18E+01	3
731.2928	36	90	KR	32.32000	S	0.14E+01	2
732.5092	54	139	XE	39.68000	S	0.16E+01	3
736.8399	35	84	BR	31.80000	M	0.13E+01	3
738.5132	36	89	KR	3.16000	M	0.42E+01	2
739.0282	56	141	BA	18.27000	M	0.43E+01	3
739.4824	53	130	I	12.36000	H	0.82E+02	2
739.5078	42	99	MO	2.7470	D	0.12E+02	2
743.3182	40	97	ZR	16.90000	H	0.93E+02	2
744.2170	52	131	TEM	1.2500	D	0.16E+01	3
744.2770	47	110	AGM	249.7600	D	0.47E+01	3
749.7239	38	91	SR	9.52000	H	0.24E+02	2
751.6608	57	140	LA	1.6782	D	0.44E+01	2
752.6000	39	94	Y	18.70000	M	0.14E+01	3
755.2773	89	228	AC	6.13000	H	0.13E+01	2
755.2774	90	232	THAC	1.40E+10	Y	0.13E+01	3
756.7290	40	95	ZR	63.9800	D	0.55E+02	1
756.8083	63	154	EU	8.5880	Y	0.43E+01	2
763.0576	81	208	TL	3.05300	M	0.19E+01	3
763.9440	47	110	AGM	249.7600	D	0.22E+02	2
765.2646	65	160	TB	72.3000	D	0.20E+01	2
765.8070	41	95	NB	34.9750	D	0.10E+03	1
766.4124	91	234	PAM	4.47E+09	Y	0.30E+00	2
766.7329	53	134	I	52.60000	M	0.41E+01	2
768.3499	83	214	BI	19.90000	M	0.49E+01	2
768.3500	88	226	RABI	1.60E+03	Y	0.49E+01	2
772.2775	89	228	AC	6.13000	H	0.11E+01	2

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
772.2776	90	232	THAC	1.40E+10	Y	0.11E+01
772.6824	53	132	I	2.28400	H	0.76E+02
772.8931	74	187	W	23.90000	H	0.40E+01
773.6761	52	131	TEM	1.2500	D	0.38E+02
776.5160	35	82	BR	1.4708	D	0.83E+02
776.6127	36	89	KR	3.16000	M	0.11E+01
777.9188	42	99	MO	2.7470	D	0.44E+01
778.9030	63	152	EU	13.5100	Y	0.13E+02
780.1202	53	132	I	2.28400	H	0.12E+01
782.4791	52	131	TEM	1.2500	D	0.78E+01
783.9791	51	127	SB	3.8500	D	0.15E+02
785.5095	83	212	BI	2.5229	D	0.11E+01
785.5096	90	232	THBI	1.40E+10	Y	0.11E+01
785.8274	82	214	PB	26.80000	M	0.10E+01
785.8275	88	226	RAPB	1.60E+03	Y	0.10E+01
787.3228	41	98	NBM	51.30000	M	0.93E+02
788.0963	54	139	XE	39.68000	S	0.31E+01
791.6619	41	98	NBM	51.30000	M	0.76E+01
793.7713	52	131	TEM	1.2500	D	0.14E+02
794.7914	89	228	AC	6.13000	H	0.46E+01
794.7916	90	232	THAC	1.40E+10	Y	0.46E+01
795.8674	55	134	CS	2.0652	Y	0.86E+02
801.9511	55	134	CS	2.0652	Y	0.87E+01
802.3930	35	84	BR	31.80000	M	0.59E+01
806.1550	83	214	BI	19.90000	M	0.12E+01
806.1552	88	226	RABI	1.60E+03	Y	0.12E+01
807.8496	20	47	CA	4.5360	D	0.69E+01
809.7676	53	132	I	2.28400	H	0.29E+01

810.7915	27	58	CO	70.8600	D	0.99E+02	1
811.8157	63	156	EU	15.1900	D	0.10E+02	2
812.2759	53	132	I	2.28400	H	0.56E+01	2
815.7825	57	140	LA	1.6782	D	0.24E+02	2
818.0310	47	110	AGM	249.7600	D	0.73E+01	3
818.5602	55	136	CS	13.1600	D	0.10E+03	2
818.7432	49	116	INM	54.15000	M	0.12E+02	2
822.7761	52	131	TEM	1.2500	D	0.61E+01	2
823.5009	41	98	NBM	51.30000	M	0.24E+01	3
827.8310	35	82	BR	1.4708	D	0.24E+02	2
831.6401	56	141	BA	18.27000	M	0.15E+01	3
833.5569	41	98	NBM	51.30000	M	0.10E+02	2
834.8430	25	54	MN	312.3000	D	0.10E+03	1
834.8587	36	88	KR	2.84000	H	0.13E+02	2
835.6032	89	228	AC	6.13000	H	0.17E+01	2
835.6033	90	232	THAC	1.40E+10	Y	0.17E+01	3
835.7230	36	89	KR	3.16000	M	0.11E+01	3
836.8210	53	135	I	6.55000	H	0.67E+01	2
840.2693	56	142	BA	10.60000	M	0.35E+01	2
841.5850	63	152	EUM	9.32000	H	0.15E+02	1
843.7573	12	27	MG	9.46200	M	0.73E+02	1
844.3140	39	92	Y	3.54000	H	0.13E+01	2
845.5031	36	87	KR	1.27183	H	0.73E+01	2
846.7640	27	56	CO	77.3500	D	0.10E+03	1

PAGE 14

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
846.8121	25	56	MN	2.57850	H	0.99E+02
847.0613	53	134	I	52.60000	M	0.95E+02
852.2351	52	131	TEM	1.2500	D	0.21E+02
856.2646	53	133	I	20.80000	H	0.12E+01
856.4606	51	126	SB	12.4000	D	0.18E+02
857.3149	53	134	I	52.60000	M	0.70E+01
860.5640	81	208	TL	3.05300	M	0.13E+02
860.5641	90	232	THTL	1.40E+10	Y	0.45E+01
861.6028	57	142	LA	1.54167	H	0.18E+01
863.9741	27	58	CO	70.8600	D	0.69E+00
867.1006	63	156	EU	15.1900	D	0.14E+01
867.2463	36	89	KR	3.16000	M	0.59E+01
867.3900	63	152	EU	13.5100	Y	0.42E+01
867.8306	57	140	LA	1.6782	D	0.56E+01
871.1190	41	94	NB	2.00E+04	Y	0.10E+03
871.7288	55	138	CS	32.20000	M	0.51E+01
873.2299	63	154	EU	8.5880	Y	0.11E+02
874.8088	76	185	OS	93.6000	D	0.66E+01
875.3145	53	133	I	20.80000	H	0.45E+01
875.8558	44	105	RU	4.44000	H	0.25E+01
876.1182	56	141	BA	18.27000	M	0.34E+01
877.2304	53	132	I	2.28400	H	0.11E+01
877.4238	42	101	MO	14.60000	M	0.34E+01
879.3671	65	160	TB	72.3000	D	0.30E+02
880.2637	76	185	OS	93.6000	D	0.50E+01
881.6857	35	84	BR	31.80000	M	0.41E+02
884.1265	53	134	I	52.60000	M	0.65E+02
884.6850	47	110	AGM	249.7600	D	0.73E+02
889.2770	21	46	SC	83.7900	D	0.10E+03
894.9255	57	142	LA	1.54167	H	0.85E+01
894.9811	56	142	BA	10.60000	M	0.13E+02
898.0420	39	88	Y	106.6300	D	0.94E+02
898.0646	37	88	RB	17.80000	M	0.15E+02
904.4658	36	89	KR	3.16000	M	0.71E+01

909.6913	41	98	NBM	51.30000	M	0.10E+01	3
909.9943	52	131	TEM	1.2500	D	0.33E+01	3
911.1572	89	228	AC	6.13000	H	0.29E+02	1
911.1573	90	232	THAC	1.40E+10	Y	0.29E+02	1
918.7400	39	94	Y	18.70000	M	0.56E+02	1
919.5476	57	140	LA	1.6782	D	0.27E+01	3
920.5714	52	131	TEM	1.2500	D	0.12E+01	3
925.1893	57	140	LA	1.6782	D	0.71E+01	2
925.7709	38	91	SR	9.52000	H	0.38E+01	2
928.6381	22	51	TI	5.76000	M	0.69E+01	3
933.8470	48	115	CDM	44.6000	D	0.20E+01	1
934.0389	83	214	BI	19.90000	M	0.32E+01	2
934.0390	88	226	RABI	1.60E+03	Y	0.32E+01	3
934.3082	42	101	MO	14.60000	M	0.34E+01	2
934.5264	39	92	Y	3.54000	H	0.14E+02	1
934.5265	41	92	NBM	10.1500	D	0.99E+02	1
937.4930	47	110	AGM	249.7600	D	0.34E+02	2
941.8035	36	90	KR	32.32000	S	0.12E+01	2

PAGE 15

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
944.3153	63	156	EU	15.1900	D	0.14E+01	2
947.1816	39	93	Y	10.25000	H	0.19E+01	2
947.8013	53	134	I	52.60000	M	0.40E+01	2
947.8834	37	89	RB	15.20000	M	0.92E+01	2
948.5482	52	131	TE	25.00000	M	0.23E+01	3
948.8912	56	142	BA	10.60000	M	0.10E+02	2
953.3985	38	92	SR	2.71000	H	0.35E+01	2
953.9274	51	126	SB	12.4000	D	0.15E+01	3
954.1294	39	95	Y	10.30000	M	0.19E+02	1
954.6213	53	132	I	2.28400	H	0.18E+02	2
960.5191	63	156	EU	15.1900	D	0.16E+01	2
962.2975	65	160	TB	72.3000	D	0.98E+01	2
963.3601	63	152	EUM	9.32000	H	0.12E+02	2
964.1102	63	152	EU	13.5100	Y	0.14E+02	2
964.6375	89	228	AC	6.13000	H	0.58E+01	2
964.6376	90	232	THAC	1.40E+10	Y	0.58E+01	2
966.1546	65	160	TB	72.3000	D	0.25E+02	2
968.2010	51	124	SB	60.2000	D	0.19E+01	3
968.9707	89	228	AC	6.13000	H	0.17E+02	2
968.9708	90	232	THAC	1.40E+10	Y	0.17E+02	2
969.4630	44	105	RU	4.44000	H	0.21E+01	2
972.6321	53	135	I	6.55000	H	0.12E+01	2
974.7084	53	134	I	52.60000	M	0.48E+01	2
977.4763	27	56	CO	77.3500	D	0.14E+01	3
983.5012	21	48	SC	1.8208	D	0.10E+03	2
985.8127	36	88	KR	2.84000	H	0.13E+01	2
989.1292	51	126	SB	12.4000	D	0.68E+01	2
993.0955	41	98	NBM	51.30000	M	0.11E+01	3
996.3293	63	154	EU	8.5880	Y	0.10E+02	2
996.4038	41	98	NBM	51.30000	M	0.20E+01	3
997.2487	52	131	TE	25.00000	M	0.33E+01	3
1000.9460	56	142	BA	10.60000	M	0.91E+01	2
1000.9970	91	234	PAM	4.47E+09	Y	0.84E+00	1
1001.7018	73	182	TA	115.0000	D	0.20E+01	3
1002.8687	65	160	TB	72.3000	D	0.10E+01	2
1004.7747	63	154	EU	8.5880	Y	0.18E+02	2
1007.5900	35	82	BR	1.4708	D	0.13E+01	3
1009.7675	55	138	CS	32.20000	M	0.30E+02	2
1011.2596	42	101	MO	14.60000	M	0.23E+01	2
1011.4744	57	142	LA	1.54167	H	0.39E+01	2

1012.5336	42	101	MO	14.60000	M	0.13E+02	1
1014.4288	12	27	MG	9.46200	M	0.29E+02	2
1016.0457	35	84	BR	31.80000	M	0.66E+01	2
1021.1875	40	97	ZR	16.90000	H	0.13E+01	3
1023.9477	41	98	NBM	51.30000	M	0.10E+01	3
1024.2842	38	91	SR	9.52000	H	0.33E+02	1
1024.7150	41	97	NB	1.20167	H	0.11E+01	3
1032.0773	37	89	RB	15.20000	M	0.58E+02	1
1034.7261	51	126	SB	12.4000	D	0.10E+01	3
1037.4961	21	48	SC	1.8208	D	0.98E+02	2
1037.8440	27	56	CO	77.3500	D	0.14E+02	2
1038.5920	55	134	CS	2.0652	Y	0.99E+00	3

PAGE 16

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
1038.7544	53	135	I	6.55000	H	0.79E+01	2
1039.3496	29	66	CU	5.10000	M	0.74E+01	1
1040.1323	53	134	I	52.60000	M	0.19E+01	3
1043.7206	57	142	LA	1.54167	H	0.27E+01	2
1044.0770	35	82	BR	1.4708	D	0.28E+02	2
1045.1310	51	124	SB	60.2000	D	0.19E+01	3
1047.5819	56	131	BA	11.8000	D	0.12E+01	3
1048.0934	55	136	CS	13.1600	D	0.80E+02	1
1048.5574	39	95	Y	10.30000	M	0.10E+01	2
1050.3599	45	106	RH	1.0200	Y	0.15E+01	2
1059.7098	52	131	TEM	1.2500	D	0.15E+01	3
1065.1561	63	156	EU	15.1900	D	0.52E+01	2
1072.5729	53	134	I	52.60000	M	0.15E+02	2
1076.6874	37	86	RB	18.6400	D	0.88E+01	1
1078.5168	56	142	BA	10.60000	M	0.11E+02	2
1079.2030	63	156	EU	15.1900	D	0.48E+01	2
1085.8420	63	152	EU	13.5100	Y	0.10E+02	2
1089.7670	63	152	EU	13.5100	Y	0.17E+01	3
1093.7137	56	142	BA	10.60000	M	0.26E+01	2
1097.2858	49	116	INM	54.15000	M	0.56E+02	2
1099.2510	26	59	FE	44.5100	D	0.56E+02	1
1101.5291	53	135	I	6.55000	H	0.16E+01	2
1107.9684	36	89	KR	3.16000	M	0.29E+01	2
1112.0870	63	152	EU	13.5100	Y	0.14E+02	2
1114.2732	54	138	XE	14.08000	M	0.15E+01	3
1115.1121	65	160	TB	72.3000	D	0.15E+01	2
1115.5179	28	65	NI	2.52000	H	0.15E+02	2
1115.5460	30	65	ZN	244.2600	D	0.51E+02	1
1116.7964	36	89	KR	3.16000	M	0.17E+01	3
1118.7194	36	90	KR	32.32000	S	0.36E+02	1
1120.2727	83	214	BI	19.90000	M	0.15E+02	2
1120.2737	88	226	RABI	1.60E+03	Y	0.15E+02	2
1120.5450	21	46	SC	83.7900	D	0.10E+03	2
1121.3010	73	182	TA	115.0000	D	0.35E+02	2
1123.9688	53	135	I	6.55000	H	0.36E+01	2
1125.4750	52	131	TEM	1.2500	D	0.11E+02	2
1129.6609	13	26	AL	7.20E+05	Y	0.25E+01	3
1131.5233	53	135	I	6.55000	H	0.23E+02	2
1136.1755	53	132	I	2.28400	H	0.30E+01	3
1136.2476	53	134	I	52.60000	M	0.92E+01	2
1138.9000	39	94	Y	18.70000	M	0.60E+01	2
1141.4043	36	88	KR	2.84000	H	0.13E+01	2
1142.4244	38	92	SR	2.71000	H	0.28E+01	2
1143.6133	53	132	I	2.28400	H	0.14E+01	3
1146.9641	52	131	TE	25.00000	M	0.50E+01	3
1147.2708	55	138	CS	32.20000	M	0.12E+01	3

1147.9438	40	97	ZR	16.90000	H	0.26E+01	1
1148.8224	52	131	TEM	1.2500	D	0.15E+01	3
1153.7537	63	156	EU	15.1900	D	0.12E+02	2
1155.1825	83	214	BI	19.90000	M	0.17E+01	2
1155.1835	88	226	RABI	1.60E+03	Y	0.17E+01	3
1157.4932	53	130	I	12.36000	H	0.11E+02	2

PAGE 17

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
1160.2694	57	142	LA	1.54167	H	0.18E+01 2
1161.0574	42	101	MO	14.60000	M	0.40E+01 2
1167.9443	55	134	CS	2.0652	Y	0.18E+01 3
1168.9194	41	98	NBM	51.30000	M	0.18E+02 2
1173.2380	27	60	CO	5.2719	Y	0.10E+03 2
1173.2606	53	132	I	2.28400	H	0.11E+01 3
1175.0990	27	56	CO	77.3500	D	0.23E+01 3
1175.4850	36	87	KR	1.27183	H	0.11E+01 3
1177.9409	65	160	TB	72.3000	D	0.15E+02 2
1189.0500	73	182	TA	115.0000	D	0.17E+02 2
1197.3574	56	141	BA	18.27000	M	0.46E+01 3
1199.9012	65	160	TB	72.3000	D	0.23E+01 2
1202.3353	56	142	BA	10.60000	M	0.62E+01 2
1204.1162	56	142	BA	10.60000	M	0.16E+02 2
1204.9567	39	91	Y	58.5100	D	0.30E+00 1
1206.6047	52	131	TEM	1.2500	D	0.97E+01 2
1212.7563	51	126	SB	12.4000	D	0.23E+01 3
1212.8492	21	48	SC	1.8208	D	0.24E+01 3
1212.9700	63	152	EU	13.5100	Y	0.14E+01 3
1213.0242	33	76	AS	1.0967	D	0.14E+01 2
1213.4058	35	84	BR	31.80000	M	0.25E+01 2
1216.1370	33	76	AS	1.0967	D	0.34E+01 2
1221.4080	73	182	TA	115.0000	D	0.27E+02 1
1228.6716	33	76	AS	1.0967	D	0.12E+01 3
1230.3926	41	98	NBM	51.30000	M	0.15E+01 3
1230.6866	63	156	EU	15.1900	D	0.88E+01 2
1231.0160	73	182	TA	115.0000	D	0.12E+02 2
1233.2196	57	142	LA	1.54167	H	0.18E+01 2
1235.4193	55	136	CS	13.1600	D	0.20E+02 2
1236.4042	53	133	I	20.80000	H	0.15E+01 3
1238.1073	83	214	BI	19.90000	M	0.59E+01 2
1238.1083	88	226	RABI	1.60E+03	Y	0.59E+01 2
1238.2870	27	56	CO	77.3500	D	0.67E+02 2
1242.4569	63	156	EU	15.1900	D	0.66E+01 2
1248.2798	37	89	RB	15.20000	M	0.43E+02 2
1250.7506	36	88	KR	2.84000	H	0.11E+01 2
1251.1622	42	101	MO	14.60000	M	0.46E+01 2
1257.4180	73	182	TA	115.0000	D	0.15E+01 3
1257.8702	41	98	NBM	51.30000	M	0.11E+01 3
1260.4200	53	135	I	6.55000	H	0.29E+02 1
1266.1678	14	31	SI	2.62167	H	0.70E-01 1
1271.8596	65	160	TB	72.3000	D	0.75E+01 2
1273.9202	36	89	KR	3.16000	M	0.14E+01 3
1274.5420	11	22	NA	2.6032	Y	0.10E+03 1
1274.5435	63	154	EU	8.5880	Y	0.35E+02 1
1276.1277	40	97	ZR	16.90000	H	0.97E+00 3
1277.4596	63	156	EU	15.1900	D	0.32E+01 2
1280.8546	38	91	SR	9.52000	H	0.93E+00 3
1280.9519	83	214	BI	19.90000	M	0.15E+01 2
1280.9529	88	226	RABI	1.60E+03	Y	0.15E+01 3
1283.3882	55	139	CS	9.27000	M	0.77E+01 1
1289.1560	73	182	TA	115.0000	D	0.14E+01 3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
1290.5994	48	115	CDM	44.6000 D	0.89E+00	2
1290.7915	53	132	I	2.28400 H	0.11E+01	3
1291.5959	26	59	FE	44.5100 D	0.44E+02	2
1293.5864	49	116	INM	54.15000 M	0.84E+02	1
1293.6400	18	41	AR	1.82667 H	0.99E+02	1
1295.3553	53	132	I	2.28400 H	0.20E+01	3
1297.0575	20	47	CA	4.5360 D	0.75E+02	1
1298.2097	53	133	I	20.80000 H	0.23E+01	3
1299.1520	63	152	EU	13.5100 Y	0.16E+01	3
1304.0348	42	101	MO	14.60000 M	0.28E+01	2
1312.0463	21	48	SC	1.8208 D	0.10E+03	1
1312.1444	65	160	TB	72.3000 D	0.29E+01	2
1317.4730	35	82	BR	1.4708 D	0.27E+02	2
1324.1309	39	95	Y	10.30000 M	0.53E+01	2
1324.5039	36	89	KR	3.16000 M	0.30E+01	2
1325.5120	51	124	SB	60.2000 D	0.16E+01	3
1332.5020	27	60	CO	5.2719 Y	0.10E+03	1
1335.5881	41	98	NBM	51.30000 M	0.13E+01	3
1343.5464	55	138	CS	32.20000 M	0.11E+01	3
1345.0065	54	139	XE	39.68000 S	0.11E+01	3
1345.7827	29	64	CU	12.70100 H	0.47E+00	1
1355.1771	51	124	SB	60.2000 D	0.10E+01	3
1356.8438	8	19	O	26.91000 S	0.50E+02	1
1360.2061	27	56	CO	77.3500 D	0.43E+01	2
1362.6510	40	97	ZR	16.90000 H	0.13E+01	3
1363.0129	57	142	LA	1.54167 H	0.21E+01	2
1365.1844	55	134	CS	2.0652 Y	0.30E+01	3
1366.4442	63	156	EU	15.1900 D	0.17E+01	2
1368.1639	51	124	SB	60.2000 D	0.27E+01	3
1368.6331	11	24	NA	14.96544 H	0.10E+03	2
1369.4159	36	88	KR	2.84000 H	0.15E+01	2
1372.0812	53	132	I	2.28400 H	0.25E+01	3
1377.6586	83	214	BI	19.90000 M	0.40E+01	2
1377.6595	88	226	RABI	1.60E+03 Y	0.40E+01	3
1379.9893	56	142	BA	10.60000 M	0.39E+01	2
1382.4690	37	88	RB	17.80000 M	0.78E+00	2
1384.0562	38	92	SR	2.71000 H	0.90E+02	1
1384.3000	47	110	AGM	249.7600 D	0.24E+02	2
1398.5646	53	132	I	2.28400 H	0.71E+01	2
1401.4832	83	214	BI	19.90000 M	0.14E+01	3
1401.4841	88	226	RABI	1.60E+03 Y	0.14E+01	3
1405.4224	39	92	Y	3.54000 H	0.48E+01	2
1407.9697	83	214	BI	19.90000 M	0.25E+01	2
1407.9707	88	226	RABI	1.60E+03 Y	0.25E+01	3
1408.0220	63	152	EU	13.5100 Y	0.21E+02	2
1413.5056	38	91	SR	9.52000 H	0.98E+00	3
1423.7694	36	90	KR	32.32000 S	0.27E+01	2
1432.2975	41	98	NBM	51.30000 M	0.54E+01	2
1434.0820	23	52	V	3.75000 M	0.10E+03	1
1435.7979	55	138	CS	32.20000 M	0.76E+02	1
1436.5630	51	124	SB	60.2000 D	0.13E+01	3
1436.6366	41	98	NBM	51.30000 M	0.25E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
1442.5428	53	132	I	2.28400 H	0.14E+01	3
1444.0852	8	19	O	26.91000 S	0.26E+01	3
1455.2560	53	134	I	52.60000 M	0.23E+01	3
1457.5729	53	135	I	6.55000 H	0.86E+01	2

1459.1932	89	228	AC	6.13000	H	0.11E+01	2
1459.1942	90	232	THAC	1.40E+10	Y	0.11E+01	3
1460.8300	19	40	K	1.26E+09	Y	0.11E+02	1
1463.6989	35	84	BR	31.80000	M	0.17E+01	2
1472.9863	36	89	KR	3.16000	M	0.68E+01	2
1474.8800	35	82	BR	1.4708	D	0.16E+02	2
1475.7880	47	110	AGM	249.7600	D	0.40E+01	3
1481.8999	28	65	NI	2.52000	H	0.24E+02	1
1489.1600	59	144	PR	17.29000	M	0.28E+00	2
1496.0311	89	228	AC	6.13000	H	0.10E+01	2
1496.0321	90	232	THAC	1.40E+10	Y	0.10E+01	3
1501.1827	36	89	KR	3.16000	M	0.13E+01	3
1502.7657	53	135	I	6.55000	H	0.11E+01	2
1505.0400	47	110	AGM	249.7600	D	0.13E+02	2
1507.6787	49	116	INM	54.15000	M	0.10E+02	2
1509.2189	83	214	BI	19.90000	M	0.22E+01	2
1509.2198	88	226	RABI	1.60E+03	Y	0.22E+01	3
1511.7538	41	98	NBM	51.30000	M	0.54E+01	2
1518.4169	36	88	KR	2.84000	H	0.22E+01	2
1524.6650	19	42	K	12.35900	H	0.18E+02	1
1529.7651	36	88	KR	2.84000	H	0.11E+02	2
1530.3679	36	89	KR	3.16000	M	0.33E+01	2
1532.5131	42	101	MO	14.60000	M	0.60E+01	2
1533.8975	36	89	KR	3.16000	M	0.51E+01	2
1537.8191	36	90	KR	32.32000	S	0.90E+01	2
1538.2642	37	89	RB	15.20000	M	0.26E+01	2
1541.0262	41	98	NBM	51.30000	M	0.21E+01	3
1545.9703	57	142	LA	1.54167	H	0.30E+01	2
1546.1930	41	98	NBM	51.30000	M	0.42E+01	2
1552.1633	36	90	KR	32.32000	S	0.20E+01	2
1553.9707	8	19	O	26.91000	S	0.14E+01	3
1562.3030	47	110	AGM	249.7600	D	0.10E+01	3
1566.3959	53	135	I	6.55000	H	0.13E+01	2
1575.4614	59	142	PR	19.13000	H	0.37E+01	1
1588.2271	89	228	AC	6.13000	H	0.36E+01	2
1588.2280	90	232	THAC	1.40E+10	Y	0.36E+01	2
1593.0000	63	154	EU	8.5880	Y	0.10E+01	3
1596.5431	57	140	LA	1.6782	D	0.95E+02	1
1596.5818	63	154	EU	8.5880	Y	0.18E+01	3
1613.7889	53	134	I	52.60000	M	0.43E+01	2
1618.3866	39	95	Y	10.30000	M	0.17E+01	2
1620.7350	83	212	BI	2.5229	D	0.15E+01	2
1620.7360	90	232	THBI	1.40E+10	Y	0.15E+01	2
1630.4746	89	228	AC	6.13000	H	0.19E+01	2
1630.4756	90	232	THAC	1.40E+10	Y	0.19E+01	3
1633.6021	9	20	F	11.03000	S	0.10E+03	1
1642.6902	17	38	CL	37.24000	M	0.31E+02	2
1646.0009	52	131	TEM	1.2500	D	0.12E+01	3

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
1658.1976	36	90	KR	32.32000	S	0.12E+01	2
1661.2577	83	214	BI	19.90000	M	0.11E+01	3
1661.2587	88	226	RABI	1.60E+03	Y	0.11E+01	3
1670.4485	54	139	XE	39.68000	S	0.10E+01	3
1671.4000	39	94	Y	18.70000	M	0.25E+01	2
1674.7526	27	58	CO	70.8600	D	0.52E+00	3
1678.0824	53	135	I	6.55000	H	0.95E+01	2
1682.1993	56	141	BA	18.27000	M	0.13E+01	3
1690.9800	51	124	SB	60.2000	D	0.47E+02	2
1693.9564	36	89	KR	3.16000	M	0.44E+01	2

1701.7430	41	98	NBM	51.30000	M	0.93E+01	2
1706.4868	53	135	I	6.55000	H	0.41E+01	2
1723.0354	57	142	LA	1.54167	H	0.15E+01	2
1729.5803	83	214	BI	19.90000	M	0.30E+01	2
1729.5813	88	226	RABI	1.60E+03	Y	0.30E+01	3
1740.6395	36	87	KR	1.27183	H	0.20E+01	3
1741.3845	35	84	BR	31.80000	M	0.16E+01	2
1741.4354	53	134	I	52.60000	M	0.27E+01	3
1750.5034	40	97	ZR	16.90000	H	0.13E+01	3
1752.8938	49	116	INM	54.15000	M	0.25E+01	3
1756.5225	57	142	LA	1.54167	H	0.30E+01	2
1764.4901	83	214	BI	19.90000	M	0.16E+02	2
1764.4911	88	226	RABI	1.60E+03	Y	0.16E+02	2
1768.3940	54	138	XE	14.08000	M	0.17E+02	2
1771.3500	27	56	CO	77.3500	D	0.16E+02	2
1778.9878	13	28	AL	2.24060	M	0.10E+03	1
1779.9926	36	90	KR	32.32000	S	0.62E+01	2
1791.2241	53	135	I	6.55000	H	0.77E+01	2
1805.9575	39	95	Y	10.30000	M	0.15E+01	2
1806.8589	53	134	I	52.60000	M	0.55E+01	2
1808.6484	13	26	AL	7.20E+05	Y	0.10E+03	1
1810.7985	25	56	MN	2.57850	H	0.27E+02	1
1836.0630	39	88	Y	106.6300	D	0.99E+02	1
1836.0770	37	88	RB	17.80000	M	0.22E+02	1
1847.4147	83	214	BI	19.90000	M	0.21E+01	2
1847.4156	88	226	RABI	1.60E+03	Y	0.21E+01	3
1850.8787	54	138	XE	14.08000	M	0.14E+01	3
1876.9659	63	156	EU	15.1900	D	0.17E+01	2
1877.3927	35	84	BR	31.80000	M	0.11E+01	3
1885.4548	41	98	NBM	51.30000	M	0.28E+01	3
1887.7219	52	131	TEM	1.2500	D	0.14E+01	3
1897.7200	35	84	BR	31.80000	M	0.13E+02	1
1901.4495	57	142	LA	1.54167	H	0.79E+01	2
1903.5444	36	89	KR	3.16000	M	0.10E+01	3
1917.8220	39	93	Y	10.25000	H	0.14E+01	2
1921.0560	53	132	I	2.28400	H	0.12E+01	3
1937.6345	63	156	EU	15.1900	D	0.21E+01	2
1940.3234	39	95	Y	10.30000	M	0.28E+01	2
1945.5189	41	98	NBM	51.30000	M	0.14E+01	3
1965.9502	63	156	EU	15.1900	D	0.41E+01	2
1980.5590	41	98	NBM	51.30000	M	0.33E+01	2
2001.0494	52	131	TEM	1.2500	D	0.20E+01	3

PAGE 21

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
2002.3453	53	132	I	2.28400	H	0.11E+01	3
2004.7804	54	138	XE	14.08000	M	0.54E+01	2
2007.7014	37	89	RB	15.20000	M	0.24E+01	2
2011.9063	36	87	KR	1.27183	H	0.29E+01	3
2012.4221	36	89	KR	3.16000	M	0.16E+01	3
2015.1790	27	56	CO	77.3500	D	0.30E+01	3
2015.9421	54	138	XE	14.08000	M	0.12E+02	2
2026.6188	63	156	EU	15.1900	D	0.35E+01	2
2029.6968	35	84	BR	31.80000	M	0.20E+01	2
2029.8834	36	88	KR	2.84000	H	0.45E+01	2
2032.1851	42	101	MO	14.60000	M	0.69E+01	2
2034.7590	27	56	CO	77.3500	D	0.78E+01	2
2035.4558	36	88	KR	2.84000	H	0.37E+01	2
2041.2540	42	101	MO	14.60000	M	0.21E+01	2
2055.1870	57	142	LA	1.54167	H	0.27E+01	2
2079.2937	54	138	XE	14.08000	M	0.14E+01	3

2090.9419	51	124	SB	60.2000	D	0.55E+01	2
2097.7800	63	156	EU	15.1900	D	0.42E+01	2
2112.3198	49	116	INM	54.15000	M	0.16E+02	2
2113.1863	25	56	MN	2.57850	H	0.14E+02	2
2118.5254	83	214	BI	19.90000	M	0.12E+01	3
2118.5264	88	226	RABI	1.60E+03	Y	0.12E+01	3
2127.5254	36	90	KR	32.32000	S	0.13E+01	2
2140.6001	39	94	Y	18.70000	M	0.95E+00	3
2167.6802	17	38	CL	37.24000	M	0.42E+02	1
2175.9556	39	95	Y	10.30000	M	0.82E+01	2
2180.9458	63	156	EU	15.1900	D	0.24E+01	2
2185.6621	59	144	PR	17.29000	M	0.70E+00	1
2186.7642	63	156	EU	15.1900	D	0.39E+01	2
2187.2170	57	142	LA	1.54167	H	0.53E+01	2
2195.8037	36	88	KR	2.84000	H	0.13E+02	2
2196.1489	37	89	RB	15.20000	M	0.13E+02	2
2200.5249	35	84	BR	31.80000	M	0.11E+01	3
2204.0906	83	214	BI	19.90000	M	0.50E+01	2
2204.0916	88	226	RABI	1.60E+03	Y	0.50E+01	2
2217.8428	55	138	CS	32.20000	M	0.15E+02	2
2231.7898	36	88	KR	2.84000	H	0.34E+01	2
2252.3279	54	138	XE	14.08000	M	0.23E+01	3
2269.9299	63	156	EU	15.1900	D	0.11E+01	2
2296.0435	39	95	Y	10.30000	M	0.14E+01	2
2392.1375	36	88	KR	2.84000	H	0.35E+02	1
2397.7681	57	142	LA	1.54167	H	0.15E+02	2
2408.6570	53	135	I	6.55000	H	0.95E+00	2
2447.6750	83	214	BI	19.90000	M	0.16E+01	2
2447.6760	88	226	RABI	1.60E+03	Y	0.16E+01	3
2484.2527	35	84	BR	31.80000	M	0.66E+01	2
2521.7209	57	140	LA	1.6782	D	0.34E+01	2
2522.9517	25	56	MN	2.57850	H	0.99E+00	3
2542.6946	57	142	LA	1.54167	H	0.10E+02	1
2554.9182	36	87	KR	1.27183	H	0.92E+01	2
2558.0820	36	87	KR	1.27183	H	0.39E+01	3
2570.3223	37	89	RB	15.20000	M	0.99E+01	2

PAGE 22

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
2598.4600	27	56	CO	77.3500	D	0.17E+02	2
2614.5330	81	208	TL	3.05300	M	0.10E+03	1
2614.5339	90	232	THTL	1.40E+10	Y	0.36E+02	1
2632.2083	39	95	Y	10.30000	M	0.51E+01	2
2639.4424	55	138	CS	32.20000	M	0.76E+01	2
2657.5811	25	56	MN	2.57850	H	0.65E+00	3
2666.8113	57	142	LA	1.54167	H	0.17E+01	2
2677.9158	37	88	RB	17.80000	M	0.20E+01	2
2707.3914	37	89	RB	15.20000	M	0.20E+01	2
2754.0300	11	24	NA	14.96544	H	0.10E+03	1
2824.2688	35	84	BR	31.80000	M	0.11E+01	3
2866.3772	36	89	KR	3.16000	M	0.17E+01	3
2971.9805	57	142	LA	1.54167	H	0.30E+01	2

A.7 Isotope Library GE\$ISOTOPE\$LIB:MASTER.LIB, Sorted by Nuclide

This listing was generated using the VAXGAP PRLIB command [2].

MASTER

PAGE 1

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
4	7	BE	53.2000 D	477.6050	1.04E+01	1
8	19	O	26.91000 S	109.8945	2.54E+00	3
				197.1420	9.59E+01	2
				1356.8438	5.03E+01	1
				1444.0852	2.64E+00	3
				1553.9707	1.39E+00	3
9	20	F	11.03000 S	1633.6021	1.00E+02	1
11	22	NA	2.6032 Y	1274.5420	9.99E+01	1
11	24	NA	14.96544 H	1368.6331	1.00E+02	2
				2754.0300	9.99E+01	1
12	27	MG	9.46200 M	170.6772	7.89E-01	3
				843.7573	7.30E+01	1
				1014.4288	2.91E+01	2
13	26	AL	7.20E+05 Y	1129.6609	2.47E+00	3
				1808.6484	9.98E+01	1
13	28	AL	2.24060 M	1778.9878	1.00E+02	1
14	31	SI	2.62167 H	1266.1678	7.00E-02	1
17	38	CL	37.24000 M	1642.6902	3.10E+01	2
				2167.6802	4.20E+01	1
18	41	AR	1.82667 H	1293.6400	9.92E+01	1
19	42	K	12.35900 H	312.7460	3.20E-01	3
				1524.6650	1.79E+01	1
19	40	K	1.26E+09 Y	1460.8300	1.07E+01	1
20	47	CA	4.5360 D	489.2168	6.74E+00	2
				807.8496	6.89E+00	2
				1297.0575	7.49E+01	1
21	47	SC	3.3510 D	159.3815	6.80E+01	1
21	48	SC	1.8208 D	175.3571	7.47E+00	2
				983.5012	1.00E+02	2
				1037.4961	9.75E+01	2
				1212.8492	2.38E+00	3
				1312.0463	1.00E+02	1
21	46	SC	83.7900 D	889.2770	1.00E+02	1

PAGE 2

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
21	46	SC	83.7900 D	1120.5450	1.00E+02	2
22	51	TI	5.76000 M	320.0841	9.30E+01	1
				608.5581	1.18E+00	3
				928.6381	6.88E+00	3
23	52	V	3.75000 M	1434.0820	1.00E+02	1
24	51	CR	27.7060 D	320.0842	9.85E+00	1
25	54	MN	312.3000 D	834.8430	1.00E+02	1
25	56	MN	2.57850 H	846.8121	9.89E+01	2
				1810.7985	2.72E+01	1
				2113.1863	1.43E+01	2
				2522.9517	9.90E-01	3
				2657.5811	6.53E-01	3
26	59	FE	44.5100 D	142.6520	9.80E-01	3
				192.3490	2.95E+00	2
				1099.2510	5.61E+01	1
				1291.5959	4.36E+01	2
27	57	CO	271.7900 D	122.0614	8.57E+01	1
				136.4743	1.07E+01	2
27	58	CO	70.8600 D	810.7915	9.94E+01	1
				863.9741	6.90E-01	3
				1674.7526	5.19E-01	3
27	56	CO	77.3500 D	846.7640	9.99E+01	1
				977.4763	1.40E+00	3
				1037.8440	1.41E+01	2
				1175.0990	2.26E+00	3
				1238.2870	6.69E+01	2
				1360.2061	4.29E+00	2
				1771.3500	1.55E+01	2
				2015.1790	3.03E+00	3
				2034.7590	7.78E+00	2
				2598.4600	1.67E+01	2
27	60	CO	5.2719 Y	1173.2380	9.99E+01	2
				1332.5020	1.00E+02	1
28	65	NI	2.52000 H	366.3223	4.61E+00	2
				1115.5179	1.48E+01	2
				1481.8999	2.35E+01	1
29	66	CU	5.10000 M	1039.3496	7.40E+00	1
29	64	CU	12.70100 H	1345.7827	4.70E-01	1
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
30	69	ZNM	13.76000 H	438.6340	9.48E+01	1
30	65	ZN	244.2600 D	1115.5460	5.08E+01	1

32	75	GE	1.37967	H	198.5959	1.17E+00	2
					264.6556	1.13E+01	1
33	76	AS	1.0967	D	559.0797	4.50E+01	1
					563.2683	1.20E+00	3
					657.0624	6.17E+00	2
					1213.0242	1.43E+00	2
					1216.1370	3.42E+00	2
					1228.6716	1.15E+00	3
34	75	SE	119.6400	D	66.0599	1.14E+00	3
					96.7337	3.48E+00	3
					121.1192	1.73E+01	2
					136.0017	5.90E+01	2
					198.5960	1.47E+00	3
					264.6557	5.91E+01	1
					279.5381	2.52E+01	2
					303.9236	1.34E+00	3
					400.6568	1.16E+01	2
35	82	BR	1.4708	D	221.4590	2.27E+00	3
					554.3500	7.06E+01	2
					606.3300	1.25E+00	3
					619.1100	4.33E+01	2
					698.3680	2.84E+01	2
					776.5160	8.34E+01	1
					827.8310	2.41E+01	2
					1007.5900	1.27E+00	3
					1044.0770	2.75E+01	2
					1317.4730	2.70E+01	2
					1474.8800	1.64E+01	2
35	83	BR	2.39000	H	520.4613	6.22E-02	2
					529.6505	1.30E+00	1
35	84	BR	31.80000	M	605.0319	1.72E+00	2
					736.8399	1.27E+00	3
					802.3930	5.90E+00	2
					881.6857	4.10E+01	2
					1016.0457	6.59E+00	2
					1213.4058	2.54E+00	2
					1463.6989	1.72E+00	2
					1741.3845	1.60E+00	2
					1877.3927	1.11E+00	3
					1897.7200	1.34E+01	1
					2029.6968	2.05E+00	2
Z	A	ID	HALF LIFE		ENERGY	BR	I-CODE
35	84	BR	31.80000	M	2200.5249	1.15E+00	3
					2484.2527	6.56E+00	2
					2824.2688	1.11E+00	3
36	90	KR	32.32000	S	121.7190	3.62E+01	2
					234.4322	2.46E+00	2
					242.1779	9.23E+00	2
					249.3230	1.27E+00	2
					433.4497	1.21E+00	2
					492.6179	1.12E+00	3
					539.4855	2.86E+01	2

			554.3762	4.71E+00	2	
			619.0703	1.01E+00	3	
			731.2928	1.38E+00	2	
			941.8035	1.24E+00	2	
			1118.7194	3.62E+01	1	
			1423.7694	2.73E+00	2	
			1537.8191	8.98E+00	2	
			1552.1633	2.04E+00	2	
			1658.1976	1.23E+00	2	
			1779.9926	6.23E+00	2	
			2127.5254	1.28E+00	2	
36	85 KRM	4.48000 H	151.1809	7.51E+01	1	
			304.8568	1.37E+01	2	
36	88 KR	2.84000 H	165.9245	3.10E+00	2	
			196.3389	2.60E+01	2	
			362.2630	2.25E+00	2	
			834.8587	1.30E+01	2	
			985.8127	1.32E+00	2	
			1141.4043	1.28E+00	2	
			1250.7506	1.12E+00	2	
			1369.4159	1.48E+00	2	
			1518.4169	2.15E+00	2	
			1529.7651	1.09E+01	2	
			2029.8834	4.53E+00	2	
			2035.4558	3.74E+00	2	
			2195.8037	1.32E+01	2	
			2231.7898	3.39E+00	2	
			2392.1375	3.46E+01	1	
36	89 KR	3.16000 M	197.2817	1.81E+00	3	
			220.9741	1.99E+01	2	
			345.1641	1.17E+00	3	
			356.1947	4.12E+00	2	
			369.4584	1.37E+00	3	
			411.5917	2.55E+00	2	
			497.7242	7.74E+00	2	
			577.1678	5.61E+00	2	
			585.9959	1.64E+01	1	
			696.3801	1.77E+00	3	
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
36	89 KR		3.16000 M	738.5132	4.18E+00	2
				776.6127	1.11E+00	3
				835.7230	1.10E+00	3
				867.2463	5.89E+00	2
				904.4658	7.14E+00	2
				1107.9684	2.90E+00	2
				1116.7964	1.65E+00	3
				1273.9202	1.35E+00	3
				1324.5039	3.05E+00	2
				1472.9863	6.85E+00	2
				1501.1827	1.31E+00	3
				1530.3679	3.30E+00	2
				1533.8975	5.09E+00	2
				1693.9564	4.36E+00	2
				1903.5444	1.03E+00	3
				2012.4221	1.55E+00	3

PAGE 5

			2866.3772	1.73E+00	3
36	87 KR	1.27183 H	402.6372	4.96E+01	1
			673.9084	1.89E+00	3
			845.5031	7.34E+00	2
			1175.4850	1.11E+00	3
			1740.6395	2.04E+00	3
			2011.9063	2.88E+00	3
			2554.9182	9.23E+00	2
			2558.0820	3.92E+00	3
36	85 KR	10.7200 Y	514.0090	4.34E-01	1
37	89 RB	15.20000 M	272.6467	1.41E+00	3
			657.8970	9.98E+00	2
			947.8834	9.22E+00	2
			1032.0773	5.80E+01	1
			1248.2798	4.26E+01	2
			1538.2642	2.55E+00	2
			2007.7014	2.38E+00	2
			2196.1489	1.33E+01	2
			2570.3223	9.86E+00	2
			2707.3914	2.03E+00	2
37	88 RB	17.80000 M	898.0646	1.47E+01	2
			1382.4690	7.80E-01	2
			1836.0770	2.24E+01	1
			2677.9158	2.05E+00	2
37	86 RB	18.6400 D	1076.6874	8.78E+00	1
38	92 SR	2.71000 H	241.6350	3.00E+00	2
			430.6625	3.30E+00	2
			953.3985	3.50E+00	2
			1142.4244	2.80E+00	2
			1384.0562	9.00E+01	1
38	91 SR	9.52000 H	274.5627	1.03E+00	3
Z	A	ID	HALF LIFE	ENERGY	BR
					I-CODE
38	91 SR	9.52000 H	555.5648	6.15E+01	2
			620.0663	1.77E+00	3
			652.8076	1.10E+01	2
			749.7239	2.36E+01	2
			925.7709	3.84E+00	2
			1024.2842	3.34E+01	1
			1280.8546	9.32E-01	3
			1413.5056	9.79E-01	3
38	85 SR	64.8490 D	514.0091	9.93E+01	1
39	90 YM	3.19000 H	202.4712	9.66E+01	1
			479.4911	9.07E+01	2
39	93 Y	10.25000 H	266.9072	6.80E+00	1
			947.1816	1.95E+00	2
			1917.8220	1.40E+00	2
39	94 Y	18.70000 M	381.6000	2.02E+00	2

				550.9000	4.93E+00	2
				752.6000	1.40E+00	3
				918.7400	5.60E+01	1
				1138.9000	5.99E+00	2
				1671.4000	2.46E+00	2
				2140.6001	9.52E-01	3
39	95	Y	10.30000 M	432.3586	2.00E+00	2
				954.1294	1.90E+01	1
				1048.5574	1.03E+00	2
				1324.1309	5.26E+00	2
				1618.3866	1.73E+00	2
				1805.9575	1.48E+00	2
				1940.3234	2.76E+00	2
				2175.9556	8.19E+00	2
				2296.0435	1.44E+00	2
				2632.2083	5.07E+00	2
39	92	Y	3.54000 H	448.5198	2.28E+00	2
				561.1139	2.39E+00	2
				844.3140	1.25E+00	2
				934.5264	1.39E+01	1
				1405.4224	4.80E+00	2
39	91	YM	49.71000 M	555.5649	9.49E+01	1
39	88	Y	106.6300 D	898.0420	9.41E+01	2
				1836.0630	9.94E+01	1
39	91	Y	58.5100 D	1204.9567	3.00E-01	1
40	97	ZR	16.90000 H	254.2669	1.25E+00	3
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
40	97	ZR	16.90000 H	355.4615	2.27E+00	3
				507.6987	5.05E+00	2
				602.5672	1.39E+00	3
				703.7614	9.28E-01	3
				743.3182	9.28E+01	2
				1021.1875	1.35E+00	3
				1147.9438	2.64E+00	1
				1276.1277	9.74E-01	3
				1362.6510	1.35E+00	3
				1750.5034	1.35E+00	3
40	95	ZR	63.9800 D	724.1990	4.42E+01	2
				756.7290	5.45E+01	1
41	93	NBM	16.1300 Y	30.4000	5.18E-04	1
41	98	NBM	51.30000 M	172.9045	3.07E+00	2
				335.3655	1.07E+01	2
				434.3734	1.21E+00	3
				644.9802	5.64E+00	2
				713.9437	8.65E+00	2
				722.6986	7.07E+01	2
				787.3228	9.30E+01	1
				791.6619	7.63E+00	2
				823.5009	2.42E+00	3

PAGE 7

			833.5569	1.02E+01	2
			909.6913	1.02E+00	3
			993.0955	1.12E+00	3
			996.4038	2.05E+00	3
			1023.9477	1.02E+00	3
			1168.9194	1.80E+01	2
			1230.3926	1.49E+00	3
			1257.8702	1.12E+00	3
			1335.5881	1.30E+00	3
			1432.2975	5.36E+00	2
			1436.6366	2.51E+00	3
			1511.7538	5.39E+00	2
			1541.0262	2.14E+00	3
			1546.1930	4.18E+00	2
			1701.7430	9.30E+00	2
			1885.4548	2.79E+00	3
			1945.5189	1.39E+00	3
			1980.5590	3.35E+00	2
41	95 NBM	3.6100 D	204.1136	2.28E+00	3
			235.6800	2.53E+01	1
41	97 NB	1.20167 H	658.2213	9.83E+01	1
			1024.7150	1.08E+00	3
41	94 NB	2.00E+04 Y	702.6450	9.98E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR
					I-CODE
41	94 NB	2.00E+04 Y	871.1190	9.99E+01	1
41	95 NB	34.9750 D	765.8070	9.98E+01	1
41	92 NBM	10.1500 D	934.5265	9.90E+01	1
42	101 MO	14.60000 M	80.9601	3.84E+00	2
			191.9430	1.88E+01	2
			195.9997	2.86E+00	2
			505.9655	1.18E+01	2
			589.6543	5.64E+00	2
			590.9282	1.64E+01	2
			695.6788	7.20E+00	2
			713.1110	3.38E+00	2
			877.4238	3.40E+00	2
			934.3082	3.40E+00	2
			1011.2596	2.26E+00	2
			1012.5336	1.28E+01	1
			1161.0574	3.97E+00	2
			1251.1622	4.61E+00	2
			1304.0348	2.78E+00	2
			1532.5131	5.96E+00	2
			2032.1851	6.94E+00	2
			2041.2540	2.11E+00	2
42	99 MO	2.7470 D	140.4740	9.10E+01	1
			181.0688	6.09E+00	3
			366.4236	1.19E+00	3
			739.5078	1.23E+01	2
			777.9188	4.37E+00	3

PAGE 8

43	101	TC	14.20000 M	127.2200	2.86E+00	3
				184.0948	1.68E+00	3
				306.8311	8.80E+01	1
				531.4449	1.02E+00	3
				545.0455	5.99E+00	3
43	99	TCM	6.00700 H	140.5110	8.90E+01	1
44	105	RU	4.44000 H	129.5966	5.60E+00	2
				149.1725	1.74E+00	2
				262.8453	6.49E+00	2
				316.4894	1.10E+01	2
				393.4020	3.73E+00	2
				413.4927	2.22E+00	2
				469.3541	1.73E+01	2
				499.2531	2.03E+00	2
				656.2463	2.03E+00	2
				676.3369	1.55E+01	2
				724.2661	4.67E+01	1
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
44	105	RU	4.44000 H	875.8558	2.47E+00	2
				969.4630	2.08E+00	2
44	103	RU	39.2550 D	497.0539	8.95E+01	1
				610.2985	5.64E+00	3
45	105	RHM	45.00000 S	129.5967	2.00E+01	1
45	105	RH	1.4733 D	306.2778	5.13E+00	2
				319.2204	1.90E+01	1
45	106	RH	1.0200 Y	621.9170	9.81E+00	1
				1050.3599	1.53E+00	2
46	109	PD	13.70000 H	88.0341	3.61E+00	1
47	108	AGM	127.0000 Y	79.1310	7.62E+00	2
				433.9360	9.05E+01	2
				614.2810	8.98E+01	2
				722.9290	9.08E+01	1
47	110	AGM	249.7600 D	446.8110	3.75E+00	3
				620.3600	2.81E+00	3
				657.7620	9.46E+01	1
				677.6230	1.04E+01	2
				687.0150	6.43E+00	3
				706.6820	1.64E+01	2
				744.2770	4.71E+00	3
				763.9440	2.23E+01	2
				818.0310	7.31E+00	3
				884.6850	7.27E+01	2
				937.4930	3.44E+01	2
				1384.3000	2.43E+01	2
				1475.7880	4.02E+00	3
				1505.0400	1.30E+01	2
				1562.3030	1.03E+00	3
48	109	CD	1.2666 Y	88.0342	3.65E+00	1

PAGE 9

48	115	CD	2.2275 D	260.9080	1.94E+00	3
				336.2585	5.01E+01	2
				492.3620	8.03E+00	2
				527.9100	2.75E+01	1
48	115	CDM	44.6000 D	933.8470	2.00E+00	1
				1290.5994	8.90E-01	2
49	116	INM	54.15000 M	138.3717	3.29E+00	2
				355.4156	8.18E-01	3
				416.9160	2.92E+01	2
				463.3292	8.27E-01	3
				818.7432	1.15E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
49	116	INM	54.15000 M	1097.2858	5.62E+01	2
				1293.5864	8.44E+01	1
				1507.6787	9.96E+00	2
				1752.8938	2.46E+00	3
				2112.3198	1.56E+01	2
49	115	INM	4.48600 H	336.2585	4.58E+01	1
50	117	SNM	13.6100 D	156.0217	2.11E+00	3
				158.5603	8.64E+01	1
50	113	SN	115.0900 D	255.0702	1.90E+00	3
				391.6898	6.49E+01	1
51	127	SB	3.8500 D	61.4268	1.38E+00	2
				252.7457	8.15E+00	2
				290.9805	1.94E+00	2
				412.1878	3.67E+00	2
				445.7084	4.16E+00	2
				473.6144	2.48E+01	2
				543.7256	2.82E+00	2
				604.0521	4.27E+00	2
				685.7275	3.53E+01	1
				698.4531	3.49E+00	2
				722.5526	1.80E+00	2
				783.9791	1.45E+01	2
51	125	SB	2.7590 Y	176.3161	6.90E+00	2
				380.4155	1.51E+00	3
				427.8752	2.94E+01	1
				463.3670	1.04E+01	2
				600.5002	1.78E+01	2
				606.6332	5.00E+00	2
				635.8897	1.13E+01	2
				671.3815	1.81E+00	3
51	126	SB	12.4000 D	278.5582	2.20E+00	3
				296.8904	4.90E+00	2
				414.6096	8.46E+01	2
				554.7978	1.80E+00	3
				573.1299	6.70E+00	2
				592.9862	7.50E+00	2
				656.3478	2.20E+00	3
				666.3698	9.97E+01	1

PAGE 10

674.6799	3.70E+00	2
694.9048	9.97E+01	2
696.8638	3.20E+01	2
720.4111	5.38E+01	2
856.4606	1.75E+01	2
953.9274	1.50E+00	3
989.1292	6.80E+00	2
1034.7261	1.00E+00	3

PAGE 11

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
51	126	SB	12.4000 D	1212.7563	2.30E+00	3
51	122	SB	2.7000 D	564.0900	7.06E+01	1
				692.7900	3.70E+00	3
51	124	SB	60.2000 D	602.7300	9.78E+01	1
				645.8550	7.38E+00	2
				709.3248	1.35E+00	3
				713.7810	2.28E+00	3
				722.7860	1.09E+01	2
				968.2010	1.87E+00	3
				1045.1310	1.87E+00	3
				1325.5120	1.59E+00	3
				1355.1771	1.05E+00	3
				1368.1639	2.66E+00	3
				1436.5630	1.25E+00	3
				1690.9800	4.71E+01	2
				2090.9419	5.49E+00	2
52	129	TE	1.16000 H	27.7908	1.63E+01	2
				459.5202	7.70E+00	1
				487.3109	1.42E+00	3
52	127	TEM	109.0000 D	57.6091	5.10E-01	1
				658.8791	1.26E-02	3
52	131	TEM	1.2500 D	81.1304	4.06E+00	2
				102.0643	7.93E+00	2
				149.7167	5.07E+00	2
				200.6369	7.54E+00	2
				240.9340	7.58E+00	2
				278.5703	1.78E+00	3
				334.2768	9.55E+00	2
				365.0518	1.20E+00	3
				452.3300	1.49E+00	3
				462.9409	1.82E+00	3
				586.2961	1.97E+00	3
				665.1075	4.33E+00	2
				713.1304	1.43E+00	3
				744.2170	1.59E+00	3
				773.6761	3.81E+01	2
				782.4791	7.77E+00	2
				793.7713	1.38E+01	2
				822.7761	6.11E+00	2
				852.2351	2.06E+01	1
				909.9943	3.29E+00	3
				920.5714	1.20E+00	3
				1059.7098	1.55E+00	3
				1125.4750	1.14E+01	2

1148.8224	1.51E+00	3
1206.6047	9.74E+00	2
1646.0009	1.24E+00	3

PAGE 12

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
52	131	TEM	1.2500 D	1887.7219	1.35E+00	3
				2001.0494	2.01E+00	3
52	132	TE	3.2583 D	111.8601	1.85E+00	3
				116.4001	1.94E+00	3
				228.2601	8.82E+01	1
52	131	TE	25.00000 M	149.7166	6.89E+01	1
				452.3299	1.82E+01	2
				492.6671	4.84E+00	3
				602.0459	4.20E+00	3
				654.2997	1.53E+00	3
				948.5482	2.26E+00	3
				997.2487	3.34E+00	3
				1146.9641	4.96E+00	3
52	123	TE	119.7000 D	158.9887	8.40E+01	1
52	127	TE	9.35000 H	360.3179	1.35E-01	2
				417.9268	1.00E+00	1
52	129	TEM	33.6000 D	695.8412	2.99E+00	1
				729.5334	6.97E-01	2
53	129	I	1.57E+07 Y	39.5710	7.46E+00	1
53	131	I	8.0210 D	80.1845	2.63E+00	3
				284.2989	6.20E+00	3
				364.4832	8.16E+01	1
				636.9747	7.12E+00	3
				722.8918	1.78E+00	3
53	134	I	52.60000 M	135.4063	4.10E+00	2
				235.4974	2.19E+00	3
				405.4623	7.35E+00	2
				433.3716	4.20E+00	2
				458.8935	1.34E+00	3
				488.9096	1.43E+00	3
				514.4314	2.21E+00	3
				540.8681	7.63E+00	2
				595.3820	1.12E+01	2
				621.8187	1.06E+01	2
				627.9756	2.20E+00	3
				677.3566	7.82E+00	2
				730.7876	1.81E+00	3
				766.7329	4.10E+00	2
				847.0613	9.54E+01	2
				857.3149	6.96E+00	2
				884.1265	6.49E+01	1
				947.8013	4.01E+00	2
				974.7084	4.77E+00	2
				1040.1323	1.91E+00	3

PAGE 13

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
---	---	----	-----------	--------	----	--------

53	134	I	52.60000 M	1072.5729	1.50E+01	2
				1136.2476	9.16E+00	2
				1455.2560	2.29E+00	3
				1613.7889	4.29E+00	2
				1741.4354	2.67E+00	3
				1806.8589	5.53E+00	2
53	135	I	6.55000 H	220.5121	1.75E+00	2
				288.4619	3.09E+00	2
				417.6665	3.52E+00	2
				526.5628	1.33E+01	2
				546.5641	7.13E+00	2
				836.8210	6.67E+00	2
				972.6321	1.20E+00	2
				1038.7544	7.93E+00	2
				1101.5291	1.60E+00	2
				1123.9688	3.61E+00	2
				1131.5233	2.25E+01	2
				1260.4200	2.86E+01	1
				1457.5729	8.65E+00	2
				1502.7657	1.07E+00	2
				1566.3959	1.29E+00	2
				1678.0824	9.53E+00	2
				1706.4868	4.09E+00	2
				1791.2241	7.70E+00	2
				2408.6570	9.53E-01	2
53	132	I	2.28400 H	262.7488	1.44E+00	3
				505.9070	5.02E+00	2
				522.6762	1.61E+01	2
				547.0200	1.25E+00	3
				621.1804	1.58E+00	3
				630.2711	1.38E+01	2
				650.5872	2.66E+00	3
				667.7339	9.87E+01	2
				669.8646	4.93E+00	2
				671.6200	5.23E+00	2
				726.8862	5.33E+00	3
				728.7039	1.09E+00	3
				772.6824	7.62E+01	1
				780.1202	1.23E+00	3
				809.7676	2.86E+00	3
				812.2759	5.63E+00	2
				877.2304	1.08E+00	3
				954.6213	1.81E+01	2
				1136.1755	3.02E+00	3
				1143.6133	1.38E+00	3
				1173.2606	1.09E+00	3
				1290.7915	1.13E+00	3
				1295.3553	1.97E+00	3
				1372.0812	2.47E+00	3
				1398.5646	7.11E+00	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
53	132	I	2.28400 H	1442.5428	1.42E+00	3
				1921.0560	1.18E+00	3
				2002.3453	1.09E+00	3
53	130	I	12.36000 H	418.0133	3.42E+01	2

				536.1009	9.90E+01	1
				539.1068	1.40E+00	3
				586.1083	1.69E+00	3
				668.5634	9.61E+01	2
				686.0191	1.07E+00	3
				739.4824	8.23E+01	2
				1157.4932	1.13E+01	2
53	128	I	24.99000 M	442.9172	1.58E+01	1
				526.5750	1.50E+00	3
53	133	I	20.80000 H	529.8521	8.63E+01	1
				706.5551	1.49E+00	3
				856.2646	1.23E+00	3
				875.3145	4.47E+00	2
				1236.4042	1.49E+00	3
				1298.2097	2.33E+00	3
54	133	XE	5.2440 D	80.9894	3.80E+01	1
54	138	XE	14.08000 M	153.9039	5.95E+00	2
				242.6559	3.50E+00	2
				258.4458	3.15E+01	1
				396.5595	6.30E+00	2
				401.4551	2.17E+00	3
				434.6144	2.03E+01	2
				1114.2732	1.47E+00	3
				1768.3940	1.67E+01	2
				1850.8787	1.42E+00	3
				2004.7804	5.36E+00	2
				2015.9421	1.23E+01	2
				2079.2937	1.44E+00	3
				2252.3279	2.29E+00	3
54	131	XEM	11.9000 D	163.9316	1.92E+00	1
54	139	XE	39.68000 S	174.9815	1.85E+01	2
				218.6504	5.20E+01	1
				225.4392	2.70E+00	3
				289.8163	8.53E+00	2
				296.6051	2.02E+01	2
				393.6316	6.24E+00	2
				491.4924	1.34E+00	3
				613.1157	5.10E+00	3
				723.9975	1.67E+00	3
				732.5092	1.63E+00	3
				788.0963	3.13E+00	3
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
54	139	XE	39.68000 S	1345.0065	1.06E+00	3
				1670.4485	1.04E+00	3
54	133	XEM	2.1880 D	233.2071	1.03E+01	1
54	135	XE	9.10400 H	249.7924	9.01E+01	1
				608.1845	2.90E+00	3
54	137	XE	3.81800 M	455.4591	3.12E+01	1

PAGE 15

54	135	XEM	15.65000 M	526.5629	8.12E+01	1
55	136	CS	13.1600 D	66.9373	1.25E+01	2
				86.3575	6.31E+00	2
				153.2948	7.47E+00	2
				163.9794	4.61E+00	2
				176.6428	1.36E+01	2
				273.6848	1.27E+01	2
				340.6219	4.85E+01	2
				818.5602	9.97E+01	2
				1048.0934	7.97E+01	1
				1235.4193	1.98E+01	2
55	138	CS	32.20000 M	138.0396	1.49E+00	3
				227.7152	1.51E+00	3
				408.9480	4.66E+00	2
				462.7822	3.08E+01	2
				546.9872	1.08E+01	2
				871.7288	5.12E+00	2
				1009.7675	2.98E+01	2
				1147.2708	1.24E+00	3
				1343.5464	1.14E+00	3
				1435.7979	7.63E+01	1
				2217.8428	1.52E+01	2
				2639.4424	7.63E+00	2
55	134	CS	2.0652 Y	475.3573	1.50E+00	3
				563.2370	8.38E+00	3
				569.3207	1.54E+01	2
				604.7101	9.76E+01	2
				795.8674	8.55E+01	1
				801.9511	8.70E+00	3
				1038.5920	9.91E-01	3
				1167.9443	1.79E+00	3
				1365.1844	3.02E+00	3
55	139	CS	9.27000 M	627.3832	1.65E+00	2
				1283.3882	7.70E+00	1
55	137	CS	30.2500 Y	661.6600	8.52E+01	1
56	133	BA	10.5700 Y	53.1480	2.20E+00	3
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
56	133	BA	10.5700 Y	79.6117	2.63E+00	3
				80.9895	3.41E+01	2
				276.3877	7.17E+00	2
				302.8513	1.83E+01	2
				355.9992	6.20E+01	1
				383.8406	8.93E+00	2
56	142	BA	10.60000 M	77.5714	1.11E+01	2
				231.5658	1.18E+01	2
				255.2269	2.06E+01	1
				309.1371	2.62E+00	2
				363.8492	4.57E+00	2
				425.0786	5.66E+00	2
				840.2693	3.50E+00	2
				894.9811	1.27E+01	2

PAGE 16

			948.8912	1.03E+01	2	
			1000.9460	9.06E+00	2	
			1078.5168	1.08E+01	2	
			1093.7137	2.55E+00	2	
			1202.3353	6.18E+00	2	
			1204.1162	1.58E+01	2	
			1379.9893	3.93E+00	2	
56	131 BA	11.8000 D	123.7774	2.90E+01	2	
			133.5819	2.19E+00	3	
			216.0480	1.99E+01	2	
			239.6122	2.41E+00	3	
			249.4167	2.81E+00	3	
			373.1938	1.33E+01	2	
			403.9879	1.29E+00	3	
			486.4539	1.89E+00	3	
			496.2583	4.38E+01	1	
			584.9670	1.23E+00	3	
			620.0352	1.57E+00	3	
			1047.5819	1.19E+00	3	
56	140 BA	12.7650 D	162.7047	6.20E+00	2	
			304.8671	4.32E+00	2	
			423.7198	3.12E+00	2	
			437.5849	1.92E+00	2	
			537.3096	2.40E+01	1	
56	139 BA	1.41000 H	165.8530	2.20E+01	1	
56	141 BA	18.27000 M	190.3143	4.60E+01	1	
			276.9071	2.33E+01	2	
			304.1778	2.52E+01	2	
			343.6760	1.42E+01	2	
			389.8273	1.32E+00	3	
			457.5394	4.78E+00	2	
			462.1221	4.78E+00	2	
			467.2209	5.47E+00	2	
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
56	141 BA	18.27000 M	625.1650	3.27E+00	3	
			647.8530	5.61E+00	2	
			739.0282	4.28E+00	3	
			831.6401	1.52E+00	3	
			876.1182	3.40E+00	3	
			1197.3574	4.60E+00	3	
			1682.1993	1.34E+00	3	
57	140 LA	1.6782 D	328.7576	2.05E+01	2	
			432.5231	2.87E+00	3	
			487.0262	4.38E+01	2	
			751.6608	4.42E+00	2	
			815.7825	2.37E+01	2	
			867.8306	5.61E+00	2	
			919.5476	2.68E+00	3	
			925.1893	7.06E+00	2	
			1596.5431	9.54E+01	1	
			2521.7209	3.43E+00	2	
57	142 LA	1.54167 H	641.2542	4.74E+01	2	

PAGE 17

			861.6028	1.80E+00	2	
			894.9255	8.48E+00	2	
			1011.4744	3.93E+00	2	
			1043.7206	2.75E+00	2	
			1160.2694	1.75E+00	2	
			1233.2196	1.85E+00	2	
			1363.0129	2.13E+00	2	
			1545.9703	2.99E+00	2	
			1723.0354	1.52E+00	2	
			1756.5225	2.99E+00	2	
			1901.4495	7.87E+00	2	
			2055.1870	2.65E+00	2	
			2187.2170	5.26E+00	2	
			2397.7681	1.48E+01	2	
			2542.6946	1.01E+01	1	
			2666.8113	1.71E+00	2	
			2971.9805	2.99E+00	2	
58	143	CE	1.3750 D	57.3817	1.18E+01	2
				231.5753	2.02E+00	3
				293.2735	4.20E+01	1
				350.6550	3.36E+00	3
				490.3829	1.97E+00	3
				664.5760	5.25E+00	2
				721.9574	5.12E+00	2
58	144	CE	285.0000 D	80.1034	1.55E+00	2
				133.5386	1.12E+01	1
58	141	CE	32.5000 D	145.4404	4.85E+01	1
58	139	CE	137.6400 D	165.8531	7.99E+01	1
59	144	PRM	7.20000 M	59.0315	7.94E-02	1
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
59	144	PR	17.29000 M	696.5100	1.34E+00	2
				1489.1600	2.79E-01	2
				2185.6621	7.00E-01	1
59	142	PR	19.13000 H	1575.4614	3.70E+00	1
60	147	ND	10.9800 D	91.1040	2.81E+01	2
				319.4140	1.91E+00	3
				439.9097	1.17E+00	3
				531.0134	1.27E+01	1
61	147	PM	2.6234 Y	121.2579	2.85E-03	1
62	153	SM	1.9458 D	69.6734	5.32E+00	2
				103.1806	2.83E+01	1
63	155	EU	4.8500 Y	60.0153	1.14E+00	3
				86.5423	3.26E+01	2
				105.3129	2.23E+01	1
63	156	EU	15.1900 D	88.9854	8.87E+00	2
				599.5115	2.26E+00	2
				646.2847	6.96E+00	1

PAGE 18

				723.4995	5.91E+00	2
				811.8157	1.02E+01	2
				867.1006	1.38E+00	2
				944.3153	1.37E+00	2
				960.5191	1.59E+00	2
				1065.1561	5.16E+00	2
				1079.2030	4.79E+00	2
				1153.7537	1.22E+01	2
				1230.6866	8.77E+00	2
				1242.4569	6.63E+00	2
				1277.4596	3.15E+00	2
				1366.4442	1.72E+00	2
				1876.9659	1.69E+00	2
				1937.6345	2.10E+00	2
				1965.9502	4.12E+00	2
				2026.6188	3.47E+00	2
				2097.7800	4.19E+00	2
				2180.9458	2.39E+00	2
				2186.7642	3.88E+00	2
				2269.9299	1.10E+00	2
63	152	EUM	9.32000 H	121.7758	7.21E+00	2
				344.2859	2.44E+00	2
				841.5850	1.46E+01	1
				963.3601	1.20E+01	2
63	152	EU	13.5100 Y	121.7824	2.84E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
63	152	EU	13.5100 Y	244.6989	7.53E+00	2
				344.2811	2.66E+01	1
				411.1260	2.24E+00	3
				443.9034	3.12E+00	3
				778.9030	1.30E+01	2
				867.3900	4.21E+00	2
				964.1102	1.45E+01	2
				1085.8420	1.01E+01	2
				1089.7670	1.73E+00	3
				1112.0870	1.35E+01	2
				1212.9700	1.41E+00	3
				1299.1520	1.63E+00	3
				1408.0220	2.09E+01	2
63	154	EU	8.5880 Y	123.1001	4.05E+01	2
				247.9678	6.60E+00	2
				591.8112	4.84E+00	2
				692.4855	1.69E+00	3
				723.3563	1.97E+01	2
				756.8083	4.34E+00	2
				873.2299	1.14E+01	2
				996.3293	1.03E+01	2
				1004.7747	1.79E+01	2
				1274.5435	3.55E+01	1
				1593.0000	1.03E+00	3
				1596.5818	1.83E+00	3
64	153	GD	241.6000 D	69.6734	2.30E+00	3
				97.4316	2.76E+01	1
				103.1807	1.96E+01	2

PAGE 19

65	160	TB	72.3000	D	86.7879	1.32E+01	2
					197.0338	5.15E+00	2
					215.6448	3.95E+00	2
					298.5756	2.69E+01	2
					392.4948	1.33E+00	2
					765.2646	2.00E+00	2
					879.3671	2.98E+01	1
					962.2975	9.84E+00	2
					966.1546	2.50E+01	2
					1002.8687	1.02E+00	2
					1115.1121	1.53E+00	2
					1177.9409	1.52E+01	2
					1199.9012	2.32E+00	2
					1271.8596	7.49E+00	2
					1312.1444	2.92E+00	2
66	165	DY	2.33400	H	94.7035	3.58E+00	1
68	171	ER	7.52000	H	111.6609	2.05E+01	2
					116.6894	2.30E+00	3
					124.0523	9.10E+00	2
Z	A	ID	HALF LIFE		ENERGY	BR	I-CODE
68	171	ER	7.52000	H	295.9393	2.89E+01	2
					308.3306	6.44E+01	1
69	170	TM	128.6000	D	84.2551	3.26E+00	1
70	169	YB	32.0220	D	63.1208	4.38E+01	2
					93.6151	2.66E+00	3
					109.7802	1.74E+01	2
					118.1901	1.88E+00	3
					130.5239	1.11E+01	2
					177.2144	2.15E+01	2
					197.9581	3.49E+01	1
					261.0788	1.90E+00	3
					307.7382	1.08E+01	2
70	175	YB	4.1900	D	113.8060	1.91E+00	2
					282.5220	3.06E+00	2
					396.3278	6.50E+00	1
71	177	LU	6.7100	D	112.9489	6.40E+00	2
					208.3629	1.10E+01	1
72	175	HF	70.0000	D	89.3561	2.35E+00	3
					343.3964	8.70E+01	1
					432.7523	1.57E+00	3
72	181	HF	42.3900	D	132.9411	4.33E+01	2
					136.1687	5.81E+00	3
					345.8287	1.51E+01	2
					481.9971	8.05E+01	1
73	182	TA	115.0000	D	65.7207	2.77E+00	3
					67.7500	4.12E+01	2
					84.6808	2.65E+00	3
					100.1065	1.40E+01	2
					113.6723	1.92E+00	3

PAGE 20

152.4308	7.18E+00	2
156.3874	2.73E+00	3
179.3948	3.14E+00	3
198.3530	1.52E+00	3
222.1099	7.59E+00	2
229.3220	3.63E+00	3
264.0755	3.63E+00	3
1001.7018	2.00E+00	3
1121.3010	3.48E+01	2
1189.0500	1.65E+01	2
1221.4080	2.73E+01	1
1231.0160	1.16E+01	2
1257.4180	1.52E+00	3
1289.1560	1.40E+00	3

73	183	TA	5.1000	D	84.7132	1.27E+00	3
Z	A	ID	HALF LIFE		ENERGY	BR	I-CODE

PAGE 21

73	183	TA	5.1000	D	99.0808	6.48E+00	2
					107.9332	1.08E+01	2
					144.1246	2.47E+00	3
					161.1445	1.17E+01	2
					162.3250	4.82E+00	2
					209.8695	4.55E+00	2
					244.2652	8.67E+00	2
					246.0609	2.80E+01	1
					291.7270	3.80E+00	2
					313.1553	6.94E+00	2
					353.9939	1.14E+01	2
74	187	W	23.90000	H	71.9834	1.08E+01	2
					134.2277	8.56E+00	2
					479.5337	2.11E+01	1
					551.5169	4.92E+00	2
					618.3354	6.07E+00	2
					625.4727	1.05E+00	2
					685.7441	2.64E+01	2
					772.8931	3.98E+00	2
75	188	RE	16.98000	H	155.0641	1.49E+01	1
					478.0130	1.04E+00	3
					633.0767	1.25E+00	3
76	191	OS	15.4000	D	129.4341	2.57E+01	1
76	185	OS	93.6000	D	592.0648	1.33E+00	3
					646.1100	8.10E+01	1
					717.4179	4.12E+00	2
					874.8088	6.61E+00	2
					880.2637	5.00E+00	2
77	192	IR	73.8300	D	205.7955	3.32E+00	3
					295.9582	2.87E+01	2
					308.4568	2.98E+01	2
					316.5080	8.30E+01	1
					468.0715	4.78E+01	2
					484.5779	3.17E+00	3
					588.5851	4.48E+00	3
					604.4146	8.09E+00	3

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
				612.4657	5.28E+00	3
77	194	IR	19.15000 H	293.5511	2.52E+00	2
				328.4583	1.30E+01	1
				645.1631	1.16E+00	3
78	197	PT	18.30000 H	77.3432	1.71E+01	2
				191.3619	3.71E+00	1
79	198	AU	2.6943 D	411.8044	9.56E+01	1
80	197	HG	2.6710 D	77.3433	1.88E+01	1
80	197	HG	2.6710 D	191.3620	5.80E-01	3
80	203	HG	46.5950 D	279.1967	8.15E+01	1
81	208	TL	3.05300 M	277.2830	6.40E+00	2
				583.1910	8.51E+01	2
				763.0576	1.89E+00	3
				860.5640	1.25E+01	2
				2614.5330	9.98E+01	1
82	210	PB	22.3000 Y	46.5200	4.06E+00	1
82	214	PB	26.80000 M	53.1716	1.12E+00	3
				241.9192	7.41E+00	2
				295.0908	1.87E+01	2
				351.8664	3.58E+01	1
				785.8274	1.05E+00	3
82	212	PB	10.64000 H	238.6320	4.35E+01	1
				300.0336	3.25E+00	3
83	214	BI	19.90000 M	609.3111	4.61E+01	1
				665.4420	1.56E+00	2
				768.3499	4.89E+00	2
				806.1550	1.23E+00	3
				934.0389	3.16E+00	2
				1120.2727	1.50E+01	2
				1155.1825	1.69E+00	2
				1238.1073	5.92E+00	2
				1280.9519	1.47E+00	2
				1377.6586	4.02E+00	2
				1401.4832	1.39E+00	3
				1407.9697	2.48E+00	2
				1509.2189	2.19E+00	2
				1661.2577	1.15E+00	3
				1729.5803	3.05E+00	2
				1764.4901	1.59E+01	2
				1847.4147	2.12E+00	2
				2118.5254	1.21E+00	3
				2204.0906	4.99E+00	2
				2447.6750	1.55E+00	2
83	212	BI	2.5229 D	727.2466	6.64E+00	1
				785.5095	1.10E+00	2
				1620.7350	1.49E+00	2

PAGE 22

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
88	226	RAPB	1.60E+03 Y	53.1717	1.12E+00	3
				186.1111	3.29E+00	3
				241.9193	7.41E+00	2
				295.0909	1.87E+01	2
88	226	RAPB	1.60E+03 Y	351.8665	3.58E+01	1
				785.8275	1.05E+00	3
88	226	RA	1.60E+03 Y	186.1109	3.29E+00	1
88	226	RABI	1.60E+03 Y	186.1111	3.29E+00	3
				609.3112	4.61E+01	1
				665.4421	1.56E+00	3
				768.3500	4.89E+00	2
				806.1552	1.23E+00	3
				934.0390	3.16E+00	3
				1120.2737	1.50E+01	2
				1155.1835	1.69E+00	3
				1238.1083	5.92E+00	2
				1280.9529	1.47E+00	3
				1377.6595	4.02E+00	3
				1401.4841	1.39E+00	3
				1407.9707	2.48E+00	3
				1509.2198	2.19E+00	3
				1661.2587	1.15E+00	3
				1729.5813	3.05E+00	3
				1764.4911	1.59E+01	2
				1847.4156	2.12E+00	3
				2118.5264	1.21E+00	3
				2204.0916	4.99E+00	2
				2447.6760	1.55E+00	3
88	224	RA	3.6600 D	240.9870	4.05E+00	1
89	228	AC	6.13000 H	99.5475	1.34E+00	2
				129.0348	2.93E+00	2
				209.3852	4.12E+00	2
				270.2561	3.77E+00	2
				328.0698	3.46E+00	2
				338.4199	1.24E+01	2
				409.6246	2.20E+00	2
				463.1049	4.56E+00	2
				562.6522	1.01E+00	2
				581.5247	1.50E+00	2
				755.2773	1.32E+00	2
				772.2775	1.09E+00	2
				794.7914	4.59E+00	2
				835.6032	1.71E+00	2
				911.1572	2.87E+01	1
				964.6375	5.77E+00	2
				968.9707	1.74E+01	2
				1459.1932	1.06E+00	2
				1496.0311	1.05E+00	2
				1588.2271	3.59E+00	2
				1630.4746	1.95E+00	2
90	232	TH	1.40E+10 Y	58.9921	1.90E-01	1

90	232	THPB	1.40E+10 Y	58.9922	1.90E-01	4
----	-----	------	------------	---------	----------	---

PAGE 24

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
---	---	----	-----------	--------	----	--------

90	232	THPB	1.40E+10 Y	74.8137	1.06E+01	8
				77.1073	1.77E+01	8
				87.1902	6.27E+00	8
				238.6321	4.35E+01	1
				300.0337	3.25E+00	3

90	232	THBI	1.40E+10 Y	58.9923	1.90E-01	3
				727.2467	6.64E+00	1
				785.5096	1.10E+00	2
				1620.7360	1.49E+00	2

90	232	THAC	1.40E+10 Y	58.9924	1.90E-01	3
				89.9547	3.42E+00	8
				93.3500	5.56E+00	8
				99.5476	1.34E+00	2
				105.3616	2.01E+00	8
				129.0349	2.93E+00	2
				209.3853	4.12E+00	2
				270.2562	3.77E+00	2
				328.0699	3.46E+00	2
				338.4200	1.24E+01	2
				409.6247	2.20E+00	2
				463.1050	4.56E+00	2
				562.6523	1.01E+00	3
				581.5248	1.50E+00	3
				755.2774	1.32E+00	3
				772.2776	1.09E+00	3
				794.7916	4.59E+00	2
				835.6033	1.71E+00	3
				911.1573	2.87E+01	1
				964.6376	5.77E+00	2
				968.9708	1.74E+01	2
				1459.1942	1.06E+00	3
				1496.0321	1.05E+00	3
				1588.2280	3.59E+00	2
				1630.4756	1.95E+00	3

90	232	THTL	1.40E+10 Y	58.9925	1.90E-01	3
				72.8035	7.68E-01	9
				74.9693	1.29E+00	8
				84.7893	4.56E-01	9
				277.2831	2.30E+00	2
				583.1911	3.06E+01	2
				860.5641	4.50E+00	2
				2614.5339	3.59E+01	1

90	232	THRA	1.40E+10 Y	58.9926	1.90E-01	3
				240.9871	4.05E+00	1

90	232	THTH	1.40E+10 Y	58.9927	1.90E-01	2
----	-----	------	------------	---------	----------	---

PAGE 25

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
---	---	----	-----------	--------	----	--------

90	232	THTH	1.40E+10 Y	84.2571	1.22E+00	1
				131.4969	1.27E-01	2
				166.4294	1.07E-01	3

			215.7539	2.60E-01	2	
90	234 TH	4.47E+09 Y	63.2878	3.81E+00	1	
			92.5640	5.41E+00	2	
90	228 TH	1.9130 Y	84.2570	1.22E+00	1	
			131.4968	1.27E-01	2	
			166.4293	1.07E-01	3	
			215.7538	2.60E-01	2	
91	233 PA	27.0000 D	75.3429	1.32E+00	3	
			86.6519	1.97E+00	3	
			300.1750	6.63E+00	2	
			312.0117	3.86E+01	1	
			340.5896	4.50E+00	2	
			398.6635	1.41E+00	3	
			415.9324	1.74E+00	3	
91	234 PAM	4.47E+09 Y	258.1807	8.10E-02	2	
			766.4124	2.96E-01	2	
			1000.9970	8.39E-01	1	
92	234 U	2.46E+05 Y	53.2306	1.23E-01	1	
92	232 U	69.8000 Y	57.8137	2.00E-01	1	
			129.0350	6.82E-02	2	
92	235 U	7.04E+08 Y	109.1784	1.54E+00	3	
			143.7680	1.10E+01	1	
			163.3570	5.08E+00	3	
			185.7220	5.72E+01	2	
			202.1060	1.08E+00	3	
			205.3180	5.01E+00	3	
93	239 NP	2.3550 D	61.4618	1.29E+00	3	
			106.1272	2.72E+01	2	
			209.7554	3.42E+00	2	
			228.1865	1.13E+01	2	
			277.6017	1.44E+01	1	
			315.8825	1.60E+00	3	
			334.3136	2.07E+00	3	
94	239 PU	2.41E+04 Y	129.2960	6.31E-03	2	
			144.2010	3.04E-04	3	
			203.5500	5.69E-04	3	
			332.8450	4.94E-04	3	
			345.0130	5.56E-04	3	
			375.0540	1.55E-03	3	
			380.1910	3.05E-04	3	
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
94	239 PU	2.41E+04 Y	382.6980	2.59E-04	3	
			413.7130	1.47E-03	1	
95	241 AM	432.7000 Y	59.5364	3.59E+01	1	

PAGE 26

A.8 Isotope Library GE\$ISOTOPE\$LIB:ENVLIB.LIB, Sorted by Energy

This listing was generated using the VAXGAP PRLIB command [2].

ENVLIB

PAGE 1

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
53.1480	56	133	BA	10.5700 Y	0.22E+01	3
53.1716	82	214	PB	26.80000 M	0.11E+01	3
53.1717	88	226	RAPB	1.60E+03 Y	0.11E+01	3
53.2306	92	234	U	2.46E+05 Y	0.12E+00	1
57.8137	92	232	U	69.8000 Y	0.20E+00	1
58.9921	90	232	TH	1.40E+10 Y	0.19E+00	1
58.9922	90	232	THPB	1.40E+10 Y	0.19E+00	4
58.9923	90	232	THBI	1.40E+10 Y	0.19E+00	3
58.9924	90	232	THAC	1.40E+10 Y	0.19E+00	3
58.9925	90	232	THTL	1.40E+10 Y	0.19E+00	3
58.9926	90	232	THRA	1.40E+10 Y	0.19E+00	3
58.9927	90	232	THTH	1.40E+10 Y	0.19E+00	2
59.5364	95	241	AM	432.7000 Y	0.36E+02	1
60.0153	63	155	EU	4.8500 Y	0.11E+01	3
63.2878	90	234	TH	4.47E+09 Y	0.38E+01	1
65.7207	73	182	TA	115.0000 D	0.28E+01	3
67.7500	73	182	TA	115.0000 D	0.41E+02	2
74.8137	88	226	RAPB	1.60E+03 Y	0.65E+01	8
74.8138	90	232	THPB	1.40E+10 Y	0.11E+02	8
74.9693	90	232	THTL	1.40E+10 Y	0.13E+01	8
77.1073	88	226	RAPB	1.60E+03 Y	0.11E+02	8
77.1074	90	232	THPB	1.40E+10 Y	0.18E+02	8
79.1310	47	108	AGM	127.0000 Y	0.76E+01	2
79.6117	56	133	BA	10.5700 Y	0.26E+01	3
80.1034	58	144	CE	285.0000 D	0.15E+01	2
80.1035	58	144	CEPR	285.0000 D	0.15E+01	2
80.9894	56	133	BA	10.5700 Y	0.34E+02	2
84.2570	90	228	TH	1.9130 Y	0.12E+01	1
84.2571	90	232	THTH	1.40E+10 Y	0.12E+01	1
84.6808	73	182	TA	115.0000 D	0.27E+01	3
86.5423	63	155	EU	4.8500 Y	0.33E+02	2
87.1902	88	226	RAPB	1.60E+03 Y	0.39E+01	8
87.1903	90	232	THPB	1.40E+10 Y	0.63E+01	8
89.9547	90	232	THAC	1.40E+10 Y	0.34E+01	8
92.5640	90	234	TH	4.47E+09 Y	0.54E+01	2
93.3500	90	232	THAC	1.40E+10 Y	0.56E+01	8
99.5475	89	228	AC	6.13000 H	0.13E+01	2
99.5476	90	232	THAC	1.40E+10 Y	0.13E+01	2
100.1065	73	182	TA	115.0000 D	0.14E+02	2
105.3129	63	155	EU	4.8500 Y	0.22E+02	1
105.3616	90	232	THAC	1.40E+10 Y	0.20E+01	8
109.1784	92	235	U	7.04E+08 Y	0.15E+01	3
113.6723	73	182	TA	115.0000 D	0.19E+01	3
120.9115	92	234	U	2.46E+05 Y	0.34E-01	2
121.7824	63	152	EU	13.5100 Y	0.28E+02	2
122.0614	27	57	CO	271.7900 D	0.86E+02	1
123.1001	63	154	EU	8.5880 Y	0.40E+02	2
129.0348	89	228	AC	6.13000 H	0.29E+01	2
129.0349	90	232	THAC	1.40E+10 Y	0.29E+01	2

129.0350	92 232 U	69.8000 Y	0.68E-01	2
129.2960	94 239 PU	2.41E+04 Y	0.63E-02	1
131.4968	90 228 TH	1.9130 Y	0.13E+00	2

PAGE 2

ENERGY	Z A NUCLIDE	HALF LIFE	BR	I-CODE
131.4969	90 232 THTH	1.40E+10 Y	0.13E+00	2
132.9411	72 181 HF	42.3900 D	0.43E+02	2
133.5386	58 144 CE	285.0000 D	0.11E+02	1
133.5387	58 144 CEPR	285.0000 D	0.11E+02	2
136.1687	72 181 HF	42.3900 D	0.58E+01	3
136.4743	27 57 CO	271.7900 D	0.11E+02	2
142.6520	26 59 FE	44.5100 D	0.98E+00	3
143.7680	92 235 U	7.04E+08 Y	0.11E+02	1
144.2010	94 239 PU	2.41E+04 Y	0.30E-03	3
145.4404	58 141 CE	32.5000 D	0.49E+02	1
152.4308	73 182 TA	115.0000 D	0.72E+01	2
156.3874	73 182 TA	115.0000 D	0.27E+01	3
163.3570	92 235 U	7.04E+08 Y	0.51E+01	3
166.4293	90 228 TH	1.9130 Y	0.11E+00	3
166.4294	90 232 THTH	1.40E+10 Y	0.11E+00	3
176.3161	51 125 SB	2.7590 Y	0.69E+01	2
179.3948	73 182 TA	115.0000 D	0.31E+01	3
185.7220	92 235 U	7.04E+08 Y	0.57E+02	2
186.1109	88 226 RA	1.60E+03 Y	0.33E+01	1
186.1111	88 226 RAPB	1.60E+03 Y	0.33E+01	3
186.1111	88 226 RABI	1.60E+03 Y	0.33E+01	3
192.3490	26 59 FE	44.5100 D	0.30E+01	2
198.3530	73 182 TA	115.0000 D	0.15E+01	3
202.1060	92 235 U	7.04E+08 Y	0.11E+01	3
203.5500	94 239 PU	2.41E+04 Y	0.57E-03	3
205.3180	92 235 U	7.04E+08 Y	0.50E+01	3
209.3852	89 228 AC	6.13000 H	0.41E+01	2
209.3853	90 232 THAC	1.40E+10 Y	0.41E+01	2
215.7538	90 228 TH	1.9130 Y	0.26E+00	2
215.7539	90 232 THTH	1.40E+10 Y	0.26E+00	2
222.1099	73 182 TA	115.0000 D	0.76E+01	2
229.3220	73 182 TA	115.0000 D	0.36E+01	3
238.6320	82 212 PB	10.64000 H	0.44E+02	1
238.6321	90 232 THPB	1.40E+10 Y	0.44E+02	1
240.9870	88 224 RA	3.6600 D	0.41E+01	1
240.9871	90 232 THRA	1.40E+10 Y	0.41E+01	1
241.9192	82 214 PB	26.80000 M	0.74E+01	2
241.9193	88 226 RAPB	1.60E+03 Y	0.74E+01	2
244.6989	63 152 EU	13.5100 Y	0.75E+01	2
247.9678	63 154 EU	8.5880 Y	0.66E+01	2
258.1807	91 234 PAM	4.47E+09 Y	0.81E-01	2
264.0755	73 182 TA	115.0000 D	0.36E+01	3
270.2561	89 228 AC	6.13000 H	0.38E+01	2
270.2562	90 232 THAC	1.40E+10 Y	0.38E+01	2
276.3877	56 133 BA	10.5700 Y	0.72E+01	2
277.2830	81 208 TL	3.05300 M	0.64E+01	2
277.2831	90 232 THTL	1.40E+10 Y	0.23E+01	2
279.1967	80 203 HG	46.5950 D	0.81E+02	1
295.0908	82 214 PB	26.80000 M	0.19E+02	2
295.0909	88 226 RAPB	1.60E+03 Y	0.19E+02	2
300.0336	82 212 PB	10.64000 H	0.33E+01	3
300.0337	90 232 THPB	1.40E+10 Y	0.33E+01	3

PAGE 3

ENERGY	Z A NUCLIDE	HALF LIFE	BR	I-CODE
302.8513	56 133 BA	10.5700 Y	0.18E+02	2

320.0842	24	51	CR	27.7060	D	0.99E+01	1
328.0698	89	228	AC	6.13000	H	0.35E+01	2
328.0699	90	232	THAC	1.40E+10	Y	0.35E+01	2
332.8450	94	239	PU	2.41E+04	Y	0.49E-03	3
338.4199	89	228	AC	6.13000	H	0.12E+02	2
338.4200	90	232	THAC	1.40E+10	Y	0.12E+02	2
344.2811	63	152	EU	13.5100	Y	0.27E+02	1
345.0130	94	239	PU	2.41E+04	Y	0.56E-03	3
345.8287	72	181	HF	42.3900	D	0.15E+02	2
351.8664	82	214	PB	26.80000	M	0.36E+02	1
351.8665	88	226	RAPB	1.60E+03	Y	0.36E+02	1
355.9992	56	133	BA	10.5700	Y	0.62E+02	1
375.0540	94	239	PU	2.41E+04	Y	0.16E-02	3
380.1910	94	239	PU	2.41E+04	Y	0.30E-03	3
380.4155	51	125	SB	2.7590	Y	0.15E+01	3
382.6980	94	239	PU	2.41E+04	Y	0.26E-03	3
383.8406	56	133	BA	10.5700	Y	0.89E+01	2
409.6246	89	228	AC	6.13000	H	0.22E+01	2
409.6247	90	232	THAC	1.40E+10	Y	0.22E+01	2
411.1260	63	152	EU	13.5100	Y	0.22E+01	3
413.7130	94	239	PU	2.41E+04	Y	0.15E-02	3
427.8752	51	125	SB	2.7590	Y	0.29E+02	1
433.9360	47	108	AGM	127.0000	Y	0.90E+02	2
443.9034	63	152	EU	13.5100	Y	0.31E+01	3
446.8110	47	110	AGM	249.7600	D	0.37E+01	3
463.1049	89	228	AC	6.13000	H	0.46E+01	2
463.1050	90	232	THAC	1.40E+10	Y	0.46E+01	2
463.3670	51	125	SB	2.7590	Y	0.10E+02	2
475.3573	55	134	CS	2.0652	Y	0.15E+01	3
477.6050	4	7	BE	53.2000	D	0.10E+02	1
481.9971	72	181	HF	42.3900	D	0.81E+02	1
497.0539	44	103	RU	39.2550	D	0.90E+02	1
562.6522	89	228	AC	6.13000	H	0.10E+01	2
562.6523	90	232	THAC	1.40E+10	Y	0.10E+01	3
563.2370	55	134	CS	2.0652	Y	0.84E+01	3
569.3207	55	134	CS	2.0652	Y	0.15E+02	2
581.5247	89	228	AC	6.13000	H	0.15E+01	2
581.5248	90	232	THAC	1.40E+10	Y	0.15E+01	3
583.1910	81	208	TL	3.05300	M	0.85E+02	2
583.1911	90	232	THTL	1.40E+10	Y	0.31E+02	2
591.8112	63	154	EU	8.5880	Y	0.48E+01	2
600.5002	51	125	SB	2.7590	Y	0.18E+02	2
602.7300	51	124	SB	60.2000	D	0.98E+02	1
604.7101	55	134	CS	2.0652	Y	0.98E+02	2
606.6332	51	125	SB	2.7590	Y	0.50E+01	2
609.3111	83	214	BI	19.90000	M	0.46E+02	1
609.3112	88	226	RABI	1.60E+03	Y	0.46E+02	1
610.2985	44	103	RU	39.2550	D	0.56E+01	3
614.2810	47	108	AGM	127.0000	Y	0.90E+02	1
620.3600	47	110	AGM	249.7600	D	0.28E+01	3
621.9170	45	106	RURH	1.0200	Y	0.98E+01	1

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
635.8897	51	125	SB	2.7590	Y	0.11E+02
645.8550	51	124	SB	60.2000	D	0.74E+01
657.7620	47	110	AGM	249.7600	D	0.95E+02
661.6600	55	137	CS	30.2500	Y	0.85E+02
665.4420	83	214	BI	19.90000	M	0.16E+01
665.4421	88	226	RABI	1.60E+03	Y	0.16E+01
671.3815	51	125	SB	2.7590	Y	0.18E+01

677.6230	47 110 AGM	249.7600 D	0.10E+02	2
687.0150	47 110 AGM	249.7600 D	0.64E+01	3
692.4855	63 154 EU	8.5880 Y	0.17E+01	3
696.5100	59 144 PR	17.29000 M	0.13E+01	2
696.5101	58 144 CEPR	285.0000 D	0.13E+01	2
702.6450	41 94 NB	2.00E+04 Y	0.10E+03	2
706.6820	47 110 AGM	249.7600 D	0.16E+02	2
709.3248	51 124 SB	60.2000 D	0.13E+01	3
713.7810	51 124 SB	60.2000 D	0.23E+01	3
722.7860	51 124 SB	60.2000 D	0.11E+02	2
722.9290	47 108 AGM	127.0000 Y	0.91E+02	2
723.3563	63 154 EU	8.5880 Y	0.20E+02	2
724.1990	40 95 ZR	63.9800 D	0.44E+02	2
724.1991	40 95 ZRNB	63.9800 D	0.44E+02	2
727.2466	83 212 BI	2.5229 D	0.66E+01	1
727.2467	90 232 THBI	1.40E+10 Y	0.66E+01	1
744.2770	47 110 AGM	249.7600 D	0.47E+01	3
755.2773	89 228 AC	6.13000 H	0.13E+01	2
755.2774	90 232 THAC	1.40E+10 Y	0.13E+01	3
756.7290	40 95 ZR	63.9800 D	0.55E+02	1
756.7291	40 95 ZRNB	63.9800 D	0.55E+02	2
756.8083	63 154 EU	8.5880 Y	0.43E+01	2
763.0576	81 208 TL	3.05300 M	0.19E+01	3
763.9440	47 110 AGM	249.7600 D	0.22E+02	2
765.8070	41 95 NB	34.9750 D	0.10E+03	1
765.8071	40 95 ZRNB	63.9800 D	0.22E+03	1
766.4124	91 234 PAM	4.47E+09 Y	0.30E+00	2
768.3499	83 214 BI	19.90000 M	0.49E+01	2
768.3500	88 226 RABI	1.60E+03 Y	0.49E+01	2
772.2775	89 228 AC	6.13000 H	0.11E+01	2
772.2776	90 232 THAC	1.40E+10 Y	0.11E+01	3
778.9030	63 152 EU	13.5100 Y	0.13E+02	2
785.5095	83 212 BI	2.5229 D	0.11E+01	2
785.5096	90 232 THBI	1.40E+10 Y	0.11E+01	2
785.8274	82 214 PB	26.80000 M	0.10E+01	3
785.8275	88 226 RAPB	1.60E+03 Y	0.10E+01	3
794.7914	89 228 AC	6.13000 H	0.46E+01	2
794.7916	90 232 THAC	1.40E+10 Y	0.46E+01	2
795.8674	55 134 CS	2.0652 Y	0.86E+02	1
801.9511	55 134 CS	2.0652 Y	0.87E+01	3
806.1550	83 214 BI	19.90000 M	0.12E+01	3
806.1552	88 226 RABI	1.60E+03 Y	0.12E+01	3
810.7915	27 58 CO	70.8600 D	0.99E+02	1
818.0310	47 110 AGM	249.7600 D	0.73E+01	3
834.8430	25 54 MN	312.3000 D	0.10E+03	1

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
835.6032	89	228	AC	6.13000 H	0.17E+01	2
835.6033	90	232	THAC	1.40E+10 Y	0.17E+01	3
860.5640	81	208	TL	3.05300 M	0.13E+02	2
860.5641	90	232	THTL	1.40E+10 Y	0.45E+01	2
863.9741	27	58	CO	70.8600 D	0.69E+00	3
867.3900	63	152	EU	13.5100 Y	0.42E+01	2
871.1190	41	94	NB	2.00E+04 Y	0.10E+03	1
873.2299	63	154	EU	8.5880 Y	0.11E+02	2
884.6850	47	110	AGM	249.7600 D	0.73E+02	2
889.2770	21	46	SC	83.7900 D	0.10E+03	1
911.1572	89	228	AC	6.13000 H	0.29E+02	1
911.1573	90	232	THAC	1.40E+10 Y	0.29E+02	1
934.0389	83	214	BI	19.90000 M	0.32E+01	2

934.0390	88	226	RABI	1.60E+03	Y	0.32E+01	3
937.4930	47	110	AGM	249.7600	D	0.34E+02	2
964.1102	63	152	EU	13.5100	Y	0.14E+02	2
964.6375	89	228	AC	6.13000	H	0.58E+01	2
964.6376	90	232	THAC	1.40E+10	Y	0.58E+01	2
968.2010	51	124	SB	60.2000	D	0.19E+01	3
968.9707	89	228	AC	6.13000	H	0.17E+02	2
968.9708	90	232	THAC	1.40E+10	Y	0.17E+02	2
996.3293	63	154	EU	8.5880	Y	0.10E+02	2
1000.9970	91	234	PAM	4.47E+09	Y	0.84E+00	1
1001.7018	73	182	TA	115.0000	D	0.20E+01	3
1004.7747	63	154	EU	8.5880	Y	0.18E+02	2
1038.5920	55	134	CS	2.0652	Y	0.99E+00	3
1045.1310	51	124	SB	60.2000	D	0.19E+01	3
1050.3599	45	106	RURH	2.9800	Y	0.15E+01	2
1085.8420	63	152	EU	13.5100	Y	0.10E+02	2
1089.7670	63	152	EU	13.5100	Y	0.17E+01	3
1099.2510	26	59	FE	44.5100	D	0.56E+02	1
1112.0870	63	152	EU	13.5100	Y	0.14E+02	2
1115.5460	30	65	ZN	244.2600	D	0.51E+02	1
1120.2727	83	214	BI	19.90000	M	0.15E+02	2
1120.2737	88	226	RABI	1.60E+03	Y	0.15E+02	2
1120.5450	21	46	SC	83.7900	D	0.10E+03	2
1121.3010	73	182	TA	115.0000	D	0.35E+02	2
1129.6609	13	26	AL	7.20E+05	Y	0.25E+01	3
1155.1825	83	214	BI	19.90000	M	0.17E+01	2
1155.1835	88	226	RABI	1.60E+03	Y	0.17E+01	3
1167.9443	55	134	CS	2.0652	Y	0.18E+01	3
1173.2380	27	60	CO	5.2719	Y	0.10E+03	2
1189.0500	73	182	TA	115.0000	D	0.17E+02	2
1212.9700	63	152	EU	13.5100	Y	0.14E+01	3
1221.4080	73	182	TA	115.0000	D	0.27E+02	1
1231.0160	73	182	TA	115.0000	D	0.12E+02	2
1238.1073	83	214	BI	19.90000	M	0.59E+01	2
1238.1083	88	226	RABI	1.60E+03	Y	0.59E+01	2
1257.4180	73	182	TA	115.0000	D	0.15E+01	3
1274.5420	11	22	NA	2.6032	Y	0.10E+03	1
1274.5435	63	154	EU	8.5880	Y	0.35E+02	1
1280.9519	83	214	BI	19.90000	M	0.15E+01	2

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE	
1280.9529	88	226	RABI	1.60E+03	Y	0.15E+01	3
1289.1560	73	182	TA	115.0000	D	0.14E+01	3
1291.5959	26	59	FE	44.5100	D	0.44E+02	2
1299.1520	63	152	EU	13.5100	Y	0.16E+01	3
1325.5120	51	124	SB	60.2000	D	0.16E+01	3
1332.5020	27	60	CO	5.2719	Y	0.10E+03	1
1355.1771	51	124	SB	60.2000	D	0.10E+01	3
1365.1844	55	134	CS	2.0652	Y	0.30E+01	3
1368.1639	51	124	SB	60.2000	D	0.27E+01	3
1377.6586	83	214	BI	19.90000	M	0.40E+01	2
1377.6595	88	226	RABI	1.60E+03	Y	0.40E+01	3
1384.3000	47	110	AGM	249.7600	D	0.24E+02	2
1401.4832	83	214	BI	19.90000	M	0.14E+01	3
1401.4841	88	226	RABI	1.60E+03	Y	0.14E+01	3
1407.9697	83	214	BI	19.90000	M	0.25E+01	2
1407.9707	88	226	RABI	1.60E+03	Y	0.25E+01	3
1408.0220	63	152	EU	13.5100	Y	0.21E+02	2
1436.5630	51	124	SB	60.2000	D	0.13E+01	3
1459.1932	89	228	AC	6.13000	H	0.11E+01	2

1459.1942	90	232	THAC	1.40E+10	Y	0.11E+01	3
1460.8300	19	40	K	1.26E+09	Y	0.11E+02	1
1475.7880	47	110	AGM	249.7600	D	0.40E+01	3
1489.1600	59	144	PR	17.29000	M	0.28E+00	2
1489.1610	58	144	CEPR	285.0000	D	0.28E+00	3
1496.0311	89	228	AC	6.13000	H	0.10E+01	2
1496.0321	90	232	THAC	1.40E+10	Y	0.10E+01	3
1505.0400	47	110	AGM	249.7600	D	0.13E+02	2
1509.2189	83	214	BI	19.90000	M	0.22E+01	2
1509.2198	88	226	RABI	1.60E+03	Y	0.22E+01	3
1562.3030	47	110	AGM	249.7600	D	0.10E+01	3
1588.2271	89	228	AC	6.13000	H	0.36E+01	2
1588.2280	90	232	THAC	1.40E+10	Y	0.36E+01	2
1593.0000	63	154	EU	8.5880	Y	0.10E+01	3
1596.5818	63	154	EU	8.5880	Y	0.18E+01	3
1620.7350	83	212	BI	2.5229	D	0.15E+01	2
1620.7360	90	232	THBI	1.40E+10	Y	0.15E+01	2
1630.4746	89	228	AC	6.13000	H	0.19E+01	2
1630.4756	90	232	THAC	1.40E+10	Y	0.19E+01	3
1661.2577	83	214	BI	19.90000	M	0.11E+01	3
1661.2587	88	226	RABI	1.60E+03	Y	0.11E+01	3
1674.7526	27	58	CO	70.8600	D	0.52E+00	3
1690.9800	51	124	SB	60.2000	D	0.47E+02	2
1729.5803	83	214	BI	19.90000	M	0.30E+01	2
1729.5813	88	226	RABI	1.60E+03	Y	0.30E+01	3
1764.4901	83	214	BI	19.90000	M	0.16E+02	2
1764.4911	88	226	RABI	1.60E+03	Y	0.16E+02	2
1808.6484	13	26	AL	7.20E+05	Y	0.10E+03	1
1847.4147	83	214	BI	19.90000	M	0.21E+01	2
1847.4156	88	226	RABI	1.60E+03	Y	0.21E+01	3
2090.9419	51	124	SB	60.2000	D	0.55E+01	2
2118.5254	83	214	BI	19.90000	M	0.12E+01	3
2118.5264	88	226	RABI	1.60E+03	Y	0.12E+01	3

PAGE 7

ENERGY	Z	A	NUCLIDE	HALF LIFE	BR	I-CODE
2185.6621	59	144	PR	17.29000	M	0.70E+00
2185.6631	58	144	CEPR	285.0000	D	0.70E+00
2204.0906	83	214	BI	19.90000	M	0.50E+01
2204.0916	88	226	RABI	1.60E+03	Y	0.50E+01
2223.5000	2	23	QCWA	1.00E+10	Y	0.10E+03
2447.6750	83	214	BI	19.90000	M	0.16E+01
2447.6760	88	226	RABI	1.60E+03	Y	0.16E+01
2614.5330	81	208	TL	3.05300	M	0.10E+03
2614.5339	90	232	THTL	1.40E+10	Y	0.36E+02

A.9 Isotope Library GE\$ISOTOPE\$LIB:ENVLIB.LIB, Sorted by Nuclide

This listing was generated using the VAXGAP PRLIB command [2].

ENVLIB

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
---	---	----	-----------	--------	----	--------

PAGE 1

2	23	QCWA	1.00E+10 Y	2223.5000	1.00E+02	1
4	7	BE	53.2000 D	477.6050	1.04E+01	1
11	22	NA	2.6032 Y	1274.5420	9.99E+01	1
13	26	AL	7.20E+05 Y	1129.6609 1808.6484	2.47E+00 9.98E+01	3 1
19	40	K	1.26E+09 Y	1460.8300	1.07E+01	1
21	46	SC	83.7900 D	889.2770 1120.5450	1.00E+02 1.00E+02	1 2
24	51	CR	27.7060 D	320.0842	9.85E+00	1
25	54	MN	312.3000 D	834.8430	1.00E+02	1
26	59	FE	44.5100 D	142.6520 192.3490 1099.2510 1291.5959	9.80E-01 2.95E+00 5.61E+01 4.36E+01	3 2 1 2
27	57	CO	271.7900 D	122.0614 136.4743	8.57E+01 1.07E+01	1 2
27	58	CO	70.8600 D	810.7915 863.9741 1674.7526	9.94E+01 6.90E-01 5.19E-01	1 3 3
27	60	CO	5.2719 Y	1173.2380 1332.5020	9.99E+01 1.00E+02	2 1
30	65	ZN	244.2600 D	1115.5460	5.08E+01	1
40	95	ZR	63.9800 D	724.1990 756.7290	4.42E+01 5.45E+01	2 1
40	95	ZRNB	63.9800 D	724.1991 756.7291 765.8071	4.42E+01 5.45E+01 2.20E+02	2 2 1
41	94	NB	2.00E+04 Y	702.6450 871.1190	9.98E+01 9.99E+01	2 1
41	95	NB	34.9750 D	765.8070	9.98E+01	1
44	103	RU	39.2550 D	497.0539	8.95E+01	1
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
44	103	RU	39.2550 D	610.2985	5.64E+00	3
45	106	RURH	1.0200 Y	621.9170 1050.3599	9.81E+00 1.53E+00	1 2
47	108	AGM	127.0000 Y	79.1310 433.9360 614.2810 722.9290	7.62E+00 9.05E+01 8.98E+01 9.08E+01	2 2 1 2

PAGE 2

47	110	AGM	249.7600	D	446.8110	3.75E+00	3
					620.3600	2.81E+00	3
					657.7620	9.46E+01	1
					677.6230	1.04E+01	2
					687.0150	6.43E+00	3
					706.6820	1.64E+01	2
					744.2770	4.71E+00	3
					763.9440	2.23E+01	2
					818.0310	7.31E+00	3
					884.6850	7.27E+01	2
					937.4930	3.44E+01	2
					1384.3000	2.43E+01	2
					1475.7880	4.02E+00	3
					1505.0400	1.30E+01	2
					1562.3030	1.03E+00	3
51	125	SB	2.7590	Y	176.3161	6.90E+00	2
					380.4155	1.51E+00	3
					427.8752	2.94E+01	1
					463.3670	1.04E+01	2
					600.5002	1.78E+01	2
					606.6332	5.00E+00	2
					635.8897	1.13E+01	2
					671.3815	1.81E+00	3
51	124	SB	60.2000	D	602.7300	9.78E+01	1
					645.8550	7.38E+00	2
					709.3248	1.35E+00	3
					713.7810	2.28E+00	3
					722.7860	1.09E+01	2
					968.2010	1.87E+00	3
					1045.1310	1.87E+00	3
					1325.5120	1.59E+00	3
					1355.1771	1.05E+00	3
					1368.1639	2.66E+00	3
					1436.5630	1.25E+00	3
					1690.9800	4.71E+01	2
					2090.9419	5.49E+00	2
55	134	CS	2.0652	Y	475.3573	1.50E+00	3
					563.2370	8.38E+00	3
					569.3207	1.54E+01	2
Z	A	ID	HALF LIFE		ENERGY	BR	I-CODE
55	134	CS	2.0652	Y	604.7101	9.76E+01	2
					795.8674	8.55E+01	1
					801.9511	8.70E+00	3
					1038.5920	9.91E-01	3
					1167.9443	1.79E+00	3
					1365.1844	3.02E+00	3
55	137	CS	30.2500	Y	661.6600	8.52E+01	1
56	133	BA	10.5700	Y	53.1480	2.20E+00	3
					79.6117	2.63E+00	3
					80.9894	3.41E+01	2
					276.3877	7.17E+00	2
					302.8513	1.83E+01	2
					355.9992	6.20E+01	1

PAGE 3

				383.8406	8.93E+00	2
58	144	CE	285.0000 D	80.1034	1.55E+00	2
				133.5386	1.12E+01	1
58	144	CEPR	285.0000 D	80.1035	1.55E+00	2
				133.5387	1.12E+01	2
				696.5101	1.34E+00	2
				1489.1610	2.79E-01	3
				2185.6631	7.00E-01	1
58	141	CE	32.5000 D	145.4404	4.85E+01	1
59	144	PR	17.29000 M	696.5100	1.34E+00	2
				1489.1600	2.79E-01	2
				2185.6621	7.00E-01	1
63	155	EU	4.8500 Y	60.0153	1.14E+00	3
				86.5423	3.26E+01	2
				105.3129	2.23E+01	1
63	152	EU	13.5100 Y	121.7824	2.84E+01	2
				244.6989	7.53E+00	2
				344.2811	2.66E+01	1
				411.1260	2.24E+00	3
				443.9034	3.12E+00	3
				778.9030	1.30E+01	2
				867.3900	4.21E+00	2
				964.1102	1.45E+01	2
				1085.8420	1.01E+01	2
				1089.7670	1.73E+00	3
				1112.0870	1.35E+01	2
				1212.9700	1.41E+00	3
				1299.1520	1.63E+00	3
				1408.0220	2.09E+01	2
63	154	EU	8.5880 Y	123.1001	4.05E+01	2
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
63	154	EU	8.5880 Y	247.9678	6.60E+00	2
				591.8112	4.84E+00	2
				692.4855	1.69E+00	3
				723.3563	1.97E+01	2
				756.8083	4.34E+00	2
				873.2299	1.14E+01	2
				996.3293	1.03E+01	2
				1004.7747	1.79E+01	2
				1274.5435	3.55E+01	1
				1593.0000	1.03E+00	3
				1596.5818	1.83E+00	3
72	181	HF	42.3900 D	132.9411	4.33E+01	2
				136.1687	5.81E+00	3
				345.8287	1.51E+01	2
				481.9971	8.05E+01	1
73	182	TA	115.0000 D	65.7207	2.77E+00	3
				67.7500	4.12E+01	2
				84.6808	2.65E+00	3

			100.1065	1.40E+01	2
			113.6723	1.92E+00	3
			152.4308	7.18E+00	2
			156.3874	2.73E+00	3
			179.3948	3.14E+00	3
			198.3530	1.52E+00	3
			222.1099	7.59E+00	2
			229.3220	3.63E+00	3
			264.0755	3.63E+00	3
			1001.7018	2.00E+00	3
			1121.3010	3.48E+01	2
			1189.0500	1.65E+01	2
			1221.4080	2.73E+01	1
			1231.0160	1.16E+01	2
			1257.4180	1.52E+00	3
			1289.1560	1.40E+00	3
80	203 HG	46.5950 D	279.1967	8.15E+01	1
81	208 TL	3.05300 M	277.2830	6.40E+00	2
			583.1910	8.51E+01	2
			763.0576	1.89E+00	3
			860.5640	1.25E+01	2
			2614.5330	9.98E+01	1
82	214 PB	26.80000 M	53.1716	1.12E+00	3
			241.9192	7.41E+00	2
			295.0908	1.87E+01	2
			351.8664	3.58E+01	1
			785.8274	1.05E+00	3
82	212 PB	10.64000 H	238.6320	4.35E+01	1
Z	A	ID	HALF LIFE	ENERGY	BR
					I-CODE
82	212 PB	10.64000 H	300.0336	3.25E+00	3
83	214 BI	19.90000 M	609.3111	4.61E+01	1
			665.4420	1.56E+00	2
			768.3499	4.89E+00	2
			806.1550	1.23E+00	3
			934.0389	3.16E+00	2
			1120.2727	1.50E+01	2
			1155.1825	1.69E+00	2
			1238.1073	5.92E+00	2
			1280.9519	1.47E+00	2
			1377.6586	4.02E+00	2
			1401.4832	1.39E+00	3
			1407.9697	2.48E+00	2
			1509.2189	2.19E+00	2
			1661.2577	1.15E+00	3
			1729.5803	3.05E+00	2
			1764.4901	1.59E+01	2
			1847.4147	2.12E+00	2
			2118.5254	1.21E+00	3
			2204.0906	4.99E+00	2
			2447.6750	1.55E+00	2
83	212 BI	2.5229 D	727.2466	6.64E+00	1
			785.5095	1.10E+00	2

PAGE 5

			1620.7350	1.49E+00	2
88	226	RAPB	1.60E+03 Y	53.1717	1.12E+00
				74.8137	6.52E+00
				77.1073	1.10E+01
				87.1902	3.88E+00
				186.1111	3.29E+00
				241.9193	7.41E+00
				295.0909	1.87E+01
				351.8665	3.58E+01
				785.8275	1.05E+00
88	226	RA	1.60E+03 Y	186.1109	3.29E+00
88	226	RABI	1.60E+03 Y	186.1111	3.29E+00
				609.3112	4.61E+01
				665.4421	1.56E+00
				768.3500	4.89E+00
				806.1552	1.23E+00
				934.0390	3.16E+00
				1120.2737	1.50E+01
				1155.1835	1.69E+00
				1238.1083	5.92E+00
				1280.9529	1.47E+00
				1377.6595	4.02E+00
				1401.4841	1.39E+00
				1407.9707	2.48E+00

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
88	226	RABI	1.60E+03 Y	1509.2198	2.19E+00	3
				1661.2587	1.15E+00	3
				1729.5813	3.05E+00	3
				1764.4911	1.59E+01	2
				1847.4156	2.12E+00	3
				2118.5264	1.21E+00	3
				2204.0916	4.99E+00	2
				2447.6760	1.55E+00	3
88	224	RA	3.6600 D	240.9870	4.05E+00	1
89	228	AC	6.13000 H	99.5475	1.34E+00	2
				129.0348	2.93E+00	2
				209.3852	4.12E+00	2
				270.2561	3.77E+00	2
				328.0698	3.46E+00	2
				338.4199	1.24E+01	2
				409.6246	2.20E+00	2
				463.1049	4.56E+00	2
				562.6522	1.01E+00	2
				581.5247	1.50E+00	2
				755.2773	1.32E+00	2
				772.2775	1.09E+00	2
				794.7914	4.59E+00	2
				835.6032	1.71E+00	2
				911.1572	2.87E+01	1
				964.6375	5.77E+00	2
				968.9707	1.74E+01	2
				1459.1932	1.06E+00	2
				1496.0311	1.05E+00	2

PAGE 6

				1588.2271	3.59E+00	2
				1630.4746	1.95E+00	2
90	232	TH	1.40E+10 Y	58.9921	1.90E-01	1
90	232	THPB	1.40E+10 Y	58.9922	1.90E-01	4
				74.8138	1.06E+01	8
				77.1074	1.77E+01	8
				87.1903	6.27E+00	8
				238.6321	4.35E+01	1
				300.0337	3.25E+00	3
90	232	THBI	1.40E+10 Y	58.9923	1.90E-01	3
				727.2467	6.64E+00	1
				785.5096	1.10E+00	2
				1620.7360	1.49E+00	2
90	232	THAC	1.40E+10 Y	58.9924	1.90E-01	3
				89.9547	3.42E+00	8
				93.3500	5.56E+00	8
				99.5476	1.34E+00	2
				105.3616	2.01E+00	8
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
90	232	THAC	1.40E+10 Y	129.0349	2.93E+00	2
				209.3853	4.12E+00	2
				270.2562	3.77E+00	2
				328.0699	3.46E+00	2
				338.4200	1.24E+01	2
				409.6247	2.20E+00	2
				463.1050	4.56E+00	2
				562.6523	1.01E+00	3
				581.5248	1.50E+00	3
				755.2774	1.32E+00	3
				772.2776	1.09E+00	3
				794.7916	4.59E+00	2
				835.6033	1.71E+00	3
				911.1573	2.87E+01	1
				964.6376	5.77E+00	2
				968.9708	1.74E+01	2
				1459.1942	1.06E+00	3
				1496.0321	1.05E+00	3
				1588.2280	3.59E+00	2
				1630.4756	1.95E+00	3
90	232	THTL	1.40E+10 Y	58.9925	1.90E-01	3
				74.9693	1.29E+00	8
				277.2831	2.30E+00	2
				583.1911	3.06E+01	2
				860.5641	4.50E+00	2
				2614.5339	3.59E+01	1
90	232	THRA	1.40E+10 Y	58.9926	1.90E-01	3
				240.9871	4.05E+00	1
90	232	THTH	1.40E+10 Y	58.9927	1.90E-01	2
				84.2571	1.22E+00	1
				131.4969	1.27E-01	2
				166.4294	1.07E-01	3

PAGE 7

				215.7539	2.60E-01	2
90	234	TH	4.47E+09 Y	63.2878	3.81E+00	1
				92.5640	5.41E+00	2
90	228	TH	1.9130 Y	84.2570	1.22E+00	1
				131.4968	1.27E-01	2
				166.4293	1.07E-01	3
				215.7538	2.60E-01	2
91	234	PAM	4.47E+09 Y	258.1807	8.10E-02	2
				766.4124	2.96E-01	2
				1000.9970	8.39E-01	1
92	234	U	2.46E+05 Y	53.2306	1.23E-01	1
				120.9115	3.42E-02	2
92	232	U	69.8000 Y	57.8137	2.00E-01	1
Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
92	232	U	69.8000 Y	129.0350	6.82E-02	2
92	235	U	7.04E+08 Y	109.1784	1.54E+00	3
				143.7680	1.10E+01	1
				163.3570	5.08E+00	3
				185.7220	5.72E+01	2
				202.1060	1.08E+00	3
				205.3180	5.01E+00	3
94	239	PU	2.41E+04 Y	129.2960	6.31E-03	1
				144.2010	3.04E-04	3
				203.5500	5.69E-04	3
				332.8450	4.94E-04	3
				345.0130	5.56E-04	3
				375.0540	1.55E-03	3
				380.1910	3.05E-04	3
				382.6980	2.59E-04	3
				413.7130	1.47E-03	3
95	241	AM	432.7000 Y	59.5364	3.59E+01	1

PAGE 8

A.10 Limit Library GE\$LIMIT\$LIB:ENVLIM.LIB

This listing was generated using the "P" (for print) option of MAKE_LIM [11].

LISTING FILE FOR THE LIMIT LIBRARY ENVLIM
GENERATED ON 6-JUL-94 AT 14:27:46

The current limit library documentation file is
ge\$limit\$lib:ENVLIM.doc, revision date/time = 6-JUL-1994 14:27:46.28

The associated limit library file is
 ge\$limit\$lib:ENVLIM.lib, revision date/time = 19-NOV-1993 12:30:01.51

The associated isotope library file is
 ge\$isotope\$lib:ENVLIB.lib, revision date/time = 31-JAN-1994 15:51:52.79

Entry	Nuclide	Energy	ICODE	Half-life	Branching ratio	Isomer?
1	2-QCWA- 23	2223.5000	1	1.00E+10 Y	100.000	N
2	4-BE - 7	477.6050	1	53.2000 D	10.430	N
3	11-NA - 22	1274.5420	1	2.6032 Y	99.930	N
4	13-AL - 26	1808.6484	1	7.20E+05 Y	99.760	N
5	19- K - 40	1460.8300	1	1.26E+09 Y	10.670	N
6	21-SC - 46	889.2770	1	83.7900 D	99.980	N
7	21-SC - 46	1120.5450	2	83.7900 D	99.990	N
8	24-CR - 51	320.0842	1	27.7060 D	9.850	N
9	25-MN - 54	834.8430	1	312.3000 D	99.980	N
10	26-FE - 59	1099.2510	1	44.5100 D	56.100	N
11	26-FE - 59	1291.5959	2	44.5100 D	43.600	N
12	27-CO - 58	810.7915	1	70.8600 D	99.440	N
13	27-CO - 60	1173.2380	2	5.2719 Y	99.890	N
14	27-CO - 60	1332.5020	1	5.2719 Y	99.980	N
15	30-ZN - 65	1115.5460	1	244.2600 D	50.750	N
16	40-ZR - 95	724.1990	2	63.9800 D	44.150	N
17	40-ZR - 95	756.7290	1	63.9800 D	54.500	N
18	41-NB - 94	702.6450	2	2.00E+04 Y	99.790	N
19	41-NB - 94	871.1190	1	2.00E+04 Y	99.860	N

20	41-NB	- 95	765.8070	1	34.9750 D	99.810	N
21	44-RU	-103	497.0539	1	39.2550 D	89.500	N
22	45-RURH-106		621.9170	1	1.0200 Y	9.812	N
23	45-RURH-106		1050.3599	2	2.9800 Y	1.526	N
24	47-AGM	-108	433.9360	2	127.0000 Y	90.480	N
25	47-AGM	-108	614.2810	1	127.0000 Y	89.850	N
26	47-AGM	-108	722.9290	2	127.0000 Y	90.840	N
27	47-AGM	-110	657.7620	1	249.7600 D	94.640	N
28	47-AGM	-110	884.6850	2	249.7600 D	72.680	N
29	47-AGM	-110	1384.3000	2	249.7600 D	24.290	N
30	51-SB	-124	602.7300	1	60.2000 D	97.800	N
31	51-SB	-124	1690.9800	2	60.2000 D	47.120	N
32	51-SB	-125	427.8752	1	2.7590 Y	29.400	N
33	51-SB	-125	600.5002	2	2.7590 Y	17.800	N
34	51-SB	-125	635.8897	2	2.7590 Y	11.300	N
35	55-CS	-134	569.3207	2	2.0652 Y	15.390	N
36	55-CS	-134	604.7101	2	2.0652 Y	97.630	N
37	55-CS	-134	795.8674	1	2.0652 Y	85.520	N
38	55-CS	-134	801.9511	3	2.0652 Y	8.700	N
39	55-CS	-137	661.6600	1	30.2500 Y	85.200	N
40	58-CE	-141	145.4404	1	32.5000 D	48.500	N
41	58-CE	-144	133.5386	1	285.0000 D	11.200	N
42	59-PR	-144	696.5100	2	17.2900 M	1.340	N
43	59-PR	-144	1489.1600	2	17.2900 M	0.279	N

44	59-PR	-144	2185.6621	1	17.2900 M	0.700	N
45	63-EU	-152	121.7824	2	13.5100 Y	28.370	N
46	63-EU	-152	344.2811	1	13.5100 Y	26.570	N
47	63-EU	-152	964.1102	2	13.5100 Y	14.460	N
48	63-EU	-152	1112.0870	2	13.5100 Y	13.540	N
49	63-EU	-152	1408.0220	2	13.5100 Y	20.850	N
50	63-EU	-154	123.1001	2	8.5880 Y	40.460	N
51	63-EU	-154	873.2299	2	8.5880 Y	11.450	N
52	63-EU	-154	1004.7747	2	8.5880 Y	17.880	N
53	63-EU	-154	1274.5435	1	8.5880 Y	35.490	N
54	63-EU	-155	86.5423	2	4.8500 Y	32.600	N
55	63-EU	-155	105.3129	1	4.8500 Y	22.300	N
56	72-HF	-181	481.9971	1	42.3900 D	80.520	N
57	73-TA	-182	67.7500	2	115.0000 D	41.200	N
58	73-TA	-182	1121.3010	2	115.0000 D	34.750	N
59	73-TA	-182	1189.0500	2	115.0000 D	16.510	N
60	73-TA	-182	1221.4080	1	115.0000 D	27.330	N
61	80-HG	-203	279.1967	1	46.5950 D	81.480	N
62	81-TL	-208	583.1910	2	3.0530 M	85.100	N
63	81-TL	-208	860.5640	2	3.0530 M	12.520	N
64	81-TL	-208	2614.5330	1	3.0530 M	99.830	N
65	82-PB	-212	238.6320	1	10.6400 H	43.500	N
66	82-PB	-212	300.0336	3	10.6400 H	3.250	N
67	82-PB	-214	295.0908	2	26.8000 M	18.700	N

68	82-PB	-214	351.8664	1	26.8000 M	35.800	N
69	83-BI	-212	727.2466	1	2.5229 D	6.640	N
70	83-BI	-212	1620.7350	2	2.5229 D	1.490	N
71	83-BI	-214	609.3111	1	19.9000 M	46.090	N
72	83-BI	-214	1120.2727	2	19.9000 M	15.040	N
73	83-BI	-214	1764.4901	2	19.9000 M	15.920	N
74	88-RA	-224	240.9870	1	3.6600 D	4.050	N
75	88-RA	-226	186.1109	1	1.60E+03 Y	3.290	N
76	89-AC	-228	338.4199	2	6.1300 H	12.440	N
77	89-AC	-228	911.1572	1	6.1300 H	28.730	N
78	89-AC	-228	968.9707	2	6.1300 H	17.420	N
79	90-TH	-228	84.2570	1	1.9130 Y	1.220	N
80	90-TH	-228	215.7538	2	1.9130 Y	0.260	N
81	90-TH	-234	63.2878	1	4.47E+09 Y	3.812	N
82	90-TH	-234	92.5640	2	4.47E+09 Y	5.413	N
83	91-PAM	-234	1000.9970	1	4.47E+09 Y	0.839	N
84	92- U	-232	57.8137	1	69.8000 Y	0.200	N
85	92- U	-234	53.2306	1	2.46E+05 Y	0.123	N
86	92- U	-235	109.1784	3	7.04E+08 Y	1.540	N
87	92- U	-235	143.7680	1	7.04E+08 Y	10.960	N
88	92- U	-235	163.3570	3	7.04E+08 Y	5.080	N
89	92- U	-235	185.7220	2	7.04E+08 Y	57.200	N
90	92- U	-235	205.3180	3	7.04E+08 Y	5.010	N
91	94-PU	-239	129.2960	1	2.41E+04 Y	0.006	N

92	94-PU	-239	375.0540	3	2.41E+04 Y	0.002	N
93	94-PU	-239	413.7130	3	2.41E+04 Y	0.001	N
94	95-AM	-241	59.5364	1	432.7000 Y	35.900	N

A.11 Interference Library GE\$INTERF\$LIB:ENVINTLIB.LIB

This listing was generated using the VAXGAP EDINTFER command [2].

INTERFERENCE LIBRARY ENVINTLIB									
CONTAMINATED LINES:					CLEAN LINES:				
ENERGY	Z	ELE	A	ISOMER	ENERGY	Z	ELE	A	ISOMER
53.23	92	U	234	NO	351.87	88	RAPB	226	NO
59.54	95	AM	241	NO	105.31	63	EU	155	NO
99.85	94	PU	238	NO	911.16	90	THAC	232	NO
105.31	63	EU	155	NO	911.16	90	THAC	232	NO
129.30	94	PU	239	NO	911.16	90	THAC	232	NO
133.54	58	CE	144	NO	482.00	72	HF	181	NO
133.54	58	CEPR	144	NO	482.00	72	HF	181	NO
240.99	90	THRA	232	NO	351.87	88	RAPB	226	NO
345.01	94	PU	239	NO	1408.02	63	EU	152	NO
345.01	94	PU	239	NO	482.00	72	HF	181	NO
463.37	51	SB	125	NO	911.16	90	THAC	232	NO
563.24	55	CS	134	NO	911.16	90	THAC	232	NO
722.79	51	SB	124	NO	614.28	47	AGM	108	YES
722.79	51	SB	124	NO	1004.78	63	EU	154	NO
722.93	47	AGM	108	YES	1004.78	63	EU	154	NO
724.20	40	ZR	95	NO	1004.78	63	EU	154	NO
724.20	40	ZR	95	NO	602.73	51	SB	124	NO
756.73	40	ZR	95	NO	1004.78	63	EU	154	NO
763.94	47	AGM	110	YES	2614.53	90	THTL	232	NO
765.81	40	ZRNB	95	NO	1001.00	91	PA	234	NO
795.87	55	CS	134	NO	911.16	90	THAC	232	NO
834.84	25	MN	54	NO	911.16	90	THAC	232	NO
964.11	63	EU	152	NO	911.16	90	THAC	232	NO
1120.55	21	SC	46	NO	609.31	88	RABI	226	NO
1120.55	21	SC	46	NO	1221.41	73	TA	182	NO
1274.54	11	NA	22	NO	1004.78	63	EU	154	NO
1408.02	63	EU	152	NO	609.31	88	RABI	226	NO

A.12 Isotope Library GE\$ISOTOPE\$LIB:EU_CALIBRATION.LIB

This listing was generated using the VAXGAP PRLIB command [2].

EU_CALIBRATION

PAGE 1

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
63	152	EU	13.5100 Y	121.7824	2.84E+01	2
				344.2811	2.66E+01	1
				778.9030	1.30E+01	2
				1112.0870	1.35E+01	2
				1408.0220	2.09E+01	2

A.13 Isotope Library GE\$ISOTOPE\$LIB:TH_CALIBRATION.LIB

This listing was generated using the VAXGAP PRLIB command [2].

TH_CALIBRATION

PAGE 1

Z	A	ID	HALF LIFE	ENERGY	BR	I-CODE
90	228	TH	69.8013 Y	238.6320	4.35E+01	2
				583.1910	3.06E+01	2
				860.5640	4.50E+00	3
				1620.7350	1.50E+00	4
				2614.5330	3.59E+01	1

Appendix B. GC-Related Source Code and Other Listings

The following listings include the main routine for GC and its developer-written subroutines and functions.

B.1 GC Main Routine

```
C MOIFIED TO ELIMINATE USE OF LIB$SPAWN COMMANDS AND THERE BY ELIMINATE THE
C EATING UP OF MEMORY.
C           EWK 6/24/94
C THIS ROUTINE IS A MODIFIED COPY OF THE GC PROGRAM RUNNING AT ISU
C MODIFIED FOR DETECTOR CONFIGURATION TO BE USED ON ROVER FOR THE SWEPP BARREL
C COUNTING JOB
C           EWK 4/9/92
C MODIFIED TO DETERMINE AIM ADDRESS OUT OF GLOBAL SYMBOLS
C   A1$AIM---ADC 1 THRU 2
C   A3$AIM---ADC 3 THRU 4
C
C Modified to convert the entered command to all uppercase before parsing and
C interpreting it. Modified to set IC to the length of the entered command
C plus two in instances where no semicolon is included in the command (e.g.,
C help). Modified to first check whether the command terminator was a ctrl-z,
C and then check if a zero-length string was entered as a command; for a
C ctrl-z, exit; for another zero-length command, directly return to ask for
C the next command. Returned to 6-column display, accessing A1$AIM, A3$AIM,
C and A5$AIM (ADC 5 and 6). Energy calibration options named CBT and PEQT for
C Th-228 and CBE and PEQE for Eu-152. EXIT and QUIT options added (identical
C to ctrl-Z), and all three also erase the "pasteboards." Help split into two
C screens, GC.HELP_1 and GC.HELP_2. D. A. Femec, March 29, 1993.
C
C Modified to allow starting and stopping all NO_ADC detectors simultaneously
C via BS (for barrel or box) and BSTOP, respectively. Container ID to be stored
C is CONTNR and displayed in NVDID. Number of ADCs is passed to ADCAST rather
C than the number of AIMS (in case there might be an odd number of detectors).
C Changed name of analysis routine being called to BATCH_GIDXC_ANALYSIS (since
C the various BATCH_A<n> routines could be consolidated into a single one).
C Added the "RT" command to allow the user to toggle whether sample rotation
C should occur during counting. Modified D and DA to handle multiple detectors.
C Expanded the error messages for "BAD COMMANDS" to indicate what is wrong with
C the command. Reordered the line numbers. Added the "M" command to allow the
C user to toggle whether sample counting should be stopped (and, if requested,
C analysis commenced) if the number of counts in the Pu region exceeds the
C specified level. For external starting of a barrel count (via the DCL
C command "BS <container id>," the logical CONTAINER$ID is checked for, and if
C it is present and not null, an appropriate "BS" command is effected prior to
```

C starting the AST service ADCAST; command dispatching now allows a space
 C instead of a semicolon following the command for compatibility with the DCL
 C command; if the "BS" is from the DCL level, will ask for sample weight,
 C matrix type, and any other comments before starting the counts. Added the
 C "R" command to allow the user to manually start barrel rotation for a period
 C of time. Added checks for valid external matrix types based on the available
 C tables (see AT\$TABLE, which points to GAP\$DISK:[VAXGAPDATA.TABLES]). If any
 C counts are in progress when the user attempts to exit, the user will be asked
 C to repeat the command to confirm it. Modified to rewrite all analysis
 C parameter files (SLOT\$SPC:A<n>.ANP) to contain SWEPP-specific choices before
 C displaying any screens and processing any commands. Added notifications to
 C the "MESSAGES" area as to spectra being written to file, batch and print jobs
 C being submitted, and estimated duration of count for reaching the trigger
 C count level (see ADCAST_GC for related changes). Added "N" as the first
 C parameter in BATCH_CMD so that no container summary report is generated for
 C analyses started using the "DA" command (see BATCH_GIDXC_ANALYSIS.COM).
 C Calls to subroutine SLOSYN control sample rotation. Changed the mass
 C attenuation coefficient file naming scheme to conform to the coefficient
 C numbers presented in Engineering Design File #RWMC-XXX, January 4, 1994,
 C "Drum Matrix Mass Attenuation Coefficients," W. Serrano; also, "BS" expects
 C one of the content codes specified therein instead of the old-style matrix
 C symbol. Added an "A" command for analyzing already stored spectra (this
 C option will check the GE\$SAVE and OLDSWEPP\$SAVE directories for the spectrum
 C specified), and analogous "BA" (without printed individual detector analyses)
 C and "FBA" (with printed individual detector analyses) commands for analyzing
 C a set of spectrum as if it were a barrel analysis (increased the length of
 C CMD to allow a "BA" or "FBA" command explicitly specifying up to six spectrum
 C ids to be entered). Help reworked to allow as many screens of the form
 C GC.HELP_<n> (n=1,2,3,...) as are found. Added "PEQEH" and "PEQTH" commands
 C for generating pulser-based-calibration histories. Inserted a new first
 C parameter into BATCH_CMD and BATCH_CMD2 to specify whether the individual
 C detector analyses should be printed (see also BATCH_GIDXC_ANALYSIS.COM) - for
 C "A"-, "DA"-, "BDA"-, and "FBA"-requested analyses, this parameter is "Y"; for
 C "BA"- and "BS"-requested analyses, this parameter is "N" (also see ADCAST).
 C Added an "RI" command for initializing the barrel turner (i.e., the SLOSYN
 C control). Increased default live time limit for a started count to 36,000
 C seconds. Added commands for typing and printing the most-recent version of
 C the BATCH_GIDXC_ANALYSIS.LOG file and for showing the status of all queues.
 C Added option to specify the isotope library to be used with the DAS command.
 C D. A. Femec, June 22, 1994.

C
 C

C This program, modified as per the above, constitutes version 01.00 of the
 C SWEPP version of GC. D. A. Femec, August 12, 1994.

C
 C

C IN THIS PROGRAM 2 VIRTUAL DISPLAYS ARE CREATED:

```

C      NVDID !DISPLAY OF ADC'S STATUS
C      CVDID !OPERATOR DISPLAY FOR COMMAND INPUT
C DISPLAYS NVDID AND CVDID ARE USED TOGETHER
C 'READTI' IS USED TO READ INPUT FROM THE TERMINAL AND SETS EF 6 WHEN
C A READ IS TERMINATED.
C
C AST'S ARE USED TO CHECK PULSERS DURING COUNTING
C THE FOLLOWING REQIDT'S ARE USED
C      513=AIM #1 ADC 1
C      514=AIM #1 ADC 2
C      515=AIM #2 ADC 1

      PROGRAM GC
C
C Declaration statements.
C
      implicit none      !For safety's sake.
      integer*4 i,j,k    !Common "DO" loop variables.
      INCLUDE '($SMGDEF)'
      INCLUDE '($SECDEF)'
      INCLUDE '($TRMDEF)'
      INCLUDE '($SSDEF)'
      INCLUDE '($IODEF)'
      INCLUDE '($LIBDEF)'
      include '($lnmdef)'      !For logical translation.
      include '($fordef)'      !For FORTRAN error codes.
      INCLUDE 'gap$src:ADC_STATUS.INC/LIST'
      INCLUDE 'SMAN.INC'
      INCLUDE 'ASTTRM.INC'      !COMMON BLOCKS FOR TERMINAL AST'S
      INCLUDE 'GAP$SRC:VTERM.INC'
      INCLUDE 'GAP$SRC:PKD.INC'
      INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
      INCLUDE 'ANALYSIS_PARAMETER.INC'
      INCLUDE 'PULSER_CONFIG.INC'
      INCLUDE 'gap$src:CALIBRATION.INC/LIST'
      INCLUDE 'GAP$SRC:CALB.INC'
      INCLUDE 'QC_QUALITY.INC'
      INCLUDE 'NDFUNC.INC'      !AIM MODULE FUNCTION CALLS

C
C
C      EXTERNAL      READTIB      !READ INPUT FROM THE MAIN TERMINAL
C
C
      integer*2 ibcend      !Next character to be written to in BATCH_CMD2.

      CHARACTER*40      OUTFILE      //' $USER: [OPERATOR]A1.OUTPUT' /
      CHARACTER*40      INFIL        //' $USER: [OPERATOR]A1.INPUT' /

```

```

C
INTEGER*4    SYS$MGBLSC  !MAP TO GLOBAL SECTION
INTEGER*4    MYADR(2),MFLG      !MAP TO GLOBAL SECTION PARAMETERS

INTEGER*4    SYS$ASSIGN,LIB$SPAWN,LIB$GET_LUN
INTEGER*4    SYS$UPDSEC
INTEGER*4    LIB$FREE_LUN,SYS$CANTIM,LIB$AST_IN_PROG
INTEGER*4    SYS$GETTIM,SYS$WAITFR,SYS$CLREF,SYS$READEF
INTEGER*4    SYS$ALLOC
CHARACTER*2  BT      !Mass attenuation coefficient set number.
CHARACTER*2  CH
CHARACTER*2  CLN      /'A1'/
CHARACTER*80  CMD      !COMMAND LINE TO WHICH EDITING IN PERFORMED
BYTE         ASTFLG      !FLAG USED TO TURN OFF AST'S
LOGICAL       SCCMD /.FALSE./ !COMMAND ENTERED FROM MAIN TERMINAL
CHARACTER*40  INERMSG !INPUT ERROR MESSAGE
CHARACTER*30  SCRNM(20) !SCREEN TEMPLATE CHARACTERS
INTEGER*2     POS(20,80) !SCREEN TEMPLATE CHARACTER POSITIONS
INTEGER*2     IMASK
INTEGER*4     ITERM,lnlength,lngold
CHARACTER*80  LINE      /' '/,linold/' '/
CHARACTER*40  CUR_SAVE
LOGICAL       EX      !FOR INQUIRING AS TO EXISTANCE
LOGICAL       ERR
INTEGER*4     START_C,END_C,CV(8192),SYS$SETAST
LOGICAL       NEW_PLOT
BYTE         OFF      /0/
BYTE         ON       /1/
INTEGER*4     SYS$QIO,RDVB,LIB$GET_EF
INTEGER*4     EF,RN
INTEGER*4     INSIZE      /3/
INTEGER*2     T_LUN
INTEGER*4     TERMASK(2) /0,'20000'O/ !ENABLE TERMINATOR ON CR
CHARACTER*3    INLINE

INTEGER*4     IPX
CHARACTER*1    YNO
CHARACTER*11   BSPID
CHARACTER*11   PAR(6)
INTEGER*4     SIDF !LOGICAL UNIT NO. FOR SPECTRUM ID FILE
CHARACTER*8    C_TIME
CHARACTER*14   SID_FILE /'01-MAY-92.1200'/
COMMON /S_ID_F/SIDF

STRUCTURE /IOSTAT_BLOCK/      !QIO STATUS RETURN BLOCK
      INTEGER*2    IOSTAT,
1      TERM_OFFSET,
2      TERMINATOR,

```

```

3          TERM_SIZE
END STRUCTURE
RECORD /IOSTAT_BLOCK/IOSB
C
C AIM ADDRESS SYMBOL DEFS
CHARACTER*6 AIM_SYM(3)  /'A1$AIM','A3$AIM','A5$AIM'/
C
INTEGER*4  LIB$GET_SYMBOL
CHARACTER*6 SYM_VAL
INTEGER*4  NO_AIM        !NUMBER OF AIM MODULES IN CONFIG
BYTE       ADT(6)        /2*0,'AF'X,3*0/    !TEMPORAY BYTE ARRAY
integer*4  no_adc/4/      !Number of detectors to be started via "BS".
INTEGER*4  TLUN !CHANNEL FOR TERMINAL ATTACHED TO BARREL TURNER
COMMON /AIMNO/ no_adc,TLUN
C
C
C
C COMMON FOR AST UPDATE OF STATUS VARIABLES
C IF ASTFLG=0 DONT UPDATE STATUS
C IF ASTFLG=1 UPDATE STATUS
COMMON/AST/ASTFLG
INTEGER*4  REGIDT        /15/
COMMON /CANT/REGIDT !USED TO CANCEL AST REQUEST
C
C
COMMON/EVENT_FLAG/EF
C
C Declarations related to the expanded NVDID display and data acquisition
C and analysis control.
C
character*10 contnr/' (none) '//,    !The container's id.
. cpucnt,    !Character version of the maximum counts in the Pu region.
. catrgr     !Character version of the analysis trigger limit for the above.
character*2 atcoef(29)/'16','12','03','14','17','01','06','01',
. '01','22','11','05','11','11','06','23','02','12','13','13','13',
. '07','06','06','06','06','19','19','06'/    !Coefficient numbers associated with content codes.
real*4 wkg    !Sample weight in kilograms.
integer*4 ipucnt/0/,iatrgr/0/,    !Integer versions of the above.
. datrgr/30000/,    !Default analysis trigger limit.
. icstrt/300/,icend/400/,    !Starting and ending channels for COUNTS.
. luntrm,    !Terminal input's logical unit number.
. nmc_buymodule_f,    !NMC function for "buying" an AIM.
. modifier,    !Screen management terminal modifier.
. ic,    !Position in command line after the semicolon.
. irn,    !ADC number.
. lun,    !Logical unit number for use with files.
. isec,    !Integer form of count time.

```

```

. is,      !First ADC to start.
. ie,      !Last ADC to start.
. iadc,    !Another "first ADC to start."
. jadc     !Another "last ADC to start."
integer*2 iccods(29)/      1,   3,   4,   7, 292, 300, 302, 303,
. 312, 320, 330, 335, 336, 337, 338, 339, 371, 374, 409, 411, 414,
. 432, 440, 441, 442, 464, 480, 481, 490/,      !Known content codes.
. incode,   !User-supplied content code.
. nccods/29/,      !The number of content codes in CNTCDS.
. iccode    !The element of ICCODS that matches the user-supplied INCODE.
logical ifinit/.true./,   !Should the control be initialized?
. ifrota/.true./,        !Should the sample be rotated during a count?
. ifrogo/.false./,       !Is the sample rotating?
. ifmntr/.false./,       !Is the Pu region being monitored for when to analyze?
. ifauto/.false./,       !Is a count being autostarted (based on CONTAINER$ID)?
. dvalid,    !Function for checking whether a detector id is valid.
. ifexit/.false./,       !Used in confirming an exit if counts are in progress.
. ifexst     !Does a help file exist?
common /counts/ ifinit,ifrota,ifrogo,ifmntr,icstrt,icend,iatrgr

```

C
C READ_SPECTRUM-related parameters for barrel counts.
C

```

character*100 commnt
common /barrel/ wkg,bt,commnt

```

C
C Structure used to translate logical's with a call to SYS\$TRNLNM and some
C associated declarations.
C

```

structure /itmlst/
integer*2 mtrln,      !Maximum length of translation (buffer length).
. item      !Item code.
integer*4 adress,     !Address of translated character variable.
. rtrln,    !Length of translated character variable.
. end/0/
end structure
record /itmlst/il
integer*4 lcntnr,     !Length of value returned from logical CONTAINER$ID.
. sys$trnlnm !System service for obtaining a logical.

```

C
C QC-related declarations.
C

```

CHARACTER*8 CTIME
REAL*4      WIDTH_LP
INTEGER*4 IM,ID,IY,IERR,eff      !Efficiency table for QUALITY_UPDATE.

```

C
C Format statements.
C

```

8000 FORMAT(24X,' Press the "Return" key to continue. ', $)
8010 FORMAT(A1)
C
C Beginning of the routine.
C
C ASSIGN BARREL TURNER TO CHANNEL TLUN
      STATUS=SYS$ASSIGN('$SLOSYN', TLUN, ,)
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

C Initialize the SLOSYN control and force a stop of barrel rotation so that the
C current status is known.
C
      ifinit=.true.
      call slosyn(ifinit, 'NG')
      ifrogo=.false.

C
C Rewrite all analysis parameter files (SLOT$SPC:A<n>.ANP) to use SWEPP-
C specific libraries and background correction, to print and save the results,
C and to save the spectra in GE$SAVE.
C
      ap.isotope_lib='SWEPP'
      ap.required_lib='SWEPPREQ'
      ap.interference_lib='SWEPPIF'
      ap.back_correct=.true.
      ap.print_results=.true.
      ap.save_results=.true.
      ap.save_dir='GE$SAVE'
      call lib$get_lun(lun) !Get a logical unit number to use.
      do i=1, no_adc
        open(lun, name=('slot$spc:a'//char(i+48)//'.anp'),
          type='unknown', form='unformatted', iostat=status)
        rewind(lun)
        write(lun) ap
        close(lun, status='keep')
      enddo
      call lib$free_lun(lun) !Free the logical unit number.

C
C WE WILL MAP THE REMOTES MCA STATUS TO THE PERMINATE GLOBAL SECTION
C 'REMOTE$STATUS'. PRIMARILY SO OTHER PROCESSES CAN ACCESS IT.
C
C GET STARTING AND ENDING ADDRESSES OF GLOBAL SECTION
      MYADR(1)=%LOC(AS.REAL_TIME(1))
      MYADR(2)=%LOC(AS.MESSAGE(10))

C
C SET GLOBAL SECTION FLAG TO READ AND WRITE
      MFLG=SEC$M_WRT.OR.SEC$M_SYSGBL
C

```

```

STATUS=SYS$MGBLSC(MYADR,,, %VAL(MFLG), 'ADC$STATUS',,,)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

C
C
C GET CHANNEL AND ASSIGN IT TO SYS$INPUT
STATUS=LIB$GET_LUN(LUNTRM)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=SYS$ASSIGN('SYS$INPUT',LUNTRM,,)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

C
C OPEN CHANNELS TO AIM MODULES
STATUS=NMC_USER_INIT()
IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))

C
C BUY ALL AIM MODULES THAT ARE AVAILABLE. IF THE DCL SYMBOL DEFINED
C IN CHARACTER ARRAY AIM_SYM THEN WE ASSUME THE MODULE IS AVAILABLE AND
C WE WILL ATTEMPT TO BUY IT. WE WILL ALSO INCREMENT THE NUMBER OF MODULES
C FOUND 'NO_AIM' EVERY TIME WE CAN SUCESSFULLY BUY IT.
NO_AIM=0
DO I=1,((no_adc+1)/2) !To ensure that an odd number is handled properly.
STATUS=LIB$GET_SYMBOL(AIM_SYM(I),SYM_VAL)
IF(STATUS .EQ. LIB$NOSUCHSYM)GOTO 100
IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))
READ(SYM_VAL(1:2),FMT='(Z2)')ADT(4)
READ(SYM_VAL(3:4),FMT='(Z2)')ADT(5)
READ(SYM_VAL(5:6),FMT='(Z2)')ADT(6)
STATUS=NMC_FINDMOD_BY_ADDR(AS.AIM_NO(I),ADT)
IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=NMC_BUYMODULE_F(AS.AIM_NO(I))
IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))
NO_AIM=NO_AIM+1
ENDDO

C
100 IF(NO_AIM .EQ. 0)THEN
TYPE *, ' UNABLE TO CONNECT TO ANY MODULES'
CALL EXIT
ENDIF

C
C READ THE CALIBRATION INFO OUT OF THE CALIBRATION BLOCKS FOR EACH DETECTOR
C SO WHEN WE READ A SPECTRUM THE FILES DONT HAVE TO BE RE-READ
DO I=1,no_adc
WRITE(UNIT=CLN(2:2),FMT='(I1)')I
OPEN(UNIT=1,NAME='SLOT$SPC: '//CLN//' .CLB',TYPE='OLD',
1 ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED)
READ(1'1)UC
GCB.CZERO(I)=UC.CALB.CZERO
GCB.A(I)=UC.CALB.A

```



```

GCB.B(I)=UC.CALB.B
GCB.Q(I)=UC.CALB.Q
GCB.FW1(I)=UC.CALB.FW1
GCB.FW2(I)=UC.CALB.FW2
GCB.HPEQ(I)=UC.CALB.HPEQ
GCB.LPEQ(I)=UC.CALB.LPEQ
GCB.CAL_PULSER_WD(I)=UC.CALB.CAL_PULSER_WD
GCB.NO_PULSER(I)=UC.CALB.NO_PULSER
GCB.PULSER_RATE(I)=UC.CALB.PULSER_RATE      !PULSER RATE
GCB.PLS_CHANNELS(I)=UC.CALB.PLS_CHANNELS      !NO OF CHANNELS IN PULSER BUFFER
GCB.PLS_ONE_START(I)=UC.CALB.PLS_ONE_START      !START CHANNEL OF FIRST PULSER
GCB.PLS_TWO_START(I)=UC.CALB.PLS_TWO_START      !START CHANNEL OF SECOND PULSER
GCB.PLS_ONE_PS(I)=UC.CALB.PLS_ONE_PS      !FIRST PULSER OFFSET
GCB.PLS_TWO_PS(I)=UC.CALB.PLS_TWO_PS      !SECOND PULSER OFFSET
CLOSE(UNIT=1)

```

ENDDO

C

C FILL TEMPLATE SCREEN CHARACTER ARRAY

C

```

do i=1,no_adc
SCRN(i)='A'//char(i+48)
enddo
SCRN(7)='DETECTOR'
SCRN(8)='STATUS->'
SCRN(9)='LIVE--->'
SCRN(10)='REAL--->'
SCRN(11)='INTERVAL'
SCRN(12)='ID----->'
SCRN(13)='MESSAGES:'

```

C

C SET POSITIONS

C

```

do i=1,no_adc
POS(4,(3+i*12))=i
enddo
POS(4,1)=7
POS(6,1)=8
POS(8,1)=9
POS(10,1)=10
POS(12,1)=11
POS(14,1)=12
POS(17,1)=13

```

C

C SET MODIFIER SO INPUT IN DISPLAY 'SGID' WILL TERMINATE ON ESCAPE SEQ.

MODIFIER=TRM\$M_TM_ESCAPE.OR.TRM\$M_TM_PURGE.OR.

1 TRM\$M_TM_NOEDIT.OR.TRM\$M_TM_NORECALL

C

```

C CREATE KEY BOARD
  STATUS=SMG$CREATE_KEY_TABLE(IKEYTBL)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
C CREATE PASTEBOARD
C
  STATUS=SMG$CREATE_PASTEBOARD(PBID, !RETURN VALUE
  1      'SYS$OUTPUT',
  2      ROWS,          !RETURN VALUE
  3      COLS)          !RETURN VALUE
C
  NROWS=20
  CROWS=4
  NCOLS=COLS
  CCOLS=COLS
C CREATE TWO VIRTUAL DISPLAYS ONE FOR REMOTE STATUS THE OTHER FOR COMMAND INPUT
  STATUS=SMG$CREATE_VIRTUAL_DISPLAY(NROWS,NCOLS,NVDID)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
  STATUS=SMG$CREATE_VIRTUAL_DISPLAY(CROWS,CCOLS,CVDID)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
C PASTE VIRTUAL DISPLAY TO PASTEBOARD
  STATUS=SMG$PASTE_VIRTUAL_DISPLAY(NVDID,PBID,1,1)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
  STATUS=SMG$PASTE_VIRTUAL_DISPLAY(CVDID,PBID,NROWS+1,1)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
  STATUS=SMG$CREATE_VIRTUAL_KEYBOARD(KEY)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C FILL IN TEMPLATE
  DO I=1,20
    DO J=1,80
      IF(POS(I,J).NE.0)THEN
        K=POS(I,J)
        STATUS=SMG$PUT_CHARS(NVDID,SCRN(K),I,J)
      ENDIF
    ENDDO
  ENDDO
  call smg$put_chars(nvdid,'CONTAINER:',4,70)
  call smg$put_chars(nvdid,contr,5,70,,smg$m_reverse)
  call smg$put_chars(nvdid,'ROTATING?',6,70)
  if(ifrogo)then
    call smg$put_chars(nvdid,'Y',6,79,,smg$m_reverse)
  else
    call smg$put_chars(nvdid,'N',6,79,,smg$m_reverse)
  endif
  call smg$put_chars(nvdid,'ON COUNT?',7,70)
  if(ifrota)then

```

```

        call smg$put_chars(nvdid,'Y',7,79,,smg$m_reverse)
    else
        call smg$put_chars(nvdid,'N',7,79,,smg$m_reverse)
    endif
    call smg$put_chars(nvdid,'MONITOR PU',8,70)
    call smg$put_chars(nvdid,'REGION? ',9,70)
    call smg$put_chars(nvdid,'COUNTS IN',10,70)
    call smg$put_chars(nvdid,'PU REGION:',11,70)
    write(cpuCnt,fmt='(i10)')ipucnt
    call smg$put_chars(nvdid,cpuCnt,12,70,,smg$m_reverse)
    call smg$put_chars(nvdid,'ANALYZE AT',13,70)
    iatrgr=datrgr !Start at the default value.
    write(catrgr,fmt='(i10)')iatrgr
    call smg$put_chars(nvdid,catrgr,14,70,,smg$m_reverse)
    if(ifmnr)then
        call smg$put_chars(nvdid,'Y',9,79,,smg$m_reverse)
    else
        call smg$put_chars(nvdid,'N',9,79,,smg$m_reverse)
    endif

```

C

C DRAW DELIMITING LINES

```

    K=9
    DO I=1,min((no_adc+1),6)
        STATUS=SMG$DRAW_LINE(NVDID,3,K,15,K)
        K=K+12
    ENDDO
    if(no_adc.lt.5)then !Draw diagonal lines to X out the unused area.
        call diagln(nvdid,4,(10+no_adc*12),14,68)
        call diagln(nvdid,14,(10+no_adc*12),4,68)
    endif
    STATUS=SMG$DRAW_LINE(NVDID,3,69,15,69)
    STATUS=SMG$DRAW_LINE(NVDID,5,1,5,(9+no_adc*12))
    STATUS=SMG$DRAW_LINE(NVDID,3,1,3,80)
    STATUS=SMG$DRAW_LINE(NVDID,15,1,15,80)
    STATUS=SMG$PUT_CHARS_HIGHWIDE(NVDID,
    1 'SWEPP GAMMA-RAY BARREL COUNTING SYSTEM',1,4)

```

C

C "Translate" the logical CONTAINER\$ID (note that the logical and table names
 C must be in uppercase for SYS\$TRNLNM to find the logical, i.e. SYS\$TRNLNM is
 C case sensitive). If it is not null, proceed to autostart a count with the
 C container id given.

C

```

    il.item=lnm$string
    il.mtrnln=10
    il.adress=%loc(contnr)
    il.rtrnln=%loc(lcntnr)
    status=sys$trnlnm('LNM$PROCESS_TABLE','CONTAINER$ID',,il)

```

```

        if((.not.status).and.(status.ne.ss$_nolognam))    !Do not exit if no logical found.
        . call lib$stop(%val(status))
        if((lcntnr.eq.0).or.(status.eq.ss$_nolognam))then    !No value, so no autostart.
            cntnr='    '    !Reset the displayed container id.
        else
            llngth=lcntnr+3    !Add three for 'BS;'
            line(1:llngth)='BS;'//cntnr(1:lcntnr)
            cmd(1:3)='BS;'    !So that direct processing works properly.
            ic=4
            ifauto=.true.    !So execution returns to line 110 after starting count.
            goto 300    !Directly process the command.
        endif
C
C ENABLE INPUT AST FOR TERMINAL
110  STATUS=SYS$CLREF(%VAL(6))    !CLEAR EF FOR MAIN TERMINAL
      STATUS=SMG$ENABLE_UNSOLICITED_INPUT(PBID,READTIB)
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
      ASTFLG=1    !ENABLE AST UPDATE OF DISPLAY
      CALL ADCAST
      STATUS=SMG$SET_CURSOR_ABS(CVDID,4,1)
      STATUS=SMG$PUT_CHARS(CVDID,'CMD>')
      STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
      GOTO 130
120  ASTFLG=1
      CALL ADCAST !START AST SERVICE OF STATUS UPDATE
      ASTFLG=1    !ENABLE AST DISPLAY UPDATE
      STATUS=SMG$ERASE_LINE(CVDID,4,1)
      STATUS=SMG$SET_CURSOR_ABS(CVDID,4,1)
      STATUS=SMG$PUT_CHARS(CVDID,'CMD>')
      STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
C
C WAIT FOR OPERATOR TO TYPE A COMMAND
C
130  STATUS=SYS$WAITFR(%VAL(6))
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
      STATUS=SYS$CLREF(%VAL(6))
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
      ITERM=TIITERM
      llngth=TILNGTH
      line='    '    !Reset LINE to prevent any carryover from prior commands.
C
C DISABLE AST DISPLAY UPDATE
C TO INSURE COMMUNICATION WITH TERMINAL IS COMPLETE
      ASTFLG=0
C
C CHECK TO SEE IF <CTRL Z> WAS TYPED ENDING PROGRAM

```

```

IF(ITERM.EQ.26)THEN
C
C Confirm exit if any counts are in progress.
C
    if(.not.ifexit)then
        do i=1,no_adc
            if(as.on_off(i))then
                call smg$put_chars(cvdid,('THERE ARE COUNTS IN '//
                'PROGRESS - EXITING NOW WILL PREVENT AUTOMATIC '//
                'PROCESSING'),2,1)
                call smg$put_chars(cvdid,('REPEAT THE COMMAND TO '//
                'CONFIRM EXITING AT THIS TIME'),3,1)
                ifexit=.true.
                goto 120
            endif
        enddo
    endif

C
C Okay to exit, so do so.
C
    call smg$erase_display(nvdid)
    call smg$erase_display(cvdid)
    call smg$unpaste_virtual_display(nvdid,pbid)
    call smg$unpaste_virtual_display(cvdid,pbid)
    call smg$delete_virtual_keyboard(key)
    STATUS=SMG$DELETE_PASTEBOARD(PBID,0)
    STATUS=SYS$CANTIM(%VAL(REGIDT),)
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=LIB$AST_IN_PROG()
    IF(STATUS)THEN
        CALL MARK(1,1)
        STATUS=SYS$WAITFR(%VAL(1))
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    ENDIF
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C RELEASE AIM MODULES
    DO I=1,NO_AIM
        STATUS=NMC$FREEMODULE(AS.AIM_NO(I))
    ENDDO
    CALL RESET_TT
    CALL EXIT
ENDIF
if(llngth.le.0)goto 120 !No command, so ask for the next one.
LINE(1:llngth)=TILINE(1:llngth)

```

```

STATUS=SYS$CANTIM(%VAL(REGIDT),)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=LIB$AST_IN_PROG()
IF(STATUS)THEN
    CALL MARK(1,1)
    STATUS=SYS$WAITFR(%VAL(1))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
ENDIF
STATUS=SMG$FLUSH_BUFFER(PBID)
C TERMINATED WITH CR ONLY
IF(llngth.EQ.0.AND.ITERM.EQ.13)THEN
    STATUS=SMG$REPAINT_SCREEN(PBID)
    ifexit=.false.    !If an exit preceded this, it is not confirmed.
    GOTO 120
ENDIF
C START DISPATCH TO PROPER COMMAND
C FIRST DETERMINE IF COMMAND TERMINATED WITH ';'
call str$upcase(line,line)    !To ensure the command is in uppercase letters.
K=INDEX(LINE,';')
if(k.eq.0)k=index(line(1:llngth),' ')    !Let a space substitute for a semicolon.
IF(K.NE.0)THEN
    CMD=LINE(1:K)    !YES DISPATCH ONLY ON COMMAND PART OF INPUT
ELSE
    CMD=LINE    !NO-DISPATCH ON WHOLE LINE
    k=llngth+1    !IC is the position that would be following the ;.
ENDIF
IC=K+1
C
C ERASE ERROR MESSAGE ON STATUS DISPLAY
STATUS=SMG$ERASE_LINE(NVDID,17,11)
STATUS=SMG$ERASE_LINE(NVDID,18,1)
STATUS=SMG$ERASE_LINE(NVDID,19,1)
STATUS=SMG$ERASE_LINE(NVDID,20,1)
STATUS=SMG$ERASE_LINE(CVDID,1,1)
STATUS=SMG$ERASE_LINE(CVDID,2,1)
STATUS=SMG$ERASE_LINE(CVDID,3,1)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C CHECK TO SEE IF EXIT OR QUIT WAS TYPED ENDING PROGRAM
IF((CMD(1:IC-2) .EQ. 'EXIT') .OR. (CMD(1:IC-2) .EQ. 'QUIT'))THEN
C
C Confirm exit if any counts are in progress.
C
    if(.not.ifexit)then
        do i=1,no_adc
            if(as.on_off(i))then
                call smg$put_chars(cvidid,('THERE ARE COUNTS IN '//

```

```

        'PROGRESS - EXITING NOW WILL PREVENT AUTOMATIC '//
        'PROCESSING'),2,1)
        call smg$put_chars(cvdid,('REPEAT THE COMMAND TO '//
        'CONFIRM EXITING AT THIS TIME'),3,1)
        ifexit=.true.
        goto 120
    endif
enddo
endif

C
C   Okay to exit, so do so.
C
    call smg$erase_display(nvdid)
    call smg$erase_display(cvdid)
    call smg$unpaste_virtual_display(nvdid,pbid)
    call smg$unpaste_virtual_display(cvdid,pbid)
    call smg$delete_virtual_keyboard(key)
    STATUS=SMG$DELETE_PASTEBOARD(PBID,0)
    STATUS=SYS$CANTIM(%VAL(REGIDT),)
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=LIB$AST_IN_PROG()
    IF(STATUS)THEN
        CALL MARK(1,1)
        STATUS=SYS$WAITFR(%VAL(1))
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    ENDIF
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C RELEASE AIM MODULES
    DO I=1,NO_AIM
        STATUS=NMC_FREEMODULE(AS.AIM_NO(I))
    ENDDO
    CALL RESET_TT
    CALL EXIT
ENDIF

C
C If an exit preceded this and we've reached this point, it is not confirmed.
C
    ifexit=.false.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C GET ANALYSIS PARAMETERS FORMAT: SP;An
C
    IF(CMD(1:IC-2) .EQ. 'SP')THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900    !Bad command

```

```

        CALL STOP_DISPLAY
        CALL ENTER_PAR(LINE(IC:IC+1))
        CALL START_DISPLAY
        GOTO 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE SPECTRUM FORMAT: "CBT;A1"
C FOR A Th-228 SOURCE
C
    IF(CMD(1:IC-2) .EQ. 'CBT')THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900    !Bad command
        READ(UNIT=LINE(6:6),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0)THEN    !No, so set JADC equal to IADC.
            jadc=iadc
        ELSE    !Yes, so set JADC to the last detector's number.
            if(.not.dvalid((cln(1:1)//line(9:9)),cvdid,no_adc))goto 900    !Bad command
            READ(UNIT=LINE(9:9),FMT='(I1)')jadc !ADC NUMBER
        ENDIF
        CLN=LINE(IC:IC+1)
        do irn=iadc,jadc    !For processing all specified detectors.
            WRITE(UNIT=CLN(2:2),FMT='(I1)')IRN !ADC NUMBER
            if(.not.dvalid(cln,cvdid,no_adc))goto 900    !Bad command
            CALL STOP_DISPLAY
            print *,' '
            print *,'PERFORMING NON-PULSERED THORIUM '//
            'CALIBRATION FOR '//cln
            CUR_SAVE=AS.SAVE_DIR(IRN)
            AS.SAVE_DIR(IRN)='TH$SAVE:'
            CALL READ_SPECTRUM(CLN,0.001,'S',
1            'THORIUM CALIBRATION',ERR) !READ SPECTRUM
            AS.SAVE_DIR(IRN)=CUR_SAVE
            IF(ERR)THEN
200                CALL START_DISPLAY
                    GOTO 120
            ENDIF
            CALL TH_CALB(AS.SPECTRUM_ID(IRN),ERR) !ENERGY CALIBRATE
            IF(ERR)GOTO 200
            CLOSE(UNIT=6)
            CALL QUE_PRINT_DEL('TH_CALB.DAT')
C            CALL LIB$SPAWN('PRINT/DELETE/NOIDENTIFY TH_CALB.DAT',,,1)
C
C OPEN CALIBRATION FILE AND UPDATE PROPER VALUE IN CALIBRATION COMMON AREA
        CALL LIB$GET_LUN(LUN)

```



```

OPEN(UNIT=LUN,NAME='SLOT$SPC: '//CLN//''.CLB',
1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED)
READ(LUN'1)UC
GCB.CZERO(IRN)=UC.CALB.CZERO
GCB.A(IRN)=UC.CALB.A
GCB.B(IRN)=UC.CALB.B
GCB.Q(IRN)=UC.CALB.Q
GCB.FW1(IRN)=UC.CALB.FW1
GCB.FW2(IRN)=UC.CALB.FW2
GCB.HPEQ(IRN)=UC.CALB.HPEQ
GCB.LPEQ(IRN)=UC.CALB.LPEQ
GCB.CAL_PULSER_WD(IRN)=UC.CALB.CAL_PULSER_WD
GCB.NO_PULSER(IRN)=UC.CALB.NO_PULSER
GCB.PULSER_RATE(IRN)=UC.CALB.PULSER_RATE !PULSER RATE
GCB.PLS_CHANNELS(IRN)=UC.CALB.PLS_CHANNELS !NO OF CHANNELS IN PULSER BUFFER
GCB.PLS_ONE_START(IRN)=UC.CALB.PLS_ONE_START !START CHANNEL OF FIRST PULSER
GCB.PLS_TWO_START(IRN)=UC.CALB.PLS_TWO_START !START CHANNEL OF SECOND PULSER
GCB.PLS_ONE_PS(IRN)=UC.CALB.PLS_ONE_PS !FIRST PULSER OFFSET
GCB.PLS_TWO_PS(IRN)=UC.CALB.PLS_TWO_PS !SECOND PULSER OFFSET
CLOSE(UNIT=LUN)
CALL LIB$FREE_LUN(LUN)
CALL START_DISPLAY
  if(irn.eq.iadc)then
    call smg$erase_line(nvdid,18,1)
    call smg$put_chars(nvdid,('NON-PULSERED THORIUM '//
      'CALIBRATION COMPLETED FOR ADC '),18,1)
    endif
  call smg$put_chars(nvdid,cln,18,(49+irn*3))
  enddo
GOTO 120
ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE SPECTRUM FORMAT: "CBE;A1"
C FOR A Eu-152 SOURCE
C
IF(CMD(1:IC-2) .EQ. 'CBE')THEN
  if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900 !Bad command
  READ(UNIT=LINE(6:6),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
J=INDEX(LINE(1:IC+2),'-')
IF(J .EQ. 0)THEN !No, so set JADC equal to IADC.
  jadc=iadc
ELSE !Yes, so set JADC to the last detector's number.
  if(.not.dvalid((cln(1:1)//line(9:9)),cvdid,no_adc))goto 900 !Bad command
  READ(UNIT=LINE(9:9),FMT='(I1)')jadc !ADC NUMBER
ENDIF

```

```

        CLN=LINE(IC:IC+1)
        do irn=iadc,jadc !For processing all specified detectors.
        WRITE(UNIT=CLN(2:2),FMT='(I1)')IRN !ADC NUMBER
        CALL STOP_DISPLAY
        print *, ' '
        print *, 'PERFORMING NON-PULSERED EUROPIUM ' //
        'CALIBRATION FOR ' //cln
        CUR_SAVE=AS.SAVE_DIR(IRN)
        AS.SAVE_DIR(IRN)='EU$SAVE:'
        CALL READ_SPECTRUM(CLN,0.001,'S',
1      'EUROPIUM CALIBRATION',ERR) !READ SPECTRUM
        AS.SAVE_DIR(IRN)=CUR_SAVE
        IF(ERR)THEN
210      CALL START_DISPLAY
          GOTO 120
        ENDIF
        CALL EU_CALB(AS.SPECTRUM_ID(IRN),ERR) !ENERGY CALIBRATE
        IF(ERR)GOTO 210
        CLOSE(UNIT=6)
        CALL QUE_PRINT_DEL('EU_CALB.DAT')
C      CALL LIB$SPAWN('PRINT/DELETE/NOIDENTIFY EU_CALB.DAT',,,1)

C
C OPEN CALIBRATION FILE AND UPDATE PROPER VALUE IN CALIBRATION COMMON AREA
        CALL LIB$GET_LUN(LUN)
        OPEN(UNIT=LUN,NAME='SLOT$SPC: '//CLN//' .CLB',
1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED)
        READ(LUN'1)UC
        GCB.CZERO(IRN)=UC.CALB.CZERO
        GCB.A(IRN)=UC.CALB.A
        GCB.B(IRN)=UC.CALB.B
        GCB.Q(IRN)=UC.CALB.Q
        GCB.FW1(IRN)=UC.CALB.FW1
        GCB.FW2(IRN)=UC.CALB.FW2
        GCB.HPEQ(IRN)=UC.CALB.HPEQ
        GCB.LPEQ(IRN)=UC.CALB.LPEQ
        GCB.CAL_PULSER_WD(IRN)=UC.CALB.CAL_PULSER_WD
        GCB.NO_PULSER(IRN)=UC.CALB.NO_PULSER
        GCB.PULSER_RATE(IRN)=UC.CALB.PULSER_RATE !PULSER RATE
        GCB.PLS_CHANNELS(IRN)=UC.CALB.PLS_CHANNELS !NO OF CHANNELS IN PULSER BUFFER
        GCB.PLS_ONE_START(IRN)=UC.CALB.PLS_ONE_START !START CHANNEL OF FIRST PULSER
        GCB.PLS_TWO_START(IRN)=UC.CALB.PLS_TWO_START !START CHANNEL OF SECOND PULSER
        GCB.PLS_ONE_PS(IRN)=UC.CALB.PLS_ONE_PS !FIRST PULSER OFFSET
        GCB.PLS_TWO_PS(IRN)=UC.CALB.PLS_TWO_PS !SECOND PULSER OFFSET
        CLOSE(UNIT=LUN)
        CALL LIB$FREE_LUN(LUN)
        CALL START_DISPLAY

```

```

        if(irn.eq.iadc)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('NON-PULSERED EUROPIUM '//
            'CALIBRATION COMPLETED FOR ADC '),18,1)
        endif
        call smg$put_chars(nvdid,cln,18,(50+irn*3))
        enddo
        GOTO 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE SPECTRUM AND COMPUTE PULSER EQUIVALENTS FORMAT: "PEQT;A1"
C FOR A Th-228 SOURCE
C
    IF(CMD(1:IC-2) .EQ. 'PEQT')THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900    !Bad command
        READ(UNIT=LINE(7:7),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0)THEN    !No, so set JADC equal to IADC.
            jadc=iadc
        ELSE    !Yes, so set JADC to the last detector's number.
            if(.not.dvalid((cln(1:1)//line(10:10)),cvdid,no_adc))goto 900    !Bad command
            READ(UNIT=LINE(10:10),FMT='(I1)')jadc    !ADC NUMBER
        ENDIF
        CLN=LINE(IC:IC+1)
        CALL STOP_DISPLAY
        do irn=iadc,jadc    !For processing all specified detectors.
            WRITE(UNIT=CLN(2:2),FMT='(I1)')IRN !ADC NUMBER
            print *,' '
            print *,'PERFORMING PULSERED THORIUM CALIBRATION '//
            'FOR '//cln
            CUR_SAVE=AS.SAVE_DIR(IRN)
            AS.SAVE_DIR(IRN)='TH$SAVE:'
            CALL READ_SPECTRUM(CLN,0.001,'S',
1          'THORIUM CALIBRATION',ERR) !READ SPECTRUM
            AS.SAVE_DIR(IRN)=CUR_SAVE
            IF(ERR)THEN
220          CALL START_DISPLAY
              GOTO 120
            ENDIF
            CALL PEQ_TH(AS.SPECTRUM_ID(IRN),ERR) !ENERGY CALIBRATE
            IF(ERR)GOTO 220
        enddo
C
C OPEN CALIBRATION FILE AND UPDATE PROPER VALUE IN CALIBRATION COMMON AREA
        CALL LIB$GET_LUN(LUN)

```

```

      OPEN(UNIT=LUN,NAME='SLOT$SPC: '//CLN//'.CLB',
1  TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED)
      READ(LUN'1)UC
      GCB.CZERO(IRN)=UC.CALB.CZERO
      GCB.A(IRN)=UC.CALB.A
      GCB.B(IRN)=UC.CALB.B
      GCB.Q(IRN)=UC.CALB.Q
      GCB.FW1(IRN)=UC.CALB.FW1
      GCB.FW2(IRN)=UC.CALB.FW2
      GCB.HPEQ(IRN)=UC.CALB.HPEQ
      GCB.LPEQ(IRN)=UC.CALB.LPEQ
      GCB.CAL_PULSER_WD(IRN)=UC.CALB.CAL_PULSER_WD
      GCB.NO_PULSER(IRN)=UC.CALB.NO_PULSER
      GCB.PULSER_RATE(IRN)=UC.CALB.PULSER_RATE !PULSER RATE
      GCB.PLS_CHANNELS(IRN)=UC.CALB.PLS_CHANNELS !NO OF CHANNELS IN PULSER BUFFER
      GCB.PLS_ONE_START(IRN)=UC.CALB.PLS_ONE_START !START CHANNEL OF FIRST PULSER
      GCB.PLS_TWO_START(IRN)=UC.CALB.PLS_TWO_START !START CHANNEL OF SECOND PULSER
      GCB.PLS_ONE_PS(IRN)=UC.CALB.PLS_ONE_PS !FIRST PULSER OFFSET
      GCB.PLS_TWO_PS(IRN)=UC.CALB.PLS_TWO_PS !SECOND PULSER OFFSET
      WIDTH_LP=PKD(1,1,WV) !SAVE WIDTH OF LOW PULSER TO PUT IN PARAM BLOCK
      CLOSE(UNIT=LUN)
      CALL LIB$FREE_LUN(LUN)
C  UPDATE PARAMETER BLOCK OF SPECTRUM WITH NEW PULSER EQUIVALENTS
      CALL OPEN$SP(AS.SPECTRUM_ID(IRN),IERR)
      P.PLS_ONE_KEV=PKD(1,3,ERA)
      P.PLS_TWO_KEV=PKD(2,3,ERA)
      P.CAL_PULSER_WD=WIDTH_LP !LOW PULSER WIDTH
      WRITE(1'2)P
C
C  UPDATE INFORMATION INTO THE QUALITY CONTROL STRUCTURE
      CALL IDATE(IM,ID,IY)
      WRITE(QC.DATE,FMT='(I6.6)')IM*10000 + ID*100 + IY
      CALL TIME(CTIME)
      QC.TIME=CTIME(1:2)//CTIME(4:5)
      J=INDEX(AS.SPECTRUM_ID(IRN),'.DGE')-11
      QC.SPECTRUM_ID=AS.SPECTRUM_ID(IRN)(J:J+10)
      QC.ZERO=UC.CALB.CZERO
      QC.A_TERM=UC.CALB.A
      QC.B_TERM=UC.CALB.B
      QC.C_TERM=UC.CALB.Q
      QC.W_ZERO=UC.CALB.FW1
      QC.W_LIN=UC.CALB.FW2
      QC.PULSER=.TRUE.
      QC.LOW_PULSER_EQ=UC.CALB.LPEQ
      QC.HIGH_PULSER_EQ=UC.CALB.HPEQ
      QC.NONE=1
      QC.NA(1)=228

```

```

        QC.NZ(1)=90
        QC.NID(1)='TH'

C DO THE QUALITY DATA UPDATE
        CLOSE(UNIT=1)
        eff=irn*1000+110
        CALL QUALITY_UPDATE(AS.SPECTRUM_ID(IRN),
1 'GE$ISOTOPE$LIB:TH_CALIBRATION.LIB',eff)

        CLOSE(UNIT=6)
        CALL QUE_PRINT_DEL('TH_CALB.DAT')
C        CALL LIB$SPAWN('PRINT/DELETE/NOIDENTIFY TH_CALB.DAT',,,1)

        if(irn.eq.iadc)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('PULSERED THORIUM '//
            'CALIBRATION COMPLETED FOR ADC '),18,1)
        endif
        call smg$put_chars(nvdid,cln,18,(45+irn*3))
        enddo
        CALL START_DISPLAY
        GOTO 120
    ENDIF

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE WITH PULSER EQUIVALENTS HISTORY FORMAT: "PEQTH;A1"
C FOR A Th-228 SOURCE
C
    IF(CMD(1:IC-2) .EQ. 'PEQTH')THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900    !Bad command
        READ(UNIT=LINE(8:8),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0)THEN    !No, so set JADC equal to IADC.
            jadc=iadc
        ELSE    !Yes, so set JADC to the last detector's number.
            if(.not.dvalid((cln(1:1)//line(11:11)),cvdid,no_adc))goto 900    !Bad command
            READ(UNIT=LINE(11:11),FMT='(I1)')jadc    !ADC NUMBER
        ENDIF
        CLN=LINE(IC:IC+1)
        CALL STOP_DISPLAY
        do irn=iadc,jadc    !For processing all specified detectors.
        WRITE(UNIT=CLN(2:2),FMT='(I1)')IRN !ADC NUMBER
        print *,' '
        print *,'GENERATING PULSERED THORIUM CALIBRATION '//
        'HISTORY FOR '//cln
        call qa_report_sub(cln,'TH')
    
```

```

        enddo
        CALL START_DISPLAY
        GOTO 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE SPECTRUM AND COMPUTE PULSER EQUIVALENTS FORMAT: "PEQE;A1"
C FOR A Eu-152 SOURCE
C
    IF(CMD(1:IC-2) .EQ. 'PEQE') THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc)) goto 900    !Bad command
        READ(UNIT=LINE(7:7),FMT='(I1)') iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0) THEN    !No, so set JADC equal to IADC.
            jadc=iadc
        ELSE    !Yes, so set JADC to the last detector's number.
            if(.not.dvalid((cln(1:1)//line(10:10)),cvdid,no_adc)) goto 900    !Bad command
            READ(UNIT=LINE(10:10),FMT='(I1)') jadc    !ADC NUMBER
        ENDIF
        CLN=LINE(IC:IC+1)
        CALL STOP_DISPLAY
        do irn=iadc,jadc    !For processing all specified detectors.
        WRITE(UNIT=CLN(2:2),FMT='(I1)') IRN !ADC NUMBER
            print *, ' '
            print *, 'PERFORMING PULSERED EUROPIUM CALIBRATION ' //
            'FOR '//cln
            CUR_SAVE=AS.SAVE_DIR(IRN)
            AS.SAVE_DIR(IRN)='EU$SAVE:'
            CALL READ_SPECTRUM(CLN,0.001,'S',
1          'EUROPIUM CALIBRATION',ERR) !READ SPECTRUM
            AS.SAVE_DIR(IRN)=CUR_SAVE
            IF(ERR) THEN
230          CALL START_DISPLAY
              GOTO 120
            ENDIF
            CALL PEQ_EU(AS.SPECTRUM_ID(IRN),ERR) !ENERGY CALIBRATE
            IF(ERR) GOTO 230
        enddo
C
C OPEN CALIBRATION FILE AND UPDATE PROPER VALUE IN CALIBRATION COMMON AREA
        CALL LIB$GET_LUN(LUN)
        OPEN(UNIT=LUN,NAME='SLOT$SPC: '//CLN//' .CLB',
1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED)
        READ(LUN'1) UC
        GCB.CZERO(IRN)=UC.CALB.CZERO
        GCB.A(IRN)=UC.CALB.A

```

```

GCB.B( IRN)=UC.CALB.B
GCB.Q( IRN)=UC.CALB.Q
GCB.FW1( IRN)=UC.CALB.FW1
GCB.FW2( IRN)=UC.CALB.FW2
GCB.HPEQ( IRN)=UC.CALB.HPEQ
GCB.LPEQ( IRN)=UC.CALB.LPEQ
GCB.CAL_PULSER_WD( IRN)=UC.CALB.CAL_PULSER_WD
GCB.NO_PULSER( IRN)=UC.CALB.NO_PULSER
GCB.PULSER_RATE( IRN)=UC.CALB.PULSER_RATE !PULSER RATE
GCB.PLS_CHANNELS( IRN)=UC.CALB.PLS_CHANNELS !NO OF CHANNELS IN PULSER BUFFER
GCB.PLS_ONE_START( IRN)=UC.CALB.PLS_ONE_START !START CHANNEL OF FIRST PULSER
GCB.PLS_TWO_START( IRN)=UC.CALB.PLS_TWO_START !START CHANNEL OF SECOND PULSER
GCB.PLS_ONE_PS( IRN)=UC.CALB.PLS_ONE_PS !FIRST PULSER OFFSET
GCB.PLS_TWO_PS( IRN)=UC.CALB.PLS_TWO_PS !SECOND PULSER OFFSET
WIDTH_LP=PKD(1,1,WV) !SAVE WIDTH OF LOW PULSER TO PUT IN PARAM BLOCK
CLOSE(UNIT=LUN)
CALL LIB$FREE_LUN(LUN)
C UPDATE PARAMETER BLOCK OF SPECTRUM WITH NEW PULSER EQUIVALENTS
CALL OPEN$SP(AS.SPECTRUM_ID( IRN), IERR)
P.PLS_ONE_KEV=PKD(1,3,ERA)
P.PLS_TWO_KEV=PKD(2,3,ERA)
P.CAL_PULSER_WD=WIDTH_LP !LOW PULSER WIDTH
WRITE(1'2)P
C
C UPDATE INFORMATION INTO THE QUALITY CONTROL STRUCTURE
CALL IDATE(IM, ID, IY)
WRITE(QC.DATE, FMT='(I6.6)') IM*10000 + ID*100 + IY
CALL TIME(CTIME)
QC.TIME=CTIME(1:2)//CTIME(4:5)
J=INDEX(AS.SPECTRUM_ID( IRN), '.DGE')-11
QC.SPECTRUM_ID=AS.SPECTRUM_ID( IRN) (J:J+10)
QC.ZERO=UC.CALB.CZERO
QC.A_TERM=UC.CALB.A
QC.B_TERM=UC.CALB.B
QC.C_TERM=UC.CALB.Q
QC.W_ZERO=UC.CALB.FW1
QC.W_LIN=UC.CALB.FW2
QC.PULSER=.TRUE.
QC.LOW_PULSER_EQ=UC.CALB.LPEQ
QC.HIGH_PULSER_EQ=UC.CALB.HPEQ
QC.NONE=1
QC.NA(1)=152
QC.NZ(1)=63
QC.NID(1)='EU'
C DO THE QUALITY DATA UPDATE
CLOSE(UNIT=1)

```

```

        eff=irn*1000+110
        CALL QUALITY_UPDATE(AS.SPECTRUM_ID(IRN),
1 'GE$ISOTOPE$LIB:EU_CALIBRATION.LIB',eff)

        CLOSE(UNIT=6)
        CALL QUE_PRINT_DEL('EU_CALB.DAT')
C      CALL LIB$SPAWN('PRINT/DELETE/NOIDENTIFY EU_CALB.DAT',,,1)

        if(irn.eq.iadc)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('PULSERED EUROPIUM '//
            'CALIBRATION COMPLETED FOR ADC '),18,1)
            endif
            call smg$put_chars(nvdid,cln,18,(46+irn*3))
            enddo
        CALL START_DISPLAY
        GOTO 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ENERGY CALIBRATE WITH PULSER EQUIVALENTS HISTORY FORMAT: "PEQEH;A1"
C FOR A Eu-152 SOURCE
C
    IF(CMD(1:IC-2) .EQ. 'PEQEH')THEN
        if(.not.dvalid(line(ic:(ic+1)),cvdid,no_adc))goto 900    !Bad command
        READ(UNIT=LINE(8:8),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR CALIBRATION?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0)THEN    !No, so set JADC equal to IADC.
            jadc=iadc
        ELSE    !Yes, so set JADC to the last detector's number.
            if(.not.dvalid((cln(1:1)//line(11:11)),cvdid,no_adc))goto 900    !Bad command
            READ(UNIT=LINE(11:11),FMT='(I1)')jadc    !ADC NUMBER
        ENDIF
        CLN=LINE(IC:IC+1)
        CALL STOP_DISPLAY
        do irn=iadc,jadc    !For processing all specified detectors.
        WRITE(UNIT=CLN(2:2),FMT='(I1)')IRN !ADC NUMBER
            print *, ' '
            print *, 'GENERATING PULSERED EUROPIUM CALIBRATION '//
            'HISTORY FOR '//cln
            call qa_report_sub(cln,'EU')
            enddo
        CALL START_DISPLAY
        GOTO 120
    ENDIF
C

```



```

P.EFF_FILE=' '
P.VOLUME=1.0
P.VOLUME_UNITS='CC'
P.NO_CHANNELS=8192.0
P.ZERO_OFFSET=GCB.CZERO(RN)
P.EA_CO=GCB.A(RN)
P.EB_CO=GCB.B(RN)
P.EC_CO=GCB.Q(RN)
P.WA_CO=GCB.FW1(RN)
P.WB_CO=GCB.FW2(RN)
P.PLS_ONE_KEV=GCB.LPEQ(RN)
P.PLS_TWO_KEV=GCB.HPEQ(RN)
P.CAL_PULSER_WD=GCB.CAL_PULSER_WD(RN)
P.NO_PULSER=GCB.NO_PULSER(RN)
P.PULSER_RATE=GCB.PULSER_RATE(RN)
P.PLS_CHANNELS=GCB.PLS_CHANNELS(RN)
P.PLS_ONE_START=GCB.PLS_ONE_START(RN)
P.PLS_TWO_START=GCB.PLS_TWO_START(RN)
P.PLS_ONE_PS=GCB.PLS_ONE_PS(RN)
P.PLS_TWO_PS=GCB.PLS_TWO_PS(RN)

```

```

C
CALL DAS_SUB(LINE(5:6))
CALL START_DISPLAY
STATUS=SYSS$SETAST(%VAL(ON)) !ENABLE AST DELIVERY
GOTO 120

```

```

ENDIF

```

```

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C START DETECTOR COUNTING FORMAT: S;A1-A6

```

```

C
C "BS" is used for a simultaneous start of all NO_ADC detectors. The
C container id follows the semicolon, the count length (unless the default
C is to be used) follows the comma.

```

```

C
300 if(cmd(1:(ic-2)).eq.'BS')then
    contr=' ' !Reset the container id.
    j=index(line(1:llngth),',')
    if(j.eq.0)then !Default count length; id from IC to LLNGTH.
        if((1+min((ic+9),llngth)-ic).gt.0)contr(1:(1+min((ic+9),
            llngth)-ic))=line(ic:min((ic+9),llngth)) !Set only the characters needed.
        line='S;A1-A'//char(no_adc+48)//',36000'
        llngth=13
    else !Assume that the given count length is correct.
        if((1+min((ic+9),(j-1))-ic).gt.0)contr(1:(1+min((ic+9),
            (j-1))-ic))=line(ic:min((ic+9),(j-1))) !Set only the characters needed.
        line='S;A1-A'//char(no_adc+48)//line(j:llngth)
        llngth=8+llngth-j
    end if
end if

```

```

endif
linold=line !Save the command line for later processing.
lngold=llength

C
C Obtain the weight in kilograms, the content code, and any comments.
C The comment always starts with the container id.
C

302 STATUS=SMG$ERASE_LINE(CVDID,2,1)
STATUS=SMG$ERASE_LINE(CVDID,3,1)
call smg$put_chars(cvdid,'ENTER THE WEIGHT IN KILOGRAMS',3,1)
if(.not.ifauto)ASTFLG=1 !ENABLE AST DISPLAY UPDATE
STATUS=SMG$ERASE_LINE(CVDID,4,1)
STATUS=SMG$SET_CURSOR_ABS(CVDID,4,1)
STATUS=SMG$PUT_CHARS(CVDID,'CMD>')
STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
if(ifauto)then !Input handling depends on whether terminal AST has been enabled.
call smg$read_string(key,tiline,,80,,,,llength,
item,cvdid)
else
STATUS=SYS$WAITFR(%VAL(6))
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=SYS$CLREF(%VAL(6))
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
ITERM=TIITERM
llength=TILNGTH
ASTFLG=0 !DISABLE AST DISPLAY UPDATE
STATUS=SYS$CANTIM(%VAL(REGIDT),)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=LIB$AST_IN_PROG()
IF(STATUS)THEN
CALL MARK(1,1)
STATUS=SYS$WAITFR(%VAL(1))
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
ENDIF
STATUS=SMG$FLUSH_BUFFER(PBID)
endif
if(llength.le.0)goto 302 !No response, so ask again.
read(TILINE(1:llength),fmt=*,err=303)wkg
if(wkg.le.0.0)then !Positive weights only.
303 call smg$put_chars(cvdid,('INVALID WEIGHT - ENTER A '//
'POSITIVE NUMBER'),2,1)
goto 302
endif
304 STATUS=SMG$ERASE_LINE(CVDID,2,1)
STATUS=SMG$ERASE_LINE(CVDID,3,1)
call smg$put_chars(cvdid,'ENTER THE CONTENT CODE',3,1)
ASTFLG=1 !ENABLE AST DISPLAY UPDATE

```

```

STATUS=SMG$ERASE_LINE(CVDID,4,1)
STATUS=SMG$SET_CURSOR_ABS(CVDID,4,1)
STATUS=SMG$PUT_CHARS(CVDID,'CMD>')
STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
if(ifauto)then      !Input handling depends on whether terminal AST has been enabled.
    call smg$read_string(key,tiline,,80,,,llength,
        iterm,cvdid)
    else
        STATUS=SYS$WAITFR(%VAL(6))
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
        STATUS=SYS$CLREF(%VAL(6))
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
        ITERM=TIITERM
        llength=TILNGTH
        ASTFLG=0      !DISABLE AST DISPLAY UPDATE
        STATUS=SYS$CANTIM(%VAL(REGIDT),)
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
        STATUS=LIB$AST_IN_PROG()
        IF(STATUS)THEN
            CALL MARK(1,1)
            STATUS=SYS$WAITFR(%VAL(1))
            IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
        ENDIF
        STATUS=SMG$FLUSH_BUFFER(PBID)
    endif
305 if(llength.le.0)then      !Invalid response, so ask again.
    call smg$put_chars(cvdid,'INVALID CONTENT CODE',2,1)
    goto 304
endif
call str$upcase(tiline,tiline)
read(tiline(1:llength),fmt=*,err=305,end=305)incode
iccode=0      !In case no match is found.
do k=1,nccods      !Is this content code known?
    if(incode.eq.iccods(k))iccode=k      !Yes it is.
enddo
if(iccode.eq.0)goto 305      !No match found.

C DOES FILE EXIST?
INQUIRE(FILE=('AT$TABLE:'//atcoef(iccode)//'.TBL'),EXIST=EX)
IF(.NOT. EX)goto 305

bt=atcoef(iccode)      !Accept the mass attenuation coefficient number.
306 STATUS=SMG$ERASE_LINE(CVDID,2,1)
STATUS=SMG$ERASE_LINE(CVDID,3,1)
call smg$put_chars(cvdid,'ENTER ANY COMMENTS',3,1)
ASTFLG=1      !ENABLE AST DISPLAY UPDATE
STATUS=SMG$ERASE_LINE(CVDID,4,1)

```

```

STATUS=SMG$SET_CURSOR_ABS(CVDID,4,1)
STATUS=SMG$PUT_CHARS(CVDID,'CMD>')
STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
if(ifauto)then      !Input handling depends on whether terminal AST has been enabled.
    call smg$read_string(key,tiline,,80,,,llength,
        iterm,cvdid)
    call smg$erase_line(cvdid,4,5)
else
    STATUS=SYS$WAITFR(%VAL(6))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=SYS$CLREF(%VAL(6))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    ITERM=TIITERM
    llength=TILNGTH
    ASTFLG=0      !DISABLE AST DISPLAY UPDATE
    STATUS=SYS$CANTIM(%VAL(REGIDT),)
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    STATUS=LIB$AST_IN_PROG()
    IF(STATUS)THEN
        CALL MARK(1,1)
        STATUS=SYS$WAITFR(%VAL(1))
        IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    ENDIF
    STATUS=SMG$FLUSH_BUFFER(PBID)
endif
STATUS=SMG$ERASE_LINE(CVDID,3,1)
commnt(1:17)='Barrel '//contr
if(llength.gt.0)then      !Append the user-supplied comment.
    call str$upcase(tiline,tiline)
    commnt=commnt(1:17)///': '//tiline(1:llength)
endif
STATUS=SMG$ERASE_LINE(CVDID,2,1)

```

C Rewrite CMD and LINE to an appropriate START command and let START handle
 C some of the error checking.
 C

```

line=linold
llength=lngold
cmd='S'
ic=3
call smg$put_chars(nvdid,contr,5,70,,smg$m_reverse)      !Update display.
if(.not.ifrota)then      !Force barrel rotation.
    ifrota=.true.
    call smg$put_chars(nvdid,'Y',7,79,,smg$m_reverse)
endif
ifmnr=.true.      !Start Pu region monitoring.
call smg$put_chars(nvdid,'Y',9,79,,smg$m_reverse)

```

```

write(catrgr,fmt='(i10)')iatrgr
call smg$put_chars(nvdid,catrgr,14,70,,smg$m_reverse)
ipucnt=0 !Reset the current counts read.
write(cpucnt,fmt='(i10)')ipucnt
call smg$put_chars(nvdid,cpucnt,12,70,,smg$m_reverse)
endif

C
  IF(CMD(1:IC-2) .EQ. 'S')THEN
    CLN=LINE(IC:IC+1)
    J=INDEX(LINE(1:IC+2),'-')
C IS COMMAND A MULTIPLE DETECTOR START?
    IF(J .EQ. 0)THEN
      if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
      READ(UNIT=CLN(2:2),FMT='(I1)')I !ADC NUMBER
      IC=IC+2
      IF(LINE(IC:IC) .NE. ',')then !Incomplete command.
310        call smg$put_chars(cvdid,('INCOMPLETE '//
          'COMMAND - COUNT TIME MISSING OR BAD'),2,1)
          goto 900
        endif
      IC=IC+1
      READ(UNIT=LINE(IC:llngth),FMT=*,ERR=310)ISEC
      if(as.on_off(i))goto 320 !Already counting on this detector!
      if(ifrota)then
        call slosyn(ifinit,'GOSL') !Start rotating the sample.
        ifrogo=.true.
        call smg$put_chars(nvdid,'Y',6,79,,
          smg$m_reverse)
        endif
      CALL START_COUNTING(ISEC,I,ERR)
      AS.SPECTRUM_ID(I)=' '
      IF(ERR)THEN
320        STATUS=SMG$ERASE_LINE(CVDID,3,1)
        STATUS=SMG$RING_BELL(CVDID,2)
        STATUS=SMG$PUT_CHARS(CVDID,
1 'UNABLE TO START DATA COLLECTION ON ADC '//CLN,3,1)
        if(ifrota)then
          call slosyn(ifinit,'NG') !Stop rotating the sample.
          ifrogo=.false.
          call smg$put_chars(nvdid,'N',6,79,,
            smg$m_reverse)
          endif
        else
          call smg$erase_line(nvdid,18,1)
          call smg$put_chars(nvdid,('DATA '//
            'COLLECTION STARTED ON ADC '//CLN),18,1)
        endif
      ENDIF
    ENDIF
  ENDIF

```

```

        GOTO 120
ELSE
    if(.not.dvalid(cln,cvdid,no_adc))goto 900    !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IS !ADC TO START
    CLN=LINE(J+1:J+2)
    if(.not.dvalid(cln,cvdid,no_adc))goto 900    !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IE !ADC TO END
    IF(IS .GE. IE)then
        call smg$put_chars(cvdid,('LAST DETECTOR '//
        'SPECIFIED DOES NOT FOLLOW THE FIRST'),2,1)
        goto 900
    endif
    J=INDEX(LINE(1:llength),'')
    IF(J .EQ. 0)then    !Incomplete command.
330      call smg$put_chars(cvdid,('INCOMPLETE '//
        'COMMAND - COUNT TIME MISSING OR BAD'),2,1)
        goto 900
    endif
    READ(UNIT=LINE(J+1:llength),FMT=*,ERR=330)ISEC
    if(ifrota)then
        call slosyn(ifinit,'GOSL')    !Start rotating the sample.
        ifrogo=.true.
        call smg$put_chars(nvdid,'Y',6,79,,
        smg$m_reverse)
    endif
C
C START EACH DETECTOR SYSTEM FROM BEGINNING TO END
    DO I=IS,IE
        CLN='A1'
        WRITE(UNIT=CLN(2:2),FMT='(I1)')I
        if(as.on_off(i))goto 320    !Already counting on this detector!
        CALL START_COUNTING(ISEC,I,ERR)
        AS.SPECTRUM_ID(I)=' '
        IF(ERR)GOTO 320
        if(i.eq.1)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('DATA '//
            'COLLECTION STARTED ON ADC '),18,1)
        endif
        call smg$put_chars(nvdid,cln,18,(29+i*3))
    ENDDO
C
C ON A MULTIPLE START MAKE SURE THE START TIMES FOR ALL ADC'S ARE THE SAME
    DO I=IS+1,IE
        AS.START_DATE(I)=AS.START_DATE(IS)
        AS.START_TIME(I)=AS.START_TIME(IS)
    ENDDO

```



```

        ifrogo=.false.
        call smg$put_chars(nvdid,'N',6,79,,
            smg$m_reverse)
        endif
    GOTO 120
ELSE
    if(.not.dvalid(cln,cvdid,no_adc))goto 900    !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IS !ADC TO START
    CLN=LINE(J+1:J+2)
    if(.not.dvalid(cln,cvdid,no_adc))goto 900    !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IE !ADC TO END
    IF(IS .GE. IE)then
        call smg$put_chars(cvdid,('LAST DETECTOR '//
            'SPECIFIED DOES NOT FOLLOW THE FIRST'),2,1)
        goto 900
    endif
C
C STOP EACH DETECTOR SYSTEM FROM BEGINNING TO END
    DO I=IS, IE
        IF(AS.AUTO_COUNT(I))THEN
            CALL STOP_COUNTING(I,ERR)
            CLOSE(UNIT=SIDF)
            CALL LIB$FREE_LUN(SIDF)
        ELSE
            CALL STOP_COUNTING(I,ERR)
        ENDIF
        CLN='A1'
        WRITE(UNIT=CLN(2:2),FMT='(I1)')I
        IF(ERR)GOTO 340
        if(i.eq.1)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('DATA '//
                'COLLECTION STOPPED ON ADC '),18,1)
            endif
            call smg$put_chars(nvdid,cln,18,(29+i*3))
        ENDDO
        if(ifrota)then
            call slosyn(ifinit,'NG') !Stop rotating the sample.
            ifrogo=.false.
            call smg$put_chars(nvdid,'N',6,79,,
                smg$m_reverse)
            endif
    GOTO 120
ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C READ DATA OUT OF AN ADC SYSTEM AND STORE ON DISK
C   FORMAT: D;A1,<WEIGHT IN KG>,<CONTENT CODE>,<COMMENTS>
C
C   For a write-to-disk of all NO_ADC detectors, rewrite CMD and LINE to
C   an appropriate D command and let D handle the error checking.
C
      if(cmd(1:(ic-2)).eq.'BD')then
        if(ic.gt.11length)then
          STATUS=SMG$ERASE_LINE(CVDID,3,1)
          STATUS=SMG$RING_BELL(CVDID,2)
          STATUS=SMG$PUT_CHARS(CVDID,
1 'ERROR!! FORMAT IS "BD;<weight>,<content code>,<comments>"',2,1)
          GOTO 900
        endif
        line='D;A1-A'//char(no_adc+48)//','//line(ic:11length)
        11length=11length+5 !Changed "BD" to "D" (subtract 1) and added "A1-An,".
        cmd='D'
        ic=3
        endif
      IF(CMD(1:IC-2) .EQ. 'D')THEN
        CLN=LINE(IC:IC+1)
        if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
        READ(UNIT=LINE(4:4),FMT='(I1)')iadc !ADC NUMBER
C IS COMMAND A MULTIPLE DETECTOR READ?
        J=INDEX(LINE(1:IC+2),'-')
        IF(J .EQ. 0)THEN !No, so set JADC equal to IADC.
          jadc=iadc
        ELSE !Yes, so set JADC to the last detector's number.
          if(.not.dvalid((cln(1:1)//line(7:7)),cvdid,no_adc))goto 900 !Bad command
          READ(UNIT=LINE(7:7),FMT='(I1)')jadc !ADC NUMBER
        ENDIF
        do i=iadc,jadc !For processing all specified detectors.
          WRITE(UNIT=CLN(2:2),FMT='(I1)')I !ADC NUMBER
          if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
          if(i.ne.iadc)goto 410 !Parsing already completed.
C
C PARSE THE LINE TO GET THE OTHER PARAMETERS OUT OF THE COMMAND LINE
      IC=IC+2 !For a single detector.
      if(iadc.ne.jadc)ic=ic+3 !Three more for multiple detectors.
      IF(LINE(IC:IC) .NE. ',')THEN
400      STATUS=SMG$ERASE_LINE(CVDID,3,1)
          STATUS=SMG$RING_BELL(CVDID,2)
          STATUS=SMG$PUT_CHARS(CVDID,
1 'ERROR!! FORMAT IS "D;<det>,<weight>,<content code>,<comments>"',
2 1,1)
          STATUS=SMG$PUT_CHARS(CVDID,
1 '
          OR "BD;<weight>,<content code>,<comments>"',2,1)

```

```

                GOTO 900
ENDIF
IC=IC+1
J=INDEX(LINE(IC:llngth),',')
IF(J .EQ. 0)GOTO 400
J=IC+J-2
READ(LINE(IC:J),FMT=*,ERR=400)WKG

IC=J+2
J=INDEX(LINE(IC:llngth),',')
IF(J .EQ. 0)GOTO 405
J=IC+J-2
read(line(ic:j),fmt=*,err=405,end=405)incode
iccode=0    !In case no match is found.
do k=1,nccods    !Is this content code known?
    if(incode.eq.iccods(k))iccode=k    !Yes it is.
enddo
if(iccode.eq.0)goto 405    !No match found.

C DOES FILE EXIST?
    bt=atcoef(iccode)    !Accept the mass attenuation coefficient number.
INQUIRE(FILE=('AT$TABLE: '//bt//'.TBL'),EXIST=EX)
IF(.NOT. EX)THEN    !Invalid content code.
405      call smg$put_chars(cvdid,('INVALID CONTENT'//
        ' CODE'),2,1)
        goto 900
    endif

    IC=J+1

    IF(LINE(IC:IC) .NE. ',')GOTO 400
    IF(IC .GE. llngth)GOTO 400
    IC=IC+1

410      CALL READ_SPECTRUM(CLN,WKG,BT,LINE(IC:llngth),ERR)
    IF(ERR)THEN
        STATUS=SMG$ERASE_LINE(CVDID,3,1)
        STATUS=SMG$RING_BELL(CVDID,2)
        STATUS=SMG$PUT_CHARS(CVDID,
1 'UNABLE TO READ DATA FROM ADC '//CLN,3,1)
        else
            if(i.eq.1)then
                call smg$erase_line(nvdid,18,1)
                call smg$put_chars(nvdid,('DATA READ'//
                    ' AND SAVED TO FILE FROM ADC '),18,1)
                endif
                call smg$put_chars(nvdid,cln,18,(35+i*3))
    ENDIF

```



```

                STATUS=SMG$RING_BELL(CVDID,2)
                STATUS=SMG$PUT_CHARS(CVDID,
1 'ERROR!! FORMAT IS "DA;<det>,<weight>,<content code>,<comments>"',
2 1,1)
                STATUS=SMG$PUT_CHARS(CVDID,
1 '
                OR "BDA;<weight>,<content code>,<comments>"',2,1)
                GOTO 900
ENDIF
IC=IC+1
J=INDEX(LINE(IC:11length),',')
IF(J .EQ. 0)GOTO 420
J=IC+J-2
READ(LINE(IC:J),FMT=*,ERR=420)WKG

IC=J+2
J=INDEX(LINE(IC:11length),',')
IF(J .EQ. 0)GOTO 425
J=IC+J-2
read(line(ic:j),fmt=*,err=425,end=425)incode
iccode=0 !In case no match is found.
do k=1,nccods !Is this content code known?
    if(incode.eq.iccods(k))iccode=k !Yes it is.
enddo
if(iccode.eq.0)goto 425 !No match found.

C DOES FILE EXIST?
    bt=atcoef(iccode) !Accept the mass attenuation coefficient number.
    INQUIRE(FILE=('AT$TABLE: '//bt//'.TBL'),EXIST=EX)
    IF(.NOT. EX)THEN !Invalid content code.
425        call smg$put_chars(cvdid,('INVALID '//
        'CONTENT CODE'),2,1)
        goto 900
    endif

    IC=J+1

    IF(LINE(IC:IC) .NE. ',')GOTO 420
    IF(IC .GE. 11length)GOTO 420

    IC=IC+1

430    CALL READ_SPECTRUM(CLN,WKG,BT,LINE(IC:11length),ERR)
    IF(ERR)THEN
        STATUS=SMG$ERASE_LINE(CVDID,3,1)
        STATUS=SMG$RING_BELL(CVDID,2)
        STATUS=SMG$PUT_CHARS(CVDID,
1 'UNABLE TO READ DATA FROM ADC '//CLN,3,1)

```

```

        else
            if(i.eq.1)then
                call smg$erase_line(nvdid,18,1)
                call smg$put_chars(nvdid,('DATA READ'//
                ' AND SAVED TO FILE FROM ADC '),19,1)
            endif
            call smg$put_chars(nvdid,cln,19,(35+i*3))
        ENDIF
C
C SPAWN THE ANALYSIS
J=INDEX(AS.SPECTRUM_ID(I),'.DGE')-11
CALL SUBMIT('Y','N',AS.SPECTRUM_ID(I)(J:J+10),
1 ' ',' ',' ',' ')
        if(i.eq.1)then
            call smg$erase_line(nvdid,18,1)
            call smg$put_chars(nvdid,('SPAWNING '//
            'BATCH ANALYSIS FOR ADC '),18,1)
        endif
        call smg$put_chars(nvdid,cln,18,(30+i*3))

        if(i.eq.jadc)then
            call smg$erase_line(nvdid,19,1)
            call smg$erase_line(nvdid,20,1)
            call smg$put_chars(nvdid,('PRINTED OUTPUT '//
            'FROM THESE ANALYSES SHOULD APPEAR IN A '//
            'FEW MINUTES'),19,1)
            call smg$put_chars(nvdid,('IF '//
            'NONE IS FOUND, CHECK THE [OPERATOR]BATCH_ '//
            'GIDXC_ANALYSIS.LOG FILE FOR ERRORS'),20,1)
        endif
        enddo
        GOTO 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C READ DATA OUT OF AN ADC SYSTEM AND STORE IT IN THE BACKGROUND DIRECTORY
C FORMAT BKG;A1-A6
    IF(CMD(1:IC-2).EQ.'BKG')THEN
        CLN=LINE(IC:IC+1)
        J=INDEX(LINE(1:IC+2),'-')
C IS COMMAND A MULTIPLE DETECTOR READ?
        IF(J.EQ. 0)THEN
            if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
            READ(UNIT=CLN(2:2),FMT='(I1)')I !ADC NUMBER
C
C FOR THIS FUNCTION WILL SAVE THE CURRENT SAVE DIRECTORY IN CUR_SAVE
C SET IT TO THE BACKGROUND DIRECTORY FOR THIS DETECTOR SYSTEM

```

C

```

CUR_SAVE=AS.SAVE_DIR(I)
AS.SAVE_DIR(I)='BK$SAVE:'
CALL READ_SPECTRUM(CLN,0.001,'S','BACKGROUND',ERR)
AS.SAVE_DIR(I)=CUR_SAVE
IF(ERR)THEN
    STATUS=SMG$ERASE_LINE(CVDID,3,1)
    STATUS=SMG$RING_BELL(CVDID,2)
    STATUS=SMG$PUT_CHARS(CVDID,
1 'UNABLE TO READ BACKGROUND DATA FROM ADC '//CLN,3,1)
    else
        call smg$erase_line(nvdid,18,1)
        call smg$put_chars(nvdid,('BACKGROUND '//
        'DATA READ FROM ADC '//CLN//
        ' AND SAVED TO FILE'),18,1)
ENDIF
GOTO 120
ELSE
    if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IS !ADC TO START
    CLN=LINE(J+1:J+2)
    if(.not.dvalid(cln,cvdid,no_adc))goto 900 !Bad command
    READ(UNIT=CLN(2:2),FMT='(I1)')IE !ADC TO END
    IF(IS .GE. IE)then
        call smg$put_chars(cvdid,('LAST DETECTOR '//
        'SPECIFIED DOES NOT FOLLOW THE FIRST'),2,1)
        goto 900
    endif

```

C

```

C READ DATA FROM EACH DETECTOR SYSTEM FROM BEGINNING TO END
DO I=IS,IE
    CLN='A1'
    WRITE(UNIT=CLN(2:2),FMT='(I1)')I
    CUR_SAVE=AS.SAVE_DIR(I)
    AS.SAVE_DIR(I)='BK$SAVE:'
    CALL READ_SPECTRUM(CLN,0.001,
1 'S','BACKGROUND',ERR)
    AS.SAVE_DIR(I)=CUR_SAVE
    IF(ERR)THEN
        STATUS=SMG$ERASE_LINE(CVDID,3,1)
        STATUS=SMG$RING_BELL(CVDID,2)
        STATUS=SMG$PUT_CHARS(CVDID,
1 'UNABLE TO READ BACKGROUND DATA FROM ADC '//CLN,3,1)
        else
            if(i.eq.1)then
                call smg$erase_line(nvdid,18,1)
                call smg$put_chars(nvdid,

```

[illegible]


```

        ifrota=.not.ifrota
        if(ifrota)then
            call smg$put_chars(nvdid,'Y',7,79,,smg$m_reverse)
        else
            call smg$put_chars(nvdid,'N',7,79,,smg$m_reverse)
        endif
        GOTO 120
    ENDIF

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C INITIALIZE BARREL TURNER COMMAND
    IF(CMD(1:IC-2) .EQ. 'RI') THEN
C
C     Initialize the SLOSYN control and force a stop of barrel rotation so that
C     the current status is known.
C
        ifinit=.true.
        call slosyn(ifinit,'NG')
        ifrogo=.false.
        call smg$put_chars(nvdid,'N',6,79,,smg$m_reverse)
        GOTO 120
    ENDIF

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ANALYZE ALREADY STORED SPECTRUM
    IF(CMD(1:IC-2) .EQ. 'A') THEN
        if((llngth-ic).ne.10)then      !Not a valid spectrum name.
            call smg$put_chars(cvdid,('THE SPECIFIED SPECTRUM '//
            'NAME '//line(IC:llngth)//
            ' " IS NOT 11 CHARACTERS LONG'),2,1)
            goto 900
        endif
        OPEN(UNIT=LUN,NAME='GE$SAVE: '//line(IC:llngth) //' .DGE',
1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED,
2 IOSTAT=STATUS)
        if(status.ne.0)then
            if(status.eq.for$_filnotfou)then      !Try in OLDSWEPP$SAVE.
                OPEN(UNIT=LUN,NAME='OLDSWEPP$SAVE: '//
1 line(IC:llngth) //' .DGE',TYPE='OLD',ACCESS='DIRECT',
2 RECORDSIZE=128,READONLY,SHARED,IOSTAT=STATUS)
                if(status.eq.0)then      !Found it in OLDSWEPP$SAVE.
500
                    close(lun,status='keep')
                endif
            endif
        endif
        700
C
C SPAWN THE ANALYSIS
    CALL SUBMIT('Y','N',LINE(IC:LLNGTH),' ',' ',' ')

        call smg$erase_line(nvdid,18,1)

```

```

        call smg$erase_line(nvdid,19,1)
        call smg$put_chars(nvdid,('SPAWNING '//
        'BATCH ANALYSIS FOR SPECTRUM '//
        line(IC:llngth)//
        ' - LOOK FOR OUTPUT IN A '//
        'FEW MINUTES -'),18,1)
        call smg$put_chars(nvdid,('IF NONE '//
        'IS FOUND, CHECK THE [OPERATOR] '//
        'BATCH_GIDXC_ANALYSIS.LOG FILE FOR '//
        'ERRORS'),19,1)
        goto 120
    endif
endif
else !Found it in GE$SAVE.
    goto 700
endif
if(status.eq.for$_filnotfou)then !Not found either place.
    call smg$put_chars(cvdid,('THE SPECIFIED SPECTRUM "'//
    line(IC:llngth)///" DOES NOT EXIST'),2,1)
else
    call smg$put_chars(cvdid,('THE SPECIFIED SPECTRUM "'//
    line(IC:llngth)///" IS INVALID'),2,1)
endif
goto 900
ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C ANALYZE ALREADY STORED SET OF BARREL SPECTRA
    IF(CMD(1:IC-2) .EQ. 'FBA')THEN !Reset BATCH_CMD2's first parameter.
        YNO='Y'
        goto 704
    ENDIF
    IF(CMD(1:IC-2) .EQ. 'BA')THEN
        YNO='N'
704      DO I=1,4
          PAR(I)=' '
        ENDDO
        IPX=0
705      j=index(line(ic:llngth),',')
          if(j.eq.0)j=2+llngth-ic !Last spectrum to be included.

        IF(J .NE. 12)THEN
            IF((J .EQ. 3) .AND. (PAR(1) .EQ. ' '))THEN
                call smg$put_chars(cvdid,('A SPECIFIED SPECTRUM '//
                'NAME "'//line(ic:(ic+j-2))//
                '" IS NOT 11 CHARACTERS LONG'),2,1)
                goto 900
            
```

```

endif
ENDIF
    if(j.eq.12)then      !Full spectrum name given.
        OPEN(UNIT=LUN,NAME='GE$SAVE: '//line(IC:(ic+10))//'.DGE',
1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED,
2 IOSTAT=STATUS)
        else      !Just the slot, so use the last "end" part.
        OPEN(UNIT=LUN,NAME='GE$SAVE: '//line(IC:(ic+1))//
1 BSPID(3:11)//'.DGE',
2 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED,
3 IOSTAT=STATUS)
        endif
        if(status.ne.0)then
            if(status.eq.for$_filnotfou)then      !Try in OLDSWEPP$SAVE.
                if(j.eq.12)then      !Full spectrum name given.
                    OPEN(UNIT=LUN,NAME='OLDSWEPP$SAVE: '//
1 line(IC:(ic+10))//'.DGE',TYPE='OLD',ACCESS='DIRECT',
2 RECORDSIZE=128,READONLY,SHARED,IOSTAT=STATUS)
                    else      !Just the slot, so use the last "end" part.
                    OPEN(UNIT=LUN,NAME='OLDSWEPP$SAVE: '//
1 line(IC:(ic+1))//BSPID(3:11)//'.DGE',
2 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,READONLY,SHARED,
3 IOSTAT=STATUS)
                    endif
                    if(status.eq.0)then      !Found it in OLDSWEPP$SAVE.
710      close(lun,status='keep')
                        if(j.eq.12)then      !Full spectrum name given.
                            BSPID=LINE(IC:(IC+10))
                                else      !Just the slot, so use the last "end" part.

                            BSPID=LINE(IC:(IC+1))//BSPID(3:11)
                                endif
                            IPX=IPX+1
                            PAR(IPX)=BSPID
                                ic=ic+j      !To accomodate full id or slot alone.
                                if(ic.ge.11length)goto 715      !No more spectra to be included.
                                goto 705
                                endif
                            endif
                        else      !Found it in GE$SAVE.
                            goto 710
                        endif
                    if(status.eq.for$_filnotfou)then      !Not found either place.
                        call smg$put_chars(cvdid,('THE SPECIFIED SPECTRUM "'//
                            line(IC:(ic+10))//'" DOES NOT EXIST'),2,1)
                        else

```

```

        call smg$put_chars(cvdid,('THE SPECIFIED SPECTRUM "'//
        line(IC:(ic+10))//'" IS INVALID'),2,1)
    endif
    goto 900
C
C SPAWN THE ANALYSIS
C
715      CALL SUBMIT(YNO, 'Y', PAR(1), PAR(2), PAR(3), PAR(4))
        call smg$erase_line(nvdid,18,1)
        call smg$erase_line(nvdid,19,1)
        call smg$put_chars(nvdid,('SPAWNING '//
        'BATCH ANALYSIS FOR THESE SPECTRA'//
        ' - LOOK FOR OUTPUT IN '//
        'A FEW MINUTES -'),18,1)
        call smg$put_chars(nvdid,('IF NONE '//
        'IS FOUND, CHECK THE [OPERATOR]'//
        'BATCH_GIDXC_ANALYSIS.LOG FILE FOR '//
        'ERRORS'),19,1)
        goto 120
    ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C PU-REGION MONITOR TOGGLE/RESET COMMAND
    IF(CMD(1:IC-2) .EQ. 'M') THEN
        if((.not.ifmnr).and.(llngth.ge.2)) then
            if(llngth.eq.2) iatrgr=datrgr !Use the default trigger level.
            if(llngth.gt.2) then !New trigger level.
                read(line(3:llngth),fmt='(i<llngth-2>)',err=810,
                end=810) iatrgr !Too large or bad? Do not use it!
                if(iatrgr.le.0) goto 810 !Must be greater than zero.
            endif
            write(catrgr,fmt='(i10)',err=810) iatrgr
        endif
        do i=1,no_adc !Toggle monitoring flag if counting.
            if(as.on_off(i)) then !A count is in progress.
                ifmnr=.not.ifmnr
                goto 800
            endif
        enddo
800      if(ifmnr) then
            call smg$put_chars(nvdid, 'Y', 9, 79, , smg$m_reverse)
            call smg$put_chars(nvdid, catrgr, 14, 70, , smg$m_reverse)
        else
            call smg$put_chars(nvdid, 'N', 9, 79, , smg$m_reverse)
            call smg$erase_line(nvdid, 19, 1) !Erase the "estimated count length" message.
        endif
    GOTO 120

```



```

        ifauto=.false.
        goto 110      !To properly start ADCAST.
    endif
    GOTO 120
END

C
C Function to check for valid detector id.
C
    logical function dvalid(slot,cvdid,no_adc)
C
C Function checks whether the detector id passed in SLOT is valid, prints an
C error message if needed in the CVDID virtual display space, and returns a
C value of true if the id is valid, false if it is not.
C
C Declaration statements.
C
    character*2 slot
    integer*4 cvdid,no_adc
C
C Beginning of the function.
C
    dvalid=.false.      !By default, assume that the detector id is invalid.
C
C Check for a valid detector id:
C
C (1) only the letter A is acceptable and
C
    if(slot(1:1).ne.'A')then
        call smg$put_chars(cvdid,('UNKNOWN DETECTOR LETTER:  "'//
        .   slot(1:1)//'"'),2,1)
        return
    endif
C
C (2) only numbers between 1 and NO_ADC.
C
    read(slot(2:2),fmt='(i1)',err=100)islot
    if((islot.lt.1).or.(islot.gt.no_adc))then
100    call smg$put_chars(cvdid,('UNKNOWN DETECTOR NUMBER:  "'//
        .   slot(2:2)//'"'),2,1)
        return
    endif
C
C The detector id is valid.
C
    dvalid=.true.
    return
C

```

C End of the dvalid function.

C

end

B.2 GC Help Screen #1

COMMANDS AVAILABLE IN "GC" - SCREEN 1 OF 4

DATA ACQUISITION COMMANDS

M;<count limit>....Toggle whether Pu-count monitoring occurs during a count.

RI;.....Initialize the barrel turner (and stop rotation).

RT;.....Toggle whether sample rotation occurs during a count.

R;.....Manually start (or stop) sample rotation.

SP;<detector>.....Set the analysis parameters for the specified detector.

S;<detector(s)>,<sec>....

.....Start a count on the specified detector(s) for "sec" seconds.

BS;<container id>,<sec>....

.....Same as S, but for all detectors.

STOP;<detectors>...Stop data collection on the specified detector(s).

BSTOP;.....Same as STOP, but for all detectors.

B.3 GC Help Screen #2

COMMANDS AVAILABLE IN "GC" - SCREEN 2 OF 4

SPECTRUM STORAGE AND ANALYSIS COMMANDS

D;<detector(s)>,<weight in kg>,<content code>,<comments>....

.....Read the data out of each of the specified detectors and
store the spectrum on the disk.

BD;.....Same as D, but for all detectors.

DA;<detector(s)>,<weight in kg>,<content code>,<comments>....

.....Same as D, and then initiate an analysis of the spectrum.

BDA;.....Same as DA, but for all detectors.

BKG;<detector(s)>.....Read the data out of the specified detector(s) and store the spectrum(a) in the background directory on the disk.

A;<spectrum id>.....Analyze the specified, already stored spectrum.

BA;<spectrum id(s)>...Analyze the specified spectra (comma-delimited list) and generate a barrel summary report (the full id is required for the first spectrum; the detector id alone can be used to for subsequent spectra with otherwise identical ids).

FBA;<spectrum id(s)>..Same as BA, but with printed individual detector analyses.

B.4 GC Help Screen #3

COMMANDS AVAILABLE IN "GC" - SCREEN 3 OF 4

DETECTOR CALIBRATION COMMANDS

CBE;<detector>.....Energy calibrate the specified detector using a Eu-152 calibration source.

PEQE;<detector>....Energy calibrate the specified detector using a Eu-152 calibration source and determine pulser equivalent energies.

PEQEH;<detector>...Print the calibration history from PEQE for the specified detector.

CBT;<detector>.....Energy calibrate the specified detector using a Th-228 calibration source.

PEQT;<detector>....Energy calibrate the specified detector using a Th-228 calibration source and determine pulser equivalent energies.

PEQTH;<detector>...Print the calibration history from PEQT for the specified detector.

B.5 GC Help Screen #4

COMMANDS AVAILABLE IN "GC" - SCREEN 4 OF 4

SPECTRUM DISPLAY OPTIONS

DAS;<detector>(,<optional isotope library>)....
.....Create a spectrum display window.

BATCH JOB- AND QUEUE-RELATED COMMANDS

TYPELOG;.....Type on the monitor the last BATCH_GIDXC_ANALYSIS.LOG file.
PRINTLOG;.....Print the last BATCH_GIDXC_ANALYSIS.LOG file (on SYS\$PRINT).
SHOWQUE;.....Display on the monitor the status of all queues.

B.6 SLOSYN Subroutine

C MODIFIED TO USE "QIO'S" TO TALK TO THE MOTOR CONTROL INTERFACE
C EWK 6/23/94
C Subroutine to control and monitor rotation of a SLO-SYN SS2000D6 step motor
C drive equipped with a SS2000I motion control by copying a file containing the
C appropriate commands to the local printer port of a VAXstation 4000, model 60
C to which the control is connected; the port is expected to be pointed to by
C the logical \$SLOSYN. The use of printer controller mode, where the commands
C would be sent directly to the port, is not possible at this time owing to
C some presently unclear difficulties with consistently printing to a local
C printer port via a DECterm window.
C
C subroutine slosyn(ifinit,commnd)

C STRUCTURE /IOSTAT_BLOCK/ !QIO STATUS RETURN BLOCK
C INTEGER*2 IOSTAT,
C 1 TERM_OFFSET,
C 2 TERMINATOR,
C 3 TERM_SIZE
C END STRUCTURE
C RECORD /IOSTAT_BLOCK/ITOSB
C BYTE BIL(30)
C INTEGER*4 TLUN !TLUN IS THE LOGICAL UNIT ASSIGNED TO SERIAL OUT
C COMMON/AIMNO/NO_ADC,TLUN

C
C The control acts according to either "programs" that are entered into it or
C some "immediate" commands. After loading and starting the vendor-supplied
C program (and associated "L" parameters) for handling the door, jog, and off-

C balance cutoff switches (i.e., initializing the control), this routine will
 C make exclusive use of immediate commands. The feedrates used in this routine
 C assume that the manual step resolution rotary switch is set to 200 pulses per
 C motor revolution; at a feedrate of 1,000 pulses per second, this was measured
 C to provide 10 full rotations of the barrel platform every 5 minutes and 57
 C seconds, or 35.7 sec/rev, so the barrel step resolution is 35,700 pulses per
 C revolution (stored in STPRES).

C
 C Continuous rotation will be handled by setting the feedrate to the appropriate
 C value using an immediate form of the F command. Halting of rotation will be
 C handled by similarly setting the feedrate to zero. Fractional rotations will
 C be handled by starting rotation and then halting it after the appropriate
 C length of time for the given the rate of rotation. A padding blank space is
 C added to the end of each command as the command "terminator."

C
 C Owing to the presence of the image associated with ADCAST_GC, all LIB\$SPAWN
 C calls must include input and output files. The output files will be deleted
 C with each new execution of SLOSYN, along with any command files. The final
 C LIB\$SPAWN file copy to \$SLOSYN (before returning), for which the timing of
 C subsequent events is not important, is performed with the NOWAIT qualifier
 C to speed execution.

C
 C No capability has been implemented for monitoring for execution or error
 C statuses returned by the control, in part owing to the need to set up the
 C printer port with the NOTYPEAHEAD flag set, thus disabling any incoming
 C communications.

C
 C Abbreviated command's format and syntax (<number of command elements>a2):
 C element 1, characters 1-2: "GO" to start rotation, "NG" to stop
 C (default = NG);
 C 2, 3-4: "SL" for slow rotation, "FA" for fast
 C (default = FA);
 C 3, 5-6: "CO" for "continuous" (i.e., maximum time)
 C rotation, "DE" for an <element 4>-degree
 C rotation, "QU" for a quarter of a turn,
 C "TE" for a tenth of a turn (default=CO);
 C 4, 7-?: degrees for element 3 of "DE" (default=CO).

C
 C Initial coding completed by D. A. Femec, June 14, 1994.

C
 C Declaration statements.

C
 C INCLUDE '(\$SSDEF)'
 C INCLUDE '(\$TRMDEF)'
 C INCLUDE '(\$IODEF)'
 C INTEGER*4 SYS\$WAITFR, SYS\$QIO
 C INTEGER*4 SYS\$QIOW

```

include '($clidef)'      !CLI definitions.
character*(*) commnd      !Abbreviated command for directing control.
character*80 blanks
character*35 infile/'$user:[operator]slosyn_spawn_in.dat'/,
. outfile/'$user:[operator]slosyn_spawn_ot.dat'/
character*11 cwaitm/'00:00:00.00'/      !Delta time to wait for desired fractional rotation.
character*6 feedrt      !Rate at which steps are taken.
character*1 spaces(80)/80*' '/
real*4 stpres/35700.0/,      !Pulses/rev obtained via calibration.
. fedspd,slospd/1.0/,fstspd/3.0/,      !Slow and fast rev/min speeds.
. waitim,      !Time to wait before halting rotation for fractional rotations.
. degrees      !Translated degrees for element 3 of "DE."
integer*4 i,j,k,      !Common "DO" loop variables.
. outlun,      !Logical unit number for output file.
. lncmnd,      !Length of the character variable COMMND.
. lib$spawn,
. status      !Value returned from LIB$SPAWN function.
logical ifinit,      !Should the control be initialized?
. start      !True to start rotation, false to stop.
equivalence (blanks(1:1),spaces(1))

```

C

C Beginning of the subroutine.

C

C

C Get the control's attention and set it to active mode.

C

```

CHARACTER*(30) IL(18)
INTEGER*4      ILEN(18)
DATA IL(1) //'L06 013 '//,ILEN(1) /8/      !Program starts upon power up.
DATA IL(2) //'L20 01010 '//,ILEN(2) /10/      !TR1, TR2, and expansion port configuration.
DATA IL(3) //'L58 "+2187600000400" '//, ILEN(3) /21/      !Input configuration.
DATA IL(4) //'L72 2 '//,ILEN(4) /6/      !"S"-shaped velocity profile.
DATA IL(5) //'N001 G91 '//,ILEN(5) /9/      !Incremental mode.
DATA IL(6) //'N002 G64 '//,ILEN(6) /9/      !Reduce on.
DATA IL(7) //'N003 G20 X+22222221 F000007 '//,ILEN(7) /28/      !Branch on input(s) 22222221 to line 7.
DATA IL(8) //'N004 G20 X+22222212 F000007 '//,ILEN(8) /28/      !Branch on input(s) 22222212 to line 7.
DATA IL(9) //'N005 G12 X+00000001 '//,ILEN(9) /20/      !Go to line 1.
DATA IL(10) //'N006 G12 X+00000012 '//,ILEN(10) /20/      !Go to line 12.
DATA IL(11) //'N007 G48 X+ '//,ILEN(11) /12/      !Start program jog X CW.
DATA IL(12) //'N008 G20 X+22222200 F000010 '//,ILEN(12) /28/      !Branch on input(s) 22222200 to line 10.
DATA IL(13) //'N009 G12 X+00000008 '//,ILEN(13) /20/      !Go to line 8.
DATA IL(14) //'N010 G49 '//,ILEN(14) /9/      !Stop program jog mode.
DATA IL(15) //'N011 G12 X+00000003 '//,ILEN(15) /20/      !Go to line 3.
DATA IL(16) //'N012 G30 '//,ILEN(16) /9/      !Program end.

```

C

```

CHARACTER*40 OUTLN

```

```
CODE=IO$_WRITEVBLK
ISPACE="40
```

```
CALL QIOWL(TLUN,'<00 ')
```

```
C
C Initialize the control, if needed.
```

```
C      if(ifinit)then
```

```
C
C      Load the vendor-supplied "L" parameters and program, and start it. (The
C      comments following the lines of the program are those supplied by the
C      vendor.)
```

```
C      "L" parameters (those supplied by the vendor that appear necessary, are
C      not just the default, and do not conflict with those determined by this
C      routine):
```

```
C      DO I=1,16
C          CALL QIOWL(TLUN,IL(I)(1:ILEN(I)))
C      ENDDO
```

```
C
C      Set the maximum (FSTSPD rev/min, L71) and low (SLOSPD rev/min, L12) speed
C      limits.
```

```
C      OUTLN='L71 '
C          write(OUTLN(5:10),fmt='(i6.6)')int(stpres*fstspd/60.0)
C      CALL QIOWL(TLUN,OUTLN(1:11))
```

```
C      OUTLN='L12 '
C          write(OUTLN(5:10),fmt='(i6.6)')int(stpres*slospd/60.0)
C      CALL QIOWL(TLUN,OUTLN(1:11))
```

```
C
C      Disable initialization since it has now been performed.
```

```
C          ifinit=.false.
C      endif
```

```
C
C      Parse the abbreviated command - first obtain the length of COMMND.
```

```
C      lncmnd=len(commnd)
```

```
C
C      First two characters indicate whether to start or stop rotation.
```

```
C      if(lncmnd.ge.2)then
C          start=commnd(1:2).eq.'GO'      !"GO" to start, "NG" (default) to stop.
```

```

else !The default is to stop.
  start=.false.
endif

```

C
C
C

```

Start or stop rotation.

```

```

if(start)then !Start rotation.
  if(lncmnd.ge.4)then
    if(commnd(3:4).eq.'SL')then !Requested speed is slow.
      fedspd=slospd
    else !Default speed is fast.
      fedspd=fstspd
    endif
  else !Default speed is fast.
    fedspd=fstspd
  endif
  waitim=0.0 !Default wait time (i.e., do not wait)
  if(lncmnd.ge.6)then
    if((commnd(5:6).eq.'TE').or.(commnd(5:6).eq.'QU').or.
      (commnd(5:6).eq.'DE'))then
      if(commnd(5:6).eq.'TE')then !A tenth of a revolution.
        waitim=60.0 !Take sixty seconds to do the turn.
        fedspd=0.1 !A tenth of a turn in one minute.
      endif
      if(commnd(5:6).eq.'QU')then !A quarter of a revolution.
        waitim=60.0 !Take sixty seconds to do the turn.
        fedspd=0.25 !A quarter of a turn in one minute.
      endif
      if(commnd(5:6).eq.'DE')then
        if(lncmnd.ge.7)then
          read(commnd(7:lncmnd),*,err=200)degrees !Any errors? Keep the default.
          waitim=60.0 !Take sixty seconds to do the turn.
          fedspd=degrees/360.0 !An n-degree turn in one minute.
        endif
      endif
    endif
  endif
endif
endif
endif

```

200

```

CALL QIOWL(TLUN,'G69 ')

```

```

OUTLN='!F'
write(OUTLN(3:8),fmt='(i6.6)')int(stpres*fedspd/60.0)
CALL QIOWL(TLUN,OUTLN(1:9))

```

```

if(waitim.gt.0.0)then !Fractional rotation, . . .
  waitim=waitim*60.0 !Convert from minutes to seconds.
  write(cwaitm(1:2),fmt='(i2.2)')int(waitim/3600.0) !Hours.

```

```

        write(cwaitm(4:5),fmt='(i2.2)')int(mod(waitim,3600.0)/60)    !Minutes.
        write(cwaitm(7:11),fmt='(f5.2)')mod(waitim,60.0)    !Seconds.
        if(mod(waitim,60.0).lt.10.0)cwaitm(7:7)='0'    !Add the missing zero.

        CALL MARK(3, WAITIM)
        STATUS=SYS$WAITFR(%VAL(3))

        CALL QIOWL(TLUN,'!F0 ')    ! . . . then stop.
        CALL QIOWL(TLUN,'G68 ')    !Turn the motor windings off to allow hand rotation.
        endif
    else    !Stop rotation.
        CALL QIOWL(TLUN,'!F0 ')    !Feedrate to zero will stop rotation.
        CALL QIOWL(TLUN,'G68 ')    !Turn the motor windings off to allow hand rotation.
    endif
C
    return
C
C End of the subroutine SLOSYN.
C
    end

```

B.7 QIOWL Subroutine

```

C SUBROUTINE TO WRITE A BYTE STRING TO TERMINAL
C      EWK 6/23/94
C
C      SUBROUTINE  QIOWL(LUN,STRING)
C
C WHERE: LUN TERMINAL LOGICAL NUMBER
C      STRING IS THE STRING TO SEND
C
C      INCLUDE          '($SSDEF)'
C      INCLUDE          '($TRMDEF)'
C      INCLUDE          '($IODEF)'
C
C      INTEGER*4    STATUS,SYS$QIO
C      INTEGER*4    LUN, TLUN
C      CHARACTER*(*)    STRING
C
C      STRUCTURE /IOSTAT_BLOCK/    !QIO STATUS RETURN BLOCK
C          INTEGER*2    IOSTAT,
C      1                TERM_OFFSET,
C      2                TERMINATOR,
C      3                TERM_SIZE
C      END STRUCTURE

```

```

C      RECORD /IOSTAT_BLOCK/ITOSB

      BYTE BIL(100)

      CODE=IO$_WRITEVBLK
      ISPACE="44

      LSTR=LEN(String)
      TLUN=LUN
      DO J=1,LSTR
         BIL(J)=ICHAR(String(J:J))
      ENDDO
      STATUS=SYS$QIO(,%VAL(TLUN),%VAL(IO$_WRITEVBLK),ITOSB,,,
1         BIL,%VAL(LSTR),,%VAL(ISPACE),,)
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

      RETURN
      END

```

B.8 DIAGLN Subroutine

```

      subroutine diagln(ivideo,irow,icol,jrow,jcol)
C
C Subroutine to draw a step-wise diagonal line from position (IROW,ICOL)
C to (JROW,JCOL) on virtual display IVIDEO using SMG$DRAW_LINE.
C
C Coded by D. A. Femec, April 7, 1993.
C
C Declaration statements.
C
      real*4 colcur,slope
      integer*4 ivideo,irow,icol,jrow,jcol,rowbeg,colbeg,rowend,colend
C
C Beginning of the subroutine.
C
C If ROWEND < ROWBEG, swap beginning and ending for both rows and columns
C (rather than keep tracking of the appropriate signs below).
C
      if(jrow.lt.irow)then
         rowbeg=jrow
         rowend=irow
         colbeg=jcol
         colend=icol
      else

```

```

        rowbeg=irow
        rowend=jrow
        colbeg=icol
        colend=jcol
    endif
C
C Calculate the slope of the line.
C
        slope=-real(colend-colbeg)/real(rowbeg-rowend)
C
C Draw the line, step-wise, one step at a time.
C
        colcur=real(colbeg)
        do i=rowbeg, (rowend-1)
            call smg$draw_line(ivideo,i,int(colcur),i,nint(colcur+slope))
            call smg$draw_line(ivideo,i,nint(colcur+slope),(i+1),
                . nint(colcur+slope))
            colcur=real(nint(colcur+slope))
        enddo
        if(nint(colcur).ne.colend) call smg$draw_line(ivideo,rowend,
            . int(colcur),rowend,colend)
        return
C
C End of the DIAGLN subroutine.
C
        end

```

B.9 ADCAST Subroutine

```

C MOIFIED TO ELIMINATE USE OF LIB$SPAWN COMMANDS AND THERE BY ELIMINATE THE
C EATING UP OF MEMORY.
C         EWK 6/24/94
C
C ROUTINE TO UPDATE THE VIRTURAL DISPLAY NVDID WITH THE
C STATUS OF THE ADC'S FOR THE WASTE RETRIEVAL SYSTEM
C
C         EWK 11/1/89
C
C Modified to pass the number of ADCs (NO_ADC) to the routine via the common
C block AIMNO rather than the number of AIMs, and to use this value in setting
C up the AST properly. D. A. Femec, April 6, 1993.
C
C GC version created (code in file ADCAST_GC.FOR): if monitoring the region
C between channels ISTART and IEND, as indicated by IFMNTR, this version will
C also update the "counts in region" field in NVDID; the number of counts is

```


C obtained via the function COUNTS. Changed name of analysis routine being
 C called to BATCH_GIDXC_ANALYSIS (since the various BATCH_A<n> routines could
 C be consolidated into a single one); all of the spectra to be analyzed are
 C passed to the analysis routine in a single SPAWN (the /PARAMETER qualifier
 C was moved to the end of the BATCH_CMD2 command string and spectrum ids are
 C now appended to this command string to allow for multiple analyses - see
 C also the routine BATCH_GIDXC_ANALYSIS.COM) and the LIB\$SPAWN input file is
 C now named GIDXC.INPUT and the output file is now named GIDXC.OUTPUT).
 C Changed name of directory for various output to OPERATOR. Expanded the
 C READ_SPECTRUM call to include the barrel-related parameters of weight in
 C kilograms, matrix type, and any other comments (all of these are passed to
 C ADCAST_GC via the common block BARREL). Added notifications to the
 C "MESSAGES" area as to spectra being written to file, batch and print jobs
 C being submitted, and estimated duration of count for reaching the trigger
 C count level. Added "N" as the first parameter in BATCH_CMD so that no
 C container summary report is generated for analyses processed via AUTO START
 C (see BATCH_GIDXC_ANALYSIS.COM). Will check for the possible presence of a
 C lead liner based on the ratio of the number of counts in the region around
 C 413.71 keV (411.7-415.7 keV) to the number of counts in the region around
 C 129.30 (127.3-131.3 keV) for the spectrum with the greatest number of counts
 C in the trigger region. Inserted a new first parameter into BATCH_CMD and
 C BATCH_CMD2 to specify whether the individual detector analyses should be
 C printed (see also BATCH_GIDXC_ANALYSIS.COM) - for BATCH_CMD-requested
 C analyses, this parameter is "Y"; for BATCH_CMD2-requested analyses, this
 C parameter is "N" (also see GC). D. A. Femec, March 22, 1994.

C
 C Added code to resend the appropriate barrel turner command on each pass (to
 C handle stopping the barrel turner when GC says it should not be rotating but
 C the vendor program's wants to start rotation again because an opened door has
 C been closed, for example). D. A. Femec, May 17, 1994.

C SUBROUTINE ADCAST

C
 C Declaration statements.

C
 C
 C implicit none !For safety's sake.
 C integer*4 i,j,k !Common "DO" loop variables.
 C INCLUDE '(\$SMGDEF)'
 C INCLUDE 'GAP\$SRC:ADC_STATUS.INC/LIST'
 C INCLUDE 'GAP\$SRC:VTERM.INC'
 C INCLUDE 'SMAN.INC'
 C INCLUDE 'ASTTRM.INC'

C
 C
 C CHARACTER*80 CHARS !HOLD INFORMATION FOR DISPLAY
 C CHARACTER*80 TEMP_MESS
 C LOGICAL STATUS_ERR(6),ERR

```

LOGICAL      AUTO_START_ERR(6) /6*.FALSE./
INTEGER*4    SYSS$WAITFR,LIB$SPAWN
INTEGER*4    FLAG,JJ,IOS
INTEGER*4    DS(6) !CONTAINS ADC NUMBERS FOR AUTO START
C CHARACTER*300  BATCH_CMD2  //'SUBMIT/NOPRINTER/NOIDENTIFY
C 1/NONOTIFY/LOG_FILE=$USER:[OPERATOR]
C 2 INEL$EXE:BATCH_GIDXC_ANALYSIS/PARAMETER=(N,Y,'/
C CHARACTER*200  BATCH_CMD  //'SUBMIT/NOPRINTER/NOIDENTIFY
C 1/NONOTIFY/PARAMETER=(Y,N,A1040211375)/LOG_FILE=$USER:[OPERATOR]
C 2INEL$EXE:BATCH_GIDXC_ANALYSIS'/
integer*2 ibcend  !Next character to be written to in BATCH_CMD2.
CHARACTER*40    OUTFILE      //' $USER:[OPERATOR]GIDXC.OUTPUT'/
CHARACTER*40    INFILE       //' $USER:[OPERATOR]GIDXC.INPUT'/
INTEGER*4    MESS_NO      /1/  !POSITION TO LOOK FOR NEXT ERROR MESS
BYTE         ASTFLG       !DISCONTINUE UPDATE FLAG
CHARACTER*11    SP_ID
CHARACTER*11    P(4)

C
C STRUCTURE TO HOLD LAST DISPLAYED VALUES
  STRUCTURE /DISPLAY/
    LOGICAL      ONOFF(6) /2,2,2,2,2,2/
    INTEGER*4    REAL(6)
    INTEGER*4    LIVE(6)
    INTEGER*4    INTERVAL(6)
    CHARACTER*11  ID(6)  //'A1110189001','A2110189001',
1    'A3110189001','A4110189001','A5110189001','A6110189001'/

  END STRUCTURE

C
  RECORD /DISPLAY/D

C
  COMMON /AST/ASTFLG

C
C NUMBER OF ADC MODULES
  INTEGER*4    TLUN !CHANNEL FOR TERMINAL ATTACHED TO BARREL TURNER
  INTEGER*4    no_adc
  COMMON /AIMNO/no_adc,TLUN

C
C If-monitoring-region flag, start and ending energies and channels of
C region to be monitored, the COUNTS function, an array to contain the
C results from the COUNTS function calls, and a character version of the
C maximum counts value obtained.
C
  include 'gap$src:calibration.inc'
  character*10 cntsmx,  !Character version of counts from trigger region.
  . ccntln,  !Character version of estimated count length based on trigger level.

```

```

. cpbrat    !Character version of the peak area ratio for the lead-liner test.
real*4 estart/144.2/,eend/148.5/,    !Pu trigger region for count length.
. pbstrt(2)/411.7,127.3/,pbend(2)/415.7,131.3/    !Regions for lead-liner test.
integer*4 counts,adcnts(6)/6*0/,pbcnts(2,6),istart,iend,ipucnt,
. ipbcnt(2),iatrgr,icntln
logical ifinit,    !Should the control be initialized?
. ifrota/.true./,    !Should the sample be rotated during a count?
. ifrogo,    !Is the sample rotating?
. ifmntr,    !Is the Pu region being monitored for when to analyze?
. ifaoff    !Are all the detectors off?
common /counts/ ifinit,ifrota,ifrogo,ifmntr,istart,iend,iatrgr

```

```

C
C READ_SPECTRUM-related parameters for barrel counts.
C

```

```

character*100 commnt    !User-supplied comment.
character*2 bt    !Barrel matrix type.
real*4 wkg    !Weight in kilograms.
common /barrel/ wkg,bt,commnt

```

```

C
C DISCONTINUE UPDATES?
IF(ASTFLG.EQ.0)RETURN

```

```

C
STATUS=SMG$BEGIN_DISPLAY_UPDATE(NVDID)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
STATUS=SMG$BEGIN_DISPLAY_UPDATE(CVDID)
IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

```

```

C
C GET STATUS OUT OF EACH REMOTE
DO I=1,no_adc
CALL READ_STATUS(I,STATUS_ERR(I))
IF(STATUS_ERR(I))THEN
STATUS=SMG$RING_BELL(NVDID,2)
CHARS='A1'
WRITE(UNIT=CHARS(2:2),FMT='(I1)')I
TEMP_MESS='UNABLE TO READ STATUS FROM DETECTOR '
1 //CHARS(1:2)
CALL SET_ERR_MSG(TEMP_MESS)
ENDIF
100 ENDDO

```

```

C
C Resend the appropriate barrel turner command.
C

```

```

if(ifrogo)then
call slosyn(ifinit,'GO')    !Stop rotating the sample.
call smg$put_chars(nvdid,'Y',6,79,,smg$m_reverse)
else
call slosyn(ifinit,'NG')    !Stop rotating the sample.

```

```

        call smg$put_chars(nvdid, 'N', 6, 79, , smg$m_reverse)
    endif

C
C START UP DATE OF DISPLAY
C
C*****
C DISPLAY DETECTOR ON-OFF
    K=14
    ifaoff=.true.    !Reset the all-detectors-off flag to assume all are off.
    DO I=1,no_adc
        if(ifmntr)then
            adcnts(i)=0    !Zero the number of counts in the regions.
            pbcnts(1,i)=0
            pbcnts(2,i)=0
            ifaoff=ifaoff.and.(.not.as.on_off(i))    !Any on makes this false.
        endif
        IF(AS.ON_OFF(I) .NE. D.ONOFF(I))THEN
            FLAG=SMG$M_REVERSE
            IF(STATUS_ERR(I))FLAG=FLAG.OR.SMG$M_BLINK !TURN BLINK ON FOR
                !QUESTIONABLE LINES
            IF(AS.ON_OFF(I))THEN
                STATUS=SMG$PUT_CHARS(NVDID, 'ON ', 6, K, , FLAG)
            ELSE
                STATUS=SMG$PUT_CHARS(NVDID, 'OFF', 6, K, , FLAG)
            ENDIF
            D.ONOFF(I)=AS.ON_OFF(I)
        ENDIF
    ENDIF

C
C If the detector is on, convert the energies ESTART and EEND to channels
C given the proper calibration parameters, then get the number of counts in
C the specified region for that detector.
C
    if(ifmntr.and.as.on_off(i))then    !If detector is on, get the counts.
        1start=((-gcb.b(i)+sqrt(gcb.b(i)**2-4*gcb.q(i)*(gcb.a(i)-
        .   estart)))/(2.0*gcb.q(i)))-gcb.czero(i)    !Only root that could be positive.
        iend=((-gcb.b(i)+sqrt(gcb.b(i)**2-4*gcb.q(i)*(gcb.a(i)-
        .   eend)))/(2.0*gcb.q(i)))-gcb.czero(i)    !Only root that could be positive.
        adcnts(i)=counts(i,istart,iend)
        do j=1,2
            1start=((-gcb.b(i)+sqrt(gcb.b(i)**2-4*gcb.q(i)*(gcb.a(i)-
            .   pbstrt(j)))/(2.0*gcb.q(i)))-gcb.czero(i)    !Only root that could be positive.
            iend=((-gcb.b(i)+sqrt(gcb.b(i)**2-4*gcb.q(i)*(gcb.a(i)-
            .   pbend(j)))/(2.0*gcb.q(i)))-gcb.czero(i)    !Only root that could be positive.
            pbcnts(j,i)=counts(i,istart,iend)
        enddo
    endif

```

```

        K=K+12
    ENDDO
C
C If monitoring, update the "counts in region" display field and the "estimated
C count length" message, . . .
C
    if(ifmntr)then
        ipucnt=0
        do i=1,no_adc !Get the largest number of counts in the trigger region . . .
            if(adcnts(i).gt.ipucnt)then
                ipucnt=adcnts(i)
                icntln=as.real_time(i) ! . . . and the respective count length.
                ipbcnt(1)=pbcnts(1,i)
                ipbcnt(2)=pbcnts(2,i)
            endif
        enddo
        write(cntsmx,fmt='(i10)')ipucnt
        call smg$put_chars(nvdid,cntsmx,12,70,,smg$m_reverse)
        call smg$erase_line(nvdid,19,1)
        if(ipucnt.ne.0)then !To avoid a divide-by-zero error, in part.
            write(ccntln,fmt='(i10)',iostat=ios)
            int((real(icntln)/real(ipucnt))*real(iatrgr))
            call smg$put_chars(nvdid,('ESTIMATED COUNT LENGTH TO '//
            'REACH ANALYSIS TRIGGER LEVEL = '//ccntln//' SEC'),19,1)
        else
            call smg$put_chars(nvdid,('ESTIMATED COUNT LENGTH TO '//
            'REACH ANALYSIS TRIGGER LEVEL NOT CALCULABLE'),19,1)
        endif
        call smg$erase_line(nvdid,20,1)
        if(ipbcnt(2).gt.100)then !To avoid a divide-by-zero error, in part.
            write(cpbrat,fmt='(i10)',iostat=ios)
            int(real(ipbcnt(1))/real(ipbcnt(2)))
            if((ipbcnt(1)/ipbcnt(2)).gt.10)then !Lead liner likely present.
                call smg$put_chars(nvdid,('LEAD LINER LIKELY '//
                'PRESENT: 413.7-keV:129.3-keV = '//cpbrat//
                ' (I.E., > 10)'),20,1)
            else
                call smg$put_chars(nvdid,('LEAD LINER LIKELY '//
                'NOT PRESENT: 413.7-keV:129.3-keV = '//cpbrat//
                ' (I.E., < 10)'),20,1)
            endif
        else
            call smg$put_chars(nvdid,('THE PRESENCE OF A LEAD '//
            'LINER CANNOT YET BE RELIABLY DETERMINED'),20,1)
        endif
    endif
C
C . . . and if it exceeds the limit, stop all counts, save the spectra, and

```

```

C   initiate the analyses. Or, if the count time has elapsed, save the
C   spectra and initiate the analyses.
C
      if((ipucnt.gt.iatrgr).or.ifaoff)then
        call smg$erase_line(nvdid,19,1) !Erase the "estimated count length" message.
        ifmntr=.false. !No need to continue monitoring.
C       ibcend=index(batch_cmd2,'=')+6 !Find the position after /PARAMETER=(Y,.
        do i=1,no_adc
          chars(1:1)='A'
          WRITE(CHARS(2:2),FMT='(I1)')I
          if(as.on_off(i))then !If the detector is still on.
            call stop_counting(i,err)
            IF(ERR)THEN
              STATUS=SMG$RING_BELL(NVDID,2)
              TEMP_MESS='UNABLE TO STOP DETECTOR '
1              //CHARS(1:2)
              CALL SET_ERR_MSG(TEMP_MESS)
              GOTO 200
            else
              if(i.eq.1)then
                call smg$erase_line(nvdid,18,1)
                call smg$put_chars(nvdid,('DATA '//
                  'COLLECTION STOPPED ON ADC '),18,1)
                endif
                call smg$put_chars(nvdid,chars(1:2),18,
                  (29+i*3))
              endif
            as.on_off(i)=.false.
            endif
          CALL READ_SPECTRUM(CHARS(1:2),wkg,bt,commnt,ERR)
          IF(ERR)THEN
            STATUS=SMG$RING_BELL(NVDID,2)
            TEMP_MESS='UNABLE TO READ SPECTRUM FROM DETECTOR'
1            //CHARS(1:2)
            CALL SET_ERR_MSG(TEMP_MESS)
            GOTO 200
          else
            if(i.eq.1)then
              call smg$erase_line(nvdid,18,1)
              call smg$put_chars(nvdid,('DATA READ AND '//
                'SAVED TO FILE FROM ADC '),18,1)
              endif
              call smg$put_chars(nvdid,chars(1:2),18,(35+i*3))
            endif
          ENDIF
        ENDIF
C
C SPAWN ANALYSIS
      J=INDEX(AS.SPECTRUM_ID(I),'.DGE')-11

```

```

C          if(i.ne.1)then      !Add a comma.
C              batch_cmd2(ibcend:ibcend)=','
C              ibcend=ibcend+1
C              endif
C          BATCH_CMD2(ibcend:(ibcend+10))=AS.SPECTRUM_ID(I) (J:J+10)
C              ibcend=ibcend+11
C          P(I)=AS.SPECTRUM_ID(I) (J:J+10)
200      enddo
C          batch_cmd2(ibcend:ibcend)='')      !Add the ending parenthesis.
C          STATUS=LIB$SPAWN(BATCH_CMD2(1:ibcend), INFILE,OUTFILE)
C          IF(.NOT. STATUS)THEN
C              STATUS=SMG$RING_BELL(NVDID,2)
C              TEMP_MESS='UNABLE TO SPAWN ANALYSIS FROM DETECTORS'
C              CALL SET_ERR_MSG(TEMP_MESS)
C              else
C              CALL SUBMIT('N','Y',P(1),P(2),P(3),P(4))
C                  call smg$erase_line(nvdid,19,1)
C                  call smg$erase_line(nvdid,20,1)
C                  call smg$put_chars(nvdid,('SPAWNING '//
C                      'BATCH ANALYSIS FOR THESE SPECTRA'//
C                      ' - LOOK FOR OUTPUT IN '//
C                      'A FEW MINUTES -'),19,1)
C                  call smg$put_chars(nvdid,('IF NONE '//
C                      'IS FOUND, CHECK THE [OPERATOR]'//
C                      'BATCH_GIDXC_ANALYSIS.LOG FILE FOR '//
C                      'ERRORS'),20,1)
C              ENDIF
C          if(ifrota)then
C              call slosyn(ifinit,'NG')      !Stop rotating the sample.
C              ifrogo=.false.
C              call smg$put_chars(nvdid,'N',6,79,,smg$m_reverse)
C              endif
C          endif
C      endif
C
C*****
C DISPLAY REAL TIME
C      K=11
C      DO I=1,no_adc
C          IF(AS.REAL_TIME(I) .NE. D.REAL(I))THEN
C              FLAG=SMG$M_REVERSE
C              IF (STATUS_ERR(I)) FLAG=FLAG.OR.SMG$M_BLINK !TURN BLINK ON FOR
C                  !QUESTIONABLE LINES
C              WRITE(UNIT=CHARS,FMT=' (I9) ',IOSTAT=IOS)AS.REAL_TIME(I)
C              STATUS=SMG$PUT_CHARS(NVDID,CHARS(1:9),
C                  10,K,,FLAG)
C              D.REAL(I)=AS.REAL_TIME(I)

```

```

      ENDIF
      K=K+12
    ENDDO

C *****
C DISPLAY LIVE TIME
  K=11
  DO I=1,no_adc
    IF (AS.LIVE_TIME(I) .NE. D.LIVE(I)) THEN
      FLAG=SMG$M_REVERSE
      IF (STATUS_ERR(I)) FLAG=FLAG.OR.SMG$M_BLINK !TURN BLINK ON FOR
        !QUESTIONABLE LINES
      WRITE (UNIT=CHARS, FMT=' (I9) ', IOSTAT=IOS) AS.LIVE_TIME(I)
      STATUS=SMG$PUT_CHARS (NVDID,CHARS(1:9),
        8,K,,FLAG)
      D.LIVE(I)=AS.LIVE_TIME(I)
    ENDIF
    K=K+12
  ENDDO

C *****
C DISPLAY COUNTING INTERVAL
  K=12
  DO I=1,no_adc
    IF (AS.COUNT_INTERVAL(I) .NE. D.INTERVAL(I)) THEN
      WRITE (UNIT=CHARS, FMT=' (I6) ',
        IOSTAT=IOS) AS.COUNT_INTERVAL(I)
      STATUS=SMG$PUT_CHARS (NVDID,CHARS(1:6),
        12,K,,SMG$M_REVERSE)
      D.INTERVAL(I)=AS.COUNT_INTERVAL(I)
    ENDIF
    K=K+12
  ENDDO

C *****
C DISPLAY LAST SPECTRUM ID READ
  K=10
  DO I=1,no_adc
    IF (AS.SPECTRUM_ID(I) .EQ. ' ') THEN
      STATUS=SMG$PUT_CHARS (NVDID,AS.SPECTRUM_ID(I)(1:11),
        14,K)
      D.ID(I)=AS.SPECTRUM_ID(I)(1:11)
      GOTO 150
    ENDIF
    J=INDEX(AS.SPECTRUM_ID(I),':')+1
    IF (J .LE. 1) THEN
      AS.SPECTRUM_ID(I)=' '
    ENDIF
  ENDDO

```



```

        GOTO 140
    ENDIF
    JJ=INDEX(AS.SPECTRUM_ID(I),'.DGE')-1
    IF((JJ-J).NE.10)GOTO 145
    SP_ID=AS.SPECTRUM_ID(I)(J:JJ)
    IF(SP_ID.NE.D.ID(I))THEN
        STATUS=SMG$PUT_CHARS(NVDID,SP_ID,14,K,,SMG$M_REVERSE)
        D.ID(I)=SP_ID
    ENDIF
150    K=K+12
    ENDDO
C
C
C *****
C DISPLAY MESSAGE
    DO I=MESS_NO,10
        IF(AS.MESSAGE_AVAILABLE(I))THEN
C
C CLEAR MESSAGE ON SCREEN
            STATUS=SMG$ERASE_LINE(NVDID,20,10)
            STATUS=SMG$SET_CURSOR_ABS(NVDID,20,10)
            IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
            STATUS=SMG$PUT_LINE(NVDID,AS.MESSAGE(I)(1:70),
1                0,SMG$M_REVERSE)
            IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
            AS.MESSAGE_AVAILABLE(I)=.FALSE.
C
C ANY OTHER MESSAGES TO DISPLAY
            DO J=I+1,10
                IF(AS.MESSAGE_AVAILABLE(J))THEN
                    MESS_NO=J !DISPLAY THIS MESSAGE NEXT TIME
                    GOTO 5000
                ENDIF
            ENDDO
            MESS_NO=1
            GOTO 5000
        ENDIF
    ENDDO
    MESS_NO=1
C
C *****
C END OF DISPLAY UPDATE
5000 STATUS=SMG$END_DISPLAY_UPDATE(CVDID)
    STATUS=SMG$END_DISPLAY_UPDATE(NVDID)
    STATUS=SMG$SET_CURSOR_ABS(CVDID,4,5)
    STATUS=SMG$FLUSH_BUFFER(PBID)
C

```

```

C HAVE DISPLAY UPDATED AGAIN IN 2 SECONDS
  CALL ASTIME('2')
  RETURN
END

```

B.10 READ_STATUS Subroutine

```

C
C MODIFIED FOR THE SYMPLIFIED AIM ROUTINES
C   EWK 1/18/90
C
C ROUTINE TO READ STATUS OF ADC TO GET INFORMATION: LIVE AND REAL TIMES,
C AND ON-OFF STATUS.
C
C   SUBROUTINE READ_STATUS(RN,ERR)
C
C WHERE:
C   RN=NUMBER OF REMOTE IN REMOTE_DATA STRUCTURE
C   ERR IS A LOGICAL INDICATING AN ERROR ACCURED DURING THE OPERATION
C
C   EWK 10/10/88
C
C Modified to count over 8192 points rather than only 4096 points (and so the
C starting address for the ADC was changed from 16384 to 32768). D. A. Femec,
C March 1, 1993.
C
C   INCLUDE          'NDFUNC.INC'
C   INCLUDE          'gap$src:ADC_STATUS.INC/LIST'
C
C   INTEGER*4  RN
C   LOGICAL    ERR
C   INTEGER*4  STATUS,SYS$SETAST
C   BYTE       OFF  /0/
C   BYTE       ON   /1/
C
C
C DISABLE AST DELIVERY WHILE WE ARE HERE
  ERR=.FALSE.
  STATUS=SYS$SETAST(%VAL(OFF))
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
C FROM THE ADC NUMBER WE MUST DETERMINE THE AIM MODULE TO USE AND THE
C ASSOCIATED ADC
C   AIM #1  ADC 1-2
C   AIM #2  ADC 3-4
C   AIM #3  ADC 5-6

```

```

C
  IF(RN .GE. 1 .AND. RN .LE. 2) THEN
    IAIM=1
    IADC=RN-1
  ENDIF
  IF(RN .GE. 3 .AND. RN .LE. 4) THEN
    IAIM=2
    IADC=RN-3
  ENDIF
  IF(RN .GE. 5 .AND. RN .LE. 6) THEN
    IAIM=3
    IADC=RN-5
  ENDIF
C
C STARTING ADDRESS OF DATA FOR THIS ADC
  ISA=IADC*32768
C
  STATUS=NMC_ACQU_STATUSUPDATE(AS.AIM_NO(IAIM), IADC, 0, 0, 0,
    1 ILIVE, IREAL, ITOT, ISTAT)
  IF(.NOT. STATUS) CALL LIB$STOP(%VAL(STATUS))
C
  AS.REAL_TIME(RN)=IREAL/100      !ELAPSED REAL TIME
  AS.LIVE_TIME(RN)=ILIVE/100     !ELAPSED LIVE TIME
  IF(ISTAT .EQ. 1) THEN
    AS.ON_OFF(RN)=.TRUE.
  ELSE
    AS.ON_OFF(RN)=.FALSE.
  ENDIF
C
C ENABLE AST DELIVERY
1000 STATUS=SYSS$SETAST(%VAL(ON))
  IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
  RETURN
  END

```

B.11 SET_ERR_MSG Subroutine

```

C PROGRAM TO PUT A DISPLAY MESSAGE IN THE MESSAGE BUFFER
C AS.MESSAGE IF THERE IS NOT ONE LIKE IT ALREADY IN THE BUFFER
C
C      EWK 12/9/88
C
  SUBROUTINE SET_ERR_MSG(MESSAGE)
C WHERE MESSAGE IS THE MESSAGE TO DISPLAY
C

```

```

C      INCLUDE      'gap$src:ADC_STATUS.INC/LIST'

C      INTEGER*4    STATUS,SYS$SETAST
C      BYTE         OFF    /0/
C      BYTE         ON     /1/
C      CHARACTER*80  MESSAGE

C  C DISABLE AST DELIVERY WHILE WE ARE HERE
C      STATUS=SYS$SETAST(%VAL(OFF))
C      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

C  C CHECK TO SEE IF THIS MESSAGE IS ALREADY IN THE MESSAGE ARRAY
C      DO J=1,10
C          IF(AS.MESSAGE_AVAILABLE(J))THEN
C              IF(AS.MESSAGE(J).EQ.MESSAGE)GOTO 100
C          ENDIF
C      ENDDO

C  C MESSAGE ISNT ALREADY THERE PUT IN THERE
C      DO J=1,10
C          IF(.NOT.AS.MESSAGE_AVAILABLE(J))THEN
C              AS.MESSAGE_AVAILABLE(J)=.TRUE.
C              AS.MESSAGE(J)=MESSAGE
C              GOTO 100
C          ENDIF
C      ENDDO

100  STATUS=SYS$SETAST(%VAL(ON))
C      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C      RETURN
C      END

```

B.12 STOP_COUNTING Subroutine

```

C  MODIFIED TO USE SIMPLIFIED AIM ROUTINES
C      EWK 1/18/90
C
C  ROUTINE TO STOP THE ACCUMULATION OF DATA ON AN ADC
C  ROUTINE TO BE USED WITH THE WASTE RETRIEVAL SYSTEM
C
C      SUBROUTINE STOP_COUNTING(RN,ERR)
C
C  C WHERE I=ADC NUMBER
C      IERR=ERROR CONDITION
C
C  C Modified to count over 8192 points rather than only 4096 points (and so the

```

C starting address for the ADC was changed from 16384 to 32768). D. A. Femec,
C March 1, 1993.

```
C
    INCLUDE 'gap$src:ADC_STATUS.INC/LIST'
    INCLUDE 'NDFUNC.INC'

C
    INTEGER*4    RN,STATUS
    LOGICAL      ERR
    INTEGER*4    SYS$SETAST,SYS$CANTIM
    BYTE         OFF    /0/
    BYTE         ON     /1/

C
    ERR=.FALSE.

C
C DISABLE AST DELIVERY WHILE WE ARE HERE
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))

C
C
C FROM THE ADC NUMBER WE MUST DETERMINE THE AIM MODULE TO USE AND THE
C ASSOCIATED ADC
C     AIM #1  ADC 1-2
C     AIM #2   ADC 3-4
C     AIM #3   ADC 5-6
C
    IF(RN .GE. 1 .AND. RN .LE. 2)THEN
        IAIM=1
        IADC=RN-1
    ENDIF
    IF(RN .GE. 3 .AND. RN .LE. 4)THEN
        IAIM=2
        IADC=RN-3
    ENDIF
    IF(RN .GE. 5 .AND. RN .LE. 6)THEN
        IAIM=3
        IADC=RN-5
    ENDIF

C
C STARTING ADDRESS OF DATA FOR THIS ADC
    ISA=IADC*32768

C
    STATUS=NMC_ACQU_SETSTATE(AS.AIM_NO(IAIM),IADC,0)
    IF(.NOT.STATUS)THEN
        ERR=.TRUE.
        RETURN
    ENDIF
    AS.ON_OFF(RN)=.FALSE.
```

```

      AS.AUTO_COUNT(RN)=.FALSE.
C
C CANCEL ANY TIMED AST'S FOR THIS DETECTOR
      STATUS=SYSSCANTIM(%VAL(RN+512),)
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
C ENDABLE AST DELIVERY
      STATUS=SYSSSETAST(%VAL(ON))
      IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
      RETURN
      END

```

B.13 READ_SPECTRUM Subroutine

```

C MODIFIED FOR USE IN THE SWEPP BARREL COUNTING ENVIRONMENT
C      EWK 4/9/92
C ROUTINE TO READ A SPECTRUM OUT OF AN AIM MCA FOR THE HANFORD JOB.
C SPECTRUM WILL BE STORED IN THE DIRECTORY NAME STORED IN
C "AS.SAVE_DIR". THE SPECTRUM FILE NAME WILL BE "DDYMDDHHMMS.DGE"
C WHERE:
C      DD = DETECTOR SYSTEM -- A1,A2...
C      Y = YEAR CODE -- 0=1990,1=1991...A=2000,B=2001
C      M = MONTH CODE -- 1=JAN,2=FEB...9=SEPT,A=OCT,B=NOV,C=DEC
C      DD = DAY OF MONTH
C      HH = HOUR OF DAY
C      MM = MINUTE OF HOUR
C      S = SEQUENCE CODE WITHIN THE MINUTE -- 0,1,...9,A,B,C,...Z
C THESE DATES ARE THE DATE AND TIME WHEN DATA COLLECTION WAS STARTED. THE
C SEQUENCE NUMBER S WILL BE DETERMINED FROM THE NUMBER OF FILES
C WHICH ALL READY EXISTS FOR THAT DATE. THE NAME OF THE SPECTRUM
C FILE WILL BE STORED IN "AS.SPECTRUM_ID".
C
C      EWK 2/4/92
C
C All efficiency tables are now set to <n>110, where <n> is the detector
C number (e.g., table 3110 for detector A3), and all distances are now set
C to 45 cm. D. A. Femec, June 22, 1993.
C
      SUBROUTINE READ_SPECTRUM(DET,WEIGHT,TYPE,COMMENT,ERR)
C
C WHERE: DET = THE 2 CHARACTER NAME OF THE DETECTOR SYSTEM
C      WEIGHT= WEIGHT OF BARREL IN KG
C      TYPE=TYPE OF MATRIX S=Si,F=Fe,C=Combustable
C      COMMENT=ALPHA-NUMERIC SPECTRUM HEADER
C      ERR= IS THE ERROR INDICATOR

```

```

C
INCLUDE '($RMSDEF)'
INCLUDE 'gap$src:ADC_STATUS.INC/LIST'
INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
INCLUDE 'gap$src:CALIBRATION.INC'
INCLUDE 'NDFUNC.INC'

C
REAL*4          WEIGHT
CHARACTER*2 TYPE
CHARACTER*(*)   COMMENT
INTEGER*4       STATUS
INTEGER*4       RN,MM,DD,YY
CHARACTER*8     CURRENT_DATE
CHARACTER*512    CHAR_HEAD
LOGICAL         ERR
BYTE            OFF /0/
BYTE            ON  /1/
INTEGER*4       SYSSSETAST
REAL*4          DATA(128)
INTEGER*4       SDAT(8192)
EQUIVALENCE (CHAR_HEAD,DATA)
CHARACTER*2 DET
CHARACTER*130   NAME
CHARACTER*40    SEARCH
CHARACTER*1 SEQNO
CHARACTER*8 CTIME
INTEGER*4       LIB$FIND_FILE

C
ERR=.FALSE.

C
C DETERMINE THE NUMBER OF ADC
  READ(UNIT=DET(2:2),FMT='(I1)')RN

C
C FROM THE ADC NUMBER WE MUST DETERMINE THE AIM MODULE TO USE AND THE
C ASSOCIATED ADC
C   AIM #1   ADC 1-2
C   AIM #2   ADC 3-4
C   AIM #3   ADC 5-6
C
  IF(RN .GE. 1 .AND. RN .LE. 2) THEN
    IAIM=1
    IADC=RN-1
  ENDIF
  IF(RN .GE. 3 .AND. RN .LE. 4) THEN
    IAIM=2
    IADC=RN-3
  ENDIF

```

```

        IF(RN .GE. 5 .AND. RN .LE. 6) THEN
            IAIM=3
            IADC=RN-5
        ENDIF
C
C STARTING ADDRESS OF DATA FOR THIS ADC
    ISA=IADC*32768
C
C
C DISABLE AST DELIVERY
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C
    CALL LIB$GET_LUN(ILUN)
C
C READ STATUS
    CALL READ_STATUS(RN,ERR)
    IF(ERR)RETURN
C
C START PROCESS TO DETERMINE FILE NAME OF SPECTRUM
C CONVERT START DATE YEAR AND MONTH TO INTEGER
    READ(AS.START_DATE(RN)(1:2),FMT='(I2)')MM
    READ(AS.START_DATE(RN)(5:6),FMT='(I2)')YY
C SET YEAR CODE
    I=YY-90
    IF(I .LE. 9)THEN
        CURRENT_DATE(1:1)=CHAR(I+"60")
    ELSE
        I=I-9
        CURRENT_DATE(1:1)=CHAR(I+"100")
    ENDIF
    IF(MM .LE. 9)THEN
        CURRENT_DATE(2:2)=CHAR(MM+"60") !MONTH CODE
    ELSE
        CURRENT_DATE(2:2)=CHAR((MM-9)+"100")
    ENDIF
    CURRENT_DATE(3:4)=AS.START_DATE(RN)(3:4) !DAY
    CURRENT_DATE(5:6)=AS.START_TIME(RN)(1:2) !HOUR
    CURRENT_DATE(7:8)=AS.START_TIME(RN)(3:4) !MINUTE
C
C SEARCH THE SAVE DIRECTORY TO FIND ANY FILES WITH THIS CURRENT DATA AND
C DETECTOR NAME
    SEARCH=AS.SAVE_DIR(RN)(1:INDEX(AS.SAVE_DIR(RN),':'))//DET//
    !CURRENT_DATE/'*.DGE'

    INO=0
1000 STATUS=LIB$FIND_FILE(SEARCH,NAME,IC)

```



```

        IF (STATUS .EQ. RMS$_FNF) THEN
            SEQNO='0'
            GOTO 1200
        ENDIF
        IF (STATUS .EQ. RMS$_NMF) THEN !NO MORE FILES IN DIRECTORY
            INO=INO+1
            IF (INO .LE. 9) THEN
                SEQNO=CHAR(INO+"060")
            ELSE
                SEQNO=CHAR(INO-8+"100")
            ENDIF
            GOTO 1200
        ENDIF
        INO=INO+1
        GOTO 1000
    C
    C WE NOW KNOW WHAT THE SEQUENCE NUMBER IS AND THE SPECTRUM FILE NAME
    C CAN BE CONSTRUCTED.
    1200 CALL LIB$FIND_FILE_END(IC)
        AS.SPECTRUM_ID(RN)=AS.SAVE_DIR(RN) (1:INDEX(AS.SAVE_DIR(RN),':'))//
        1DET//CURRENT_DATE//SEQNO//'.DGE'
    C
    C READ THE PARAMETER BLOCK FOR THIS DETECTOR SLOT
        OPEN (UNIT=ILUN, NAME='SLOT$SPC: '//DET//'.DAT', TYPE='OLD',
            1 ACCESS='DIRECT', RECORDSIZE=128, READONLY, SHARED)
        READ (ILUN'2) P
        CLOSE (UNIT=ILUN)
    C
    C OPEN SPECTRUM FILE
        OPEN (UNIT=ILUN, NAME=AS.SPECTRUM_ID(RN), TYPE='NEW',
            1 ACCESS='DIRECT', RECORDSIZE=128)
    C
    C FIX UP HEADER RECORD
        CHAR_HEAD(1:60)=' '
        CHAR_HEAD(10:20)=AS.SPECTRUM_ID(RN)
        1 (INDEX(AS.SPECTRUM_ID(RN),':')+1:
        1 INDEX(AS.SPECTRUM_ID(RN),'.DGE'))
        CHAR_HEAD(36:40)='00000'
        CHAR_HEAD(61:512)=COMMENT(1:LEN(COMMENT))
    C
        WRITE (ILUN'1) CHAR_HEAD
    C
    C SET UP PARAMETER BLOCK
        P.LTIME=AS.LIVE_TIME(RN)*100 !VALUE STORED AS CENTISECONDS
        P.RTIME=AS.REAL_TIME(RN)*100 !VALUE STORED AS CENTISECONDS
        READ (UNIT=AS.START_TIME(RN), FMT='(I4.4)') P.START_TIME
        READ (UNIT=AS.START_DATE(RN), FMT='(I6.6)') P.START_DATE

```

```

P.MCA_SYSTEM=DET
P.ADC_NUM=RN
P.COLLECT_HOUR=0.0
P.COLLECT_TIME=P.START_TIME
P.COLLECT_DATE=P.START_DATE
P.SMPL_FRACTION=1.0
P.EFF_MULT=1.0
p.eff_table=(rn*1000)+110
P.DISTANCE=38
P.EFF_FILE=' '
P.VOLUME=WEIGHT*1000.0
P.VOLUME_UNITS='G'
P.SYS_VOLUME_UNITS=TYPE
P.NO_CHANNELS=8192.0
P.ZERO_OFFSET=GCB.CZERO(RN)
P.EA_CO=GCB.A(RN)
P.EB_CO=GCB.B(RN)
P.EC_CO=GCB.Q(RN)
P.WA_CO=GCB.FW1(RN)
P.WB_CO=GCB.FW2(RN)
P.PLS_ONE_KEV=GCB.LPEQ(RN)
P.PLS_TWO_KEV=GCB.HPEQ(RN)
P.CAL_PULSER_WD=GCB.CAL_PULSER_WD(RN)
P.NO_PULSER=GCB.NO_PULSER(RN)
P.PULSER_RATE=GCB.PULSER_RATE(RN)
P.PLS_CHANNELS=GCB.PLS_CHANNELS(RN)
P.PLS_ONE_START=GCB.PLS_ONE_START(RN)
P.PLS_TWO_START=GCB.PLS_TWO_START(RN)
P.PLS_ONE_PS=GCB.PLS_ONE_PS(RN)
P.PLS_TWO_PS=GCB.PLS_TWO_PS(RN)

```

C

```
WRITE(ILUN'2)P
```

C

```

C READ THE DATA OUT OF THE AIM AND WRITE IT TO DISK
STATUS=NMC_ACQU_GETMEMORY_CMP(AS.AIM_NO(IAIM),IADC,ISA,
1,1,1,8192,SDAT)
IF(.NOT.STATUS)THEN
    ERR=.TRUE.
    CLOSE(UNIT=ILUN,DISP='DELETE')
    CALL LIB$FREE_LUN(ILUN)
    STATUS=SYS$SETAST(%VAL(ON))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    RETURN
ENDIF
DO I=1,64
    K=(I-1)*128
    DO J=1,128

```

```

        DATA(J)=SDAT(K+J)
        ENDDO
        WRITE(ILUN'I+6)DATA
    ENDDO
    CLOSE(UNIT=ILUN)
    CALL LIB$FREE_LUN(ILUN)
C
C  ENABLE AST DELIVERY
    STATUS=SYS$SETAST(%VAL(ON))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
    RETURN
    END

```

B.14 START_COUNTING Subroutine

```

C  MODIFIED TO USE THE SIMPLIFIED AIM ROUTINES TO BE USED WITH THE
C  WASTE RETRIEVAL PROGRAMS
C      EWK 1/18/90
C
C  ROUTINE TO START AN ADC COUNTING ON A REMOTE
C
C  Modified to count over 8192 points rather than only 4096 points (and so the
C  starting address for the ADC was changed from 16384 to 32768).  D. A. Femec,
C  March 1, 1993.
C
    SUBROUTINE START_COUNTING(SECONDS,RN,ERR)
C
C  WHERE: TIME=COUNT TIME IN SECONDS
C      RN=ADC NUMBER 1-6
C      ERR= ERROR CONDITION
C
    INCLUDE      'NDFUNC.INC'
    INCLUDE      'gap$src:ADC_STATUS.INC/LIST'
    INTEGER*4    RN,MM,DD,YY
    CHARACTER*8  C_TIME
    LOGICAL      ERR
    INTEGER*4    SECONDS
    INTEGER*4    STATUS,SYS$SETAST
    BYTE         OFF    /0/
    BYTE         ON     /1/
C
C  DISABLE AST DELIVERY WHILE WE ARE HERE
    STATUS=SYS$SETAST(%VAL(OFF))
    IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
C

```

```

ERR=.FALSE.
CALL IDATE(MM,DD,YY)
WRITE(UNIT=AS.START_DATE(RN)(1:2),FMT='(I2.2)')MM
WRITE(UNIT=AS.START_DATE(RN)(3:4),FMT='(I2.2)')DD
WRITE(UNIT=AS.START_DATE(RN)(5:6),FMT='(I2.2)')YY
C
C FROM THE ADC NUMBER WE MUST DETERMINE THE AIM MODULE TO USE AND THE
C ASSOCIATED ADC
C   AIM #1   ADC 1-2
C   AIM #2   ADC 3-4
C   AIM #3   ADC 5-6
C
  IF(RN .GE. 1 .AND. RN .LE. 2) THEN
    IAIM=1
    IADC=RN-1
  ENDIF
  IF(RN .GE. 3 .AND. RN .LE. 4) THEN
    IAIM=2
    IADC=RN-3
  ENDIF
  IF(RN .GE. 5 .AND. RN .LE. 6) THEN
    IAIM=3
    IADC=RN-5
  ENDIF
C
C STARTING ADDRESS OF DATA FOR THIS ADC
  ISA=IADC*32768
C
C
C ERASE THE SPECTRUM
  STATUS=NMC_ACQU_ERASE(AS.AIM_NO(IAIM),ISA,32768)
  IF(.NOT.STATUS) THEN
100    ERR=.TRUE.
C
C ENABLE AST DELIVERY
  STATUS=SYS$SETAST(%VAL(ON))
  IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
  RETURN
  ENDIF
C
C SET THE PRESET REAL TIME TO SECONDS AND PRESET LIVE TIME TO 0
  STATUS=NMC_ACQU_SETUP(AS.AIM_NO(IAIM),IADC,ISA,8192,0,
1    SECONDS*100,0,0,0,1)
  IF(.NOT.STATUS) GOTO 100
C TURN ON ADC
  STATUS=NMC_ACQU_SETSTATE(AS.AIM_NO(IAIM),IADC,1)
  IF(.NOT.STATUS) GOTO 100

```

```

C SET OTHER PARAMETERS IN GLOBAL SECTION
  CALL TIME(C_TIME)
  AS.START_TIME(RN) (1:2)=C_TIME(1:2)
  AS.START_TIME(RN) (3:4)=C_TIME(4:5)
C
  AS.ON_OFF(RN)=.TRUE.
  AS.LIVE_TIME(RN)=0
  AS.REAL_TIME(RN)=0
  AS.COUNT_INTERVAL(RN)=SECONDS
  AS.START_RETRY(RN)=0
C
C CHECK OF PULSERS ENABLED IF SO SCHEDULE THE START AST
C   IF(AS.CHECK_PULSER(RN))CALL SHED_AST_START(RN)
C
C ENABLE AST DELIVERY
  STATUS=SYS$SETAST(%VAL(ON))
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
  RETURN
  END

```

B.15 ASTIME Subroutine

```

C
C ROUTINE TO SET REFRESH TIME INTERVAL FOR STATUS UPDATE
C
  SUBROUTINE ASTIME(INTERVAL)
C
  EXTERNAL ADCAST
C
  CHARACTER*1 INTERVAL
  INTEGER*4   SYS$BINTM,SYS$SETIMR,SYS$CANTIM
  INTEGER*4   BIN_DELAY(2),STATUS
  CHARACTER*9 DELAY /'0 ::02.00'/
  INTEGER*4   REGIDT      /15/
  COMMON/CANT/REGIDT
C
C RESET TIME TO DO IT AGAIN
  DELAY(6:6)=INTERVAL(1:1)
  STATUS=SYS$CANTIM(%VAL(REGIDT),)
  IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
  STATUS=SYS$BINTIM(DELAY,BIN_DELAY)
  STATUS=SYS$SETIMR(,BIN_DELAY,ADCAST,%VAL(REGIDT))
  RETURN
  END

```

B.16 DVALID Function

The listing for this function is included at the end of the listing for the GC main routine.

B.17 STOP_DISPLAY Subroutine

```
C
C SUBROUTINE TO UPASTE DISPLAYS PRIOR TO CALLING A SUBROUTINE
C
C Location of SMAN.INC changed to GAP$SRC. D. A. Femec, February 4, 1993.
C
  SUBROUTINE STOP_DISPLAY
C
  INCLUDE 'SMAN.INC'
  INCLUDE 'GAP$SRC:VTERM.INC'
C
  STATUS=SMG$UNPASTE_VIRTUAL_DISPLAY(NVDID,PBID)
  STATUS=SMG$UNPASTE_VIRTUAL_DISPLAY(CVDID,PBID)
  STATUS=SMG$DISABLE_UNSOLICITED_INPUT(PBID)
  CALL ERASE
  RETURN
  END
```

B.18 ENTER_PAR Subroutine

```
C
C SUBROUTINE TO DISPLAY AND GET ANALYSIS PARAMETERS. ENTERED VALUES
C WILL BE PLACED IN FILE "SLOT$SPC:An.ANP
C
C          EWK 10/30/89
C
C Modified to include PRINT_RESULTS. D. A. Femec, April 7, 1993.
C
  SUBROUTINE ENTER_PAR(ID)
C
C WHERE: ID = 2 CHARACTER DETECTOR ID (A1,A2..A6)
C
  INCLUDE          'ANALYSIS_PARAMETER.INC/LIST'
  CHARACTER*2 ID
```

```

CHARACTER*80      TVAL
CHARACTER*40      FNAME /'SLOT$SPC:A1.ANP'/
CHARACTER*40      TXTV(22)
CHARACTER*10      DPVA(22)

C
COMMON /TITLE/TVAL
COMMON /TEXT/TXTV
COMMON /VALUES/NOMENU, DPVA

C
TVAL='  "Return" key advances to next line. Type "Ctrl Z" to exit'

C
TXTV(1)='NUCLIDE LIBRARY:'
TXTV(2)='REQUIRED ENERGY LIBRARY:'
TXTV(3)='INTERFERENCE LIBRARY:'
txtv(4)='BACKGROUND CORRECT (YES or NO)?'
txtv(5)='PRINT RESULTS (YES or NO)?'
txtv(6)='SAVE RESULTS (YES or NO)?'
txtv(7)='SAVE DIRECTORY:'

C
C OPEN PARAMETER FILE FOR THIS DETECTOR AND DISPLAY THE INFO CONTAINED THEREIN
CALL LIB$GET_LUN(LUN)
FNAME(10:11)=ID
OPEN(UNIT=LUN, NAME=FNAME, TYPE='OLD',
1      FORM='UNFORMATTED', IOSTAT=IOS)

C
C OPEN SUCESSFUL? IF NOT TRY OPEN AS A NEW FILE
IF(IOS.NE.0)THEN
    OPEN(UNIT=LUN, NAME=FNAME, TYPE='NEW',
1      FORM='UNFORMATTED', IOSTAT=IXS)
    IF(IXS.NE.0)THEN
        TYPE *, ' UNABLE TO OPEN PARAMTER FILE'
        CALL EXIT
    ENDIF
ELSE
    REWIND(UNIT=LUN)
    READ(LUN)AP
ENDIF

C
C SET VALUES FROM PARAMETER FILE
DPVA(1)=AP.ISOTOPE_LIB(1:10)
DPVA(2)=AP.REQUIRED_LIB(1:10)
DPVA(3)=AP.INTERFERENCE_LIB(1:10)
IF(AP.BACK_CORRECT)THEN
    dpva(4)='YES'
ELSE
    dpva(4)='NO'
ENDIF

```

```

      IF (AP.PRINT_RESULTS) THEN
        dpva(5) = 'YES'
      ELSE
        dpva(5) = 'NO'
      ENDIF
      IF (AP.SAVE_RESULTS) THEN
        dpva(6) = 'YES'
      ELSE
        dpva(6) = 'NO'
      ENDIF
      dpva(7) = AP.SAVE_DIR(1:10)
      NOMENU = 7
C
C  DISPLAY CURRENT VALUES
      CALL GCMENU
C
C  STUFF VALUES BACK IN TO PARAMETER FILE
      AP.ISOTOPE_LIB = DPVA(1)
      AP.REQUIRED_LIB = DPVA(2)
      AP.INTERFERENCE_LIB = DPVA(3)
      IF (dpva(4) .EQ. 'YES') THEN
        AP.BACK_CORRECT = .TRUE.
      ELSE
        AP.BACK_CORRECT = .FALSE.
      ENDIF
      IF (dpva(5) .EQ. 'YES') THEN
        AP.PRINT_RESULTS = .TRUE.
      ELSE
        AP.PRINT_RESULTS = .FALSE.
      ENDIF
      IF (dpva(6) .EQ. 'YES') THEN
        AP.SAVE_RESULTS = .TRUE.
      ELSE
        AP.SAVE_RESULTS = .FALSE.
      ENDIF
      AP.SAVE_DIR = dpva(7)
C
C  WRITE TO FILE
      REWIND(UNIT=LUN)
      WRITE(LUN) AP
      CLOSE(UNIT=LUN)
      CALL LIB$FREE_LUN(LUN)
      RETURN
      END

```


B.19 GCMENU Subroutine

```
C SUBROUTINE TO DISPLAY ANALYSIS PARAMETER MENU FOR GC PROGRAM
C
      SUBROUTINE GCMENU
C
C VALUE OF STRING WHICH WILL BE WRITTEN TO TITLE POSITION OF DISPLAY
      CHARACTER*80 TVAL
      COMMON /TITLE/TVAL
C
C TEXT LINES TO BE WRITTEN TWO LINES APART MAXIMUM OF 10 LINES
C
      CHARACTER*40 TXTV(22)
      COMMON /TEXT/TXTV

C
C DEFAULT VALUES OF PARAMETERS FOR EACH LINE OF TEXT. MAX OF 10 VALUES
C 10 CHARACTERS/VALUE
      CHARACTER*10 DPVA(22)
      COMMON /VALUES/NOMENU,DPVA
      INTEGER*4 SYSS$WAITFR,RESULT,SLINE
      BYTE INPUT(20)
      CHARACTER*1 BIN(20)
      EQUIVALENCE (INPUT,BIN)
C
C DISPLAY MENU AND GET INPUT

C
C SUBROUTINE TO BRING A SCREEN FULL OF DATA UP ON THE SCREEN THEN ALLOW
C OPERATOR TO CHANGE VALUES FOR A SET A PARAMETERS. FORMAT OF SCREEN:
C TITLE MESSAGE--80 CHARACTERS
C TEXT MESSAGE--40 CHARACTERS DEFAULT VALUES--10 CHARACTERS
C PARAMETERS TO BE PASSED BY CALLING PROGRAM IN THREE COMMON ARRAYS
C
      SLINE=1
C
C CLEAR SCREEN
250 CALL ERASE
C
C WRITE TITLE TO POSITION (1,1)
      CALL CURP(1,1)
      TYPE 9997,TVAL
9997 FORMAT('+',A80)
9000 FORMAT('+',A40)
C
C WRITE ALL TEXT MESSAGES
```

```

        J=3
        DO I=SLINE,SLINE+10
            IF(I.GT.NOMENU)GOTO 300
            CALL CURP(1,J)
            TYPE 9000,TXTV(I)
            J=J+2
        ENDDO
C
C WRITE ALL DEFAULT VALUES IN INVERSE VIDO
300  CALL RVID
        J=3
        DO I=SLINE,SLINE+10
            IF(I.GT.NOMENU)GOTO 400
            CALL CURP(43,J)
            TYPE 9001,DPVA(I)
9001  FORMAT('+',A10)
            J=J+2
        ENDDO
C
400  CALL OFFVID
C
C TYPE HOW TO USE MESSAGE
        CALL CURP(1,1)
        TYPE *,' '
C ACCEPT INPUT FROM OPERATOR WHICH WILL CHANGE STRING VALUES IN DPVA
500  INUM=1
600  ILINE=INUM*2+1
        CALL CURP(55,ILINE)
        DO I=1,20
            INPUT(I)=' ' !RESET INPUT LINE TO BLANKS
        ENDDO
700  READ(*,9500,END=2000)N,(BIN(I),I=1,N)
9500  FORMAT(Q,20A1)
        IF(N.EQ.0)GOTO 720
        CALL CURP(43,ILINE)
        II=INUM+SLINE-1
        DO I=1,10
            DPVA(II)(I:I)=BIN(I)
        ENDDO
        CALL RVID
        TYPE 9001,DPVA(II)
        CALL OFFVID
        CALL CURP(55,ILINE)
        DO I=1,20
            INPUT(I)=' '
        ENDDO
        TYPE 9011,"33,['','0','K' !ERASE TO END OF LINE

```

```

9011  FORMAT('+',4A1,$)
      CALL CURP(55,ILINE)
      GOTO 700
720   INUM=INUM+1 !OPERATOR TYPE <CR> ONLY ADVANCE TO NEXT LINE
      IF(INUM.GT.11.OR.((INUM+SLINE.GT.NOMENU).AND.(NOMENU.GT.11))) THEN
          SLINE=SLINE+11
          IF(SLINE.GT.NOMENU) THEN
              SLINE=1
          ENDIF
          GOTO 250
      ENDIF
      GOTO 600
C
C ON READ EOF COME HERE
2000  CALL OFFVID      !TURN OFF REVERSE VIDEO
C
      RETURN
      END

```

B.20 START_DISPLAY Subroutine

```

C
C ROUTINE TO REPASTE THE DISPLAY BACK ON AND CREATE VIRTUAL KEYBOARD
C
C Location of SMAN.INC and ASTTRM.INC changed to GAP$SRC.  D. A. Femec,
C February 4, 1993.
C
      SUBROUTINE START_DISPLAY
C
      INCLUDE 'SMAN.INC'
      INCLUDE 'GAP$SRC:VTERM.INC'
      INCLUDE 'ASTTRM.INC'
C
      EXTERNAL      READTIB
C
      CALL ERASE
      STATUS=SMG$PASTE_VIRTUAL_DISPLAY(NVDID,PBID,1,1)
      IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
      STATUS=SMG$PASTE_VIRTUAL_DISPLAY(CVDID,PBID,NROWS+1,1)
      IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
      STATUS=SMG$DELETE_VIRTUAL_KEYBOARD(KEY)
      IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
      STATUS=SMG$CREATE_VIRTUAL_KEYBOARD(KEY)
      IF(.NOT.STATUS) CALL LIB$STOP(%VAL(STATUS))
      STATUS=SMG$ENABLE_UNSOLICITED_INPUT(PBID,READTIB)

```

```

IF(.NOT.STATUS)CALL LIB$STOP(%VAL(STATUS))
RETURN
END

```

B.21 TH_CALB Subroutine

```

C
C ROUTINE TO PERFORM AN ENERGY CALIBRATION USING A Th-228 CALIBRATION SOURCE.
C THE RESULTS OF THE CALIBRATION WILL BE PLACED IN CALIBRATION FILE FOR
C THAT DETECTOR SYSTEM IN SLOT$SPC.
C THE 2614 Kev LINE MUST BE BEYOND THE LAST 1/3 OF SPECTRUM
C
C SUBROUTINE TH_CALB(SPEC_ID,ERR)
C WHERE:
C SPEC_ID IS THE FILE SPECIFICATION OF THE TH-228 SPECTRUM
C ERR = LOGICAL ERROR FLAG
C
C Modified to be closer to EU_CALB: the only TYPED information has to do with
C error messages - all other output is directed to the output file TH_CALB.DAT,
C "Press return to continue" at the end of the routine removed. D. A. Femec,
C June 24, 1993.
C
C Modified to add the spectrum id (SPEC_ID) to the output file. D. A. Femec,
C November 9, 1993.
C
C INCLUDE 'GAP$SRC:VHEAD.INC'
C INCLUDE 'GAP$SRC:PKD.INC'
C INCLUDE 'GAP$SRC:CALB.INC'
C INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
C CHARACTER*(*) SPEC_ID
C LOGICAL ERR
C
C REAL*8 MX(3,3),V(3),PP(3)
C REAL*4 E(5),DE(5),CCE(5),W(5),CC(5),DC(5),WG(5)
C INTEGER*4 STATUS,LIB$DATE_TIME,LIB$SIGNAL
C EXTERNAL LIB$DATE_TIME
C CHARACTER*23 DATETIME
C CHARACTER*40 ISPEC
C CHARACTER*2 SID
C CHARACTER*20 CLBFILE //SLOT$SPC: '//
C LOGICAL UNFIT
C
C DATA E(1),E(2),E(3)/238.632,583.191,860.564/
C DATA E(4),E(5)/1620.735,2614.533/

```

```

DATA DE(1),DE(2),DE(3),DE(4),DE(5)/.009,.013,.05,0.1,.055/
DATA ISPEC //'SLOT$SPC:A1.DAT'//
C
ERR=.FALSE.
CALL ERASE !ERASE TERMINAL SCREEN
C
C FROM SPECTRUM ID DETERMINE DETECTOR SYSTEM ID
J=INDEX(SPEC_ID,'.')-11
SID=SPEC_ID(J:J+1)
C
C OPEN SPECTRUM FILE AND LOAD VALUES IN VHEAD
C
C ROUTINE TO OPEN A SPECTRUM ON LUN=1, LOAD CORRESPONDING VALUES IN VHEAD
C FROM PARAMETER BLOCK.
CALL OPEN$SP(SPEC_ID,IERR)
IF (IERR) THEN
TYPE *,' UNABLE TO OPEN SPECTRUM FILE: ',SPEC_ID
5555 CALL LIB$GET_INPUT(SID(1:1),
1 'Press "Return" key when ready to continue')
ERR=.TRUE.
RETURN
END IF
C
C RESET FGET SO HE WILL KNOW A NEW SPECTRUM HAS BEEN OPENED
RESET=FGET(-1)
C OPEN CALIBRATION FILE
CLBFILE(10:15)=SID(1:2)//'.CLB'
OPEN(UNIT=2,NAME=CLBFILE,TYPE='OLD',ACCESS='DIRECT',
1 RECORDSIZE=128)
C
READ(2'1)UC.CLB
C
C PERFORM CALIBRATION
C A LEAST SQUARES QUADRATIC FIT IS MADE
C TO 5 TH-228 LINES. RESULTS ARE WRITTEN TO CALIBRATION BLOCK
C
C
C LOAD VALUES INTO VHEAD ARRAY
VHEAD(ZW)=UC.CALB.FW1 !ZERO TERM OF WIDTH EQ.
VHEAD(LW)=UC.CALB.FW2 !LINEAR TERM OF WIDTH EQ.
VHEAD(CZ)=UC.CALB.CZERO !CHANNEL ZERO
VHEAD(AE)=UC.CALB.A !A TERM OF ENERGY EQ.
VHEAD(BE)=UC.CALB.B !LINEAR TERM OF ENERGY EQ.
VHEAD(CE)=UC.CALB.Q !QUAD TERM OF ENERGY EQ.
VHEAD(HFW)=0.0 !FLAG WORD
VHEAD(SCA)=UC.CALB.SCA !SUMMING CORRECTION COEFFICIENT A

```

```

VHEAD(SCB)=UC.CALB.SCB !SUMMING CORRECTION COEFFICIENT B
VHEAD(CM1)=UC.CALB.EM(1) !CALIBRATION ERROR MATRIX COEF.
VHEAD(CM2)=UC.CALB.EM(2)
VHEAD(CM3)=UC.CALB.EM(3)
VHEAD(CM4)=UC.CALB.EM(4)
VHEAD(CM5)=UC.CALB.EM(5)
VHEAD(CM6)=UC.CALB.EM(6)
VHEAD(DPA)=0.0 !PRINT DETAILED FIT ANALYSIS FLAG
C
INC=VHEAD(NC) !NUMBER OF CHANNELS
C
C IF PULSERED SPECTRUM LOOK ONLY AS FAR AS GAMMA CUT OFF
IF(P.NO_PULSER.EQ. 2)THEN
    IGAMMA_CUT=P.PLS_ONE_START-1
ELSE
    IGAMMA_CUT=INC-20
ENDIF
BIG=0.0
C LOOK FOR 2614 Kev LINE AS THE LARGEST PEAK
C BEYOND (NUMBER OF CHANNELS)/2+200
C
DO J=INC/2+200,IGAMMA_CUT
    Y=FGET(J)
    IF(BIG.LT.Y)THEN
        BIG=Y
        PKD(5,1,XV)=J
    ENDIF
ENDDO
C5=PKD(5,1,XV)
C1=E(1)/(E(5)/C5)
L=C1
BIG=0.0
M=L-50
N=L+50
C LOOK FOR THE 238 Kev PEAK BASED ON THE LOCATION
C OF THE 2614 LINE ASSUMING NO ZERO OFFSET.
C
DO 20 J=M,N
    Y=FGET(J)
    IF(BIG.GT.Y)GOTO 20
    BIG=Y
    PKD(1,1,XV)=J
20 CONTINUE
C1=PKD(1,1,XV)
A=(E(5)-E(1))/(C5-C1) !SLOPE OF ENERGY CALIBRATION LINE
B=E(1)-A*C1 !ZERO OF ENERGY CALIBRATION LINE
DO 40 K=2,4

```

```

      L=(E(K)-B)/A
      BIG=0.0
      N=L-7
      M=L+7
C     WE CAN NOW FIND THE REST OF THE PEAKS
C     BASED ON A LINEAR LINE BETWEEN THE 238 AND 2614 LINES.
C
      DO 30 L=N,M
      Y=FGET(L)
      IF(BIG.GT.Y)GOTO 30
      BIG=Y
      AY=FGET(L-1)
      BY=FGET(L)
      CY=FGET(L+1)
      IF((AY+BY+CY) .NE. 0.0)THEN
        PKD(K,1,XV)=FLOAT(L)+((CY-AY)/(AY+BY+CY))
      ELSE
        PKD(K,1,XV)=L
      ENDIF
30    CONTINUE
40    CONTINUE
      VHEAD(NP)=5.0
C
C FIT PEAKS AT DETERMINED LOCATIONS
C
      IF(P.NO_PULSER .NE. 2)THEN
        CALL GAUSS !VARYING WIDTH
      ELSE
        P.NO_PULSER=0
        CALL GAUSS
        P.NO_PULSER=2
      ENDIF
C
C CHECK TO SEE IF ALL PEAKS WHERE FIT CORRECTLY
C I.E. PKD(I,3,FLG)=0. IF SOME WERE NOT FIT CORRECTLY TELL USER
C WHICH ONES THEN EXIT
      UNFIT=.FALSE.
      DO 6100 I=1,5
        IF(PKD(I,3,FLG).NE.0)THEN
          ICHN=PKD(I,1,XV)
          TYPE *, ' PEAK AT CHANNEL', ICHN, ' WAS NOT FIT'
          UNFIT=.TRUE.
        ENDIF
6100  CONTINUE

```

```

      IF(UNFIT)GOTO 5555      !ERROR RETURN
C
      ITB=VHEAD(LTB)
      DO I=1,5
          CC(I)=PKD(I,1,XV)
          W(I)=PKD(I,1,WV)
      ENDDO
      CALL SQLN(CC,E,5,A,B)
      UC.CALB.CZERO=A/B
C
C      COMPUTE WEIGHTING FACTOR FOR EACH ENERGY.
C
      DO 65 I=1,5
          IF(DE(I).EQ.0.0)GOTO 64
          WG(I)=1.0/(DE(I)*DE(I)+(B*B*
1(D_ENERGY(PKD(I,1,XV),PKD(I,1,DER)))**2))
          GOTO 65
64      WG(I)=1.0
65      CONTINUE
C
C      COMPUTE CORRECTED CHANNELS
C      I.E. ZERO AND NON-LINEARITY CORRECTED
C
      DO I=1,5
          CC(I)=PKD(I,1,XV)+UC.CALB.CZERO
      ENDDO
C
C      FILL IN MATRIX AND INVERT
C
      MX(2,3)=0.0
      MX(3,3)=0.0
      MX(1,3)=0.0
      MX(1,2)=0.0
      MX(1,1)=0.0
      DO 80 I=1,5
          MX(1,1)=MX(1,1)+WG(I)
          MX(1,2)=MX(1,2)+(CC(I)*WG(I))
          MX(1,3)=MX(1,3)+(CC(I)*CC(I)*WG(I))
          MX(2,3)=MX(2,3)+((CC(I)**3)*WG(I))
80      MX(3,3)=MX(3,3)+((CC(I)**4)*WG(I))
          MX(2,2)=MX(1,3)
          V(1)=0.0
          V(2)=0.0
          V(3)=0.0
          DO 90 I=1,5
              V(1)=V(1)+(E(I)*WG(I))
              V(2)=V(2)+(E(I)*CC(I)*WG(I))

```



```

90  V(3)=V(3)+(E(I)*CC(I)*CC(I)*WG(I))
    CALL INVT(3,MX,V,PP,3)
    UC.CALB.A=PP(1)
    UC.CALB.B=PP(2)
    UC.CALB.Q=PP(3)
    EQSQ=0.0
C
C    COMPUTE QUALITY OF FIT NUMBER
C
    DO 93 I=1,5
    CCE(I)=CC(I)*UC.CALB.B+(UC.CALB.Q*(CC(I)*CC(I)))+UC.CALB.A
93  EQSQ=EQSQ+WG(I)*((E(I)-CCE(I))**2)
    EQSQ=EQSQ/2
C
C    SAVE CALIBRATION RESULTS IN CALIBRATION BLOCK.
C
    UC.CALB.EM(1)=MX(1,1)*EQSQ
    UC.CALB.EM(2)=MX(2,2)*EQSQ
    UC.CALB.EM(3)=MX(3,3)*EQSQ
    UC.CALB.EM(4)=MX(1,2)*EQSQ
    UC.CALB.EM(5)=MX(1,3)*EQSQ
    UC.CALB.EM(6)=MX(2,3)*EQSQ
C
C    DO A LEAST SQUARES OF WIDTHS TO A LINE
C
    CALL SQLN(CC,W,5,A,B)
    UC.CALB.FW1=A
    UC.CALB.FW2=B
    WRITE(2'1)UC.CLB
C PUT NEWLY COMPUTED CALIBRATION VALUES IN SPECTRUM PARAMETER BLOCK
    READ(1'2)P !GET CURRENT VALUES
    P.ZERO_OFFSET=UC.CALB.CZERO
    P.EA_CO=UC.CALB.A
    P.EB_CO=UC.CALB.B
    P.EC_CO=UC.CALB.Q
    P.WA_CO=UC.CALB.FW1
    P.WB_CO=UC.CALB.FW2
    P.CAL_PULSER_WD=UC.CALB.CAL_PULSER_WD
    P.NO_PULSER=UC.CALB.NO_PULSER
    P.PLS_CHANNELS=UC.CALB.PLS_CHANNELS
    P.PLS_ONE_START=UC.CALB.PLS_ONE_START
    P.PLS_TWO_START=UC.CALB.PLS_TWO_START
    P.PLS_ONE_PS=UC.CALB.PLS_ONE_PS
    P.PLS_TWO_PS=UC.CALB.PLS_TWO_PS
    DO I=1, 6
        P.ERR_MX(I)=UC.CALB.EM(I)
    ENDDO

```

```

        WRITE(1'2)P
C
C PRINT CALIBRATION RESULTS
C
        OPEN(UNIT=6, NAME='TH_CALB.DAT', STATUS='NEW',
             1   CARRIAGECONTROL='FORTRAN')
C GET CURRENT TIME
        STATUS=LIB$DATE_TIME(DATETIME)
        IF(.NOT.STATUS)CALL LIB$SIGNAL(%VAL(STATUS))
        WRITE(6,1010)DATETIME
1010  FORMAT('1Th-228 CALIBRATION ',A23)
        lspecid=len(spec_id)  !How long is SPEC_ID (for format 1020).
        WRITE(6,1020)SID,spec_id
1020  FORMAT(/,' DETECTOR SYSTEM: ',A2,5x,'SPECTRUM ID: ',a<lspecid>)
      96  WRITE(6,600)UC.CALB.CZERO,UC.CALB.A,UC.CALB.B,UC.CALB.Q
      600  FORMAT(/,' ZERO=',F8.4,/1H ', 'ENERGY=',F8.4,'+',F8.5,'(X)+'
            1 ,1PE12.5,'(X)**2')
            WRITE(6,650)UC.CALB.FW1,UC.CALB.FW2
      650  FORMAT(/,' WIDTH=',F6.3,'+',1PE11.4,'(X)')
            WRITE(6,675)(UC.CALB.EM(I),I=1,6)
675  FORMAT(/,' ERROR MATRIX: ',6(1PE14.6))
            WRITE(6,700)
      700  FORMAT(/,' CHANNEL',6X,'ENERGY',6X,'CAL. ENG',5X,'D-ENG',
            1 4X,'WIDTH')
            DO 100 I=1,5
            DFE=E(I)-CCE(I)
            WRITE(6,800)PKD(I,1,XV),E(I),CCE(I),DFE,W(I)
      100  CONTINUE
      800  FORMAT(F9.3,3X,F9.3,3X,F9.3,5X,F6.3,3X,F6.2)
            CLOSE(UNIT=1)
            CLOSE(UNIT=2)
            RETURN
            END

```

B.22 EU_CALB Subroutine

```

C MODIFIED TO SAVE THE LOW PULSER WIDTH IN THE CALIBRATION BLOCK AND
C THE PARAMETER BLOCK
C      EWK 1/11/91
C
C ROUTINE TO PERFORM AN ENERGY CALIBRATION USING A EU-152 CALIBRATION SOURCE.
C THE RESULTS OF THE CALIBRATION WILL BE PLACED IN CALIBRATION FILE FOR
C THAT DETECTOR SYSTEM IN SLOT$SPEC.
C THE 1408 Kev LINE MUST BE BEYOND THE LAST 1/4 OF SPECTRUM
C

```

```

C Modified to be closer to TH_CALB: added parameter block value assignment
C statements for pulser equivalents, closing and printing of the output file
C EU_CALB.DAT removed (handled in the calling routine GC), "Press return to
C continue" at the end of the routine removed. D. A. Femec, June 24, 1993.
C
C Modified to add the spectrum id (SPEC_ID) to the output file. D. A. Femec,
C November 9, 1993.
C
      SUBROUTINE EU_CALB(SPEC_ID,ERR)
C WHERE:
C   SPEC_ID IS THE FILE SPECIFICATION OF THE EU-152 SPECTRUM
C   ERR = LOGICAL ERROR FLAG
C
      INCLUDE 'GAP$SRC:VHEAD.INC'
      INCLUDE 'GAP$SRC:PKD.INC'
      INCLUDE 'GAP$SRC:CALB.INC'
      INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
      CHARACTER*(*)      SPEC_ID
      LOGICAL             ERR
C
      REAL*8 MX(3,3),V(3),PP(3)
      REAL*4 E(5),DE(5),CCE(5),W(5),CC(5),DC(5),WG(5)
      INTEGER*4 STATUS,LIB$DATE_TIME,LIB$SIGNAL
      EXTERNAL LIB$DATE_TIME
      CHARACTER*23 DATETIME
      CHARACTER*40 ISPEC
      CHARACTER*2 SID
      CHARACTER*20      CLBFILE      //'SLOT$SPC: '//
      LOGICAL UNFIT
C
      DATA E(1),E(2),E(3)/121.783,344.274,778.872/
      DATA E(4),E(5)/1112.092,1408.024/
      DATA DE(1),DE(2),DE(3),DE(4),DE(5)/.009,.012,.015,.020,.03/
      DATA ISPEC //'SLOT$SPC:A1.DAT'/
C
      ERR=.FALSE.
      CALL ERASE !ERASE TERMINAL SCREEN
C
C FROM SPECTRUM ID DETERMINE DETECTOR SYSTEM ID
      J=INDEX(SPEC_ID,'.')-11
      SID=SPEC_ID(J:J+1)
C
C OPEN SPECTRUM FILE AND LOAD VALUES IN VHEAD
C
C ROUTINE TO OPEN A SPECTRUM ON LUN=1, LOAD CORRESPONDING VALUES IN VHEAD
C FROM PARAMETER BLOCK.

```

```

        CALL OPEN$SP(SPEC_ID,IERR)
        IF (IERR) THEN
            TYPE *, ' UNABLE TO OPEN SPECTRUM FILE: ',SPEC_ID
5555      CALL LIB$GET_INPUT(SID(1:1),
            1 'Press "Return" key when ready to continue')
            ERR=.TRUE.
            RETURN
        END IF

C
C RESET FGET SO HE WILL KNOW A NEW SPECTRUM HAS BEEN OPENED
        RESET=FGET(-1)
C OPEN CALIBRATION FILE
        CLBFILE(10:15)=SID(1:2) //' .CLB'
        OPEN(UNIT=2,NAME=CLBFILE,TYPE='OLD',ACCESS='DIRECT',
            1 RECORDSIZE=128)

C
        READ(2'1)UC.CLB

C
C PERFORM CALIBRATION
C     A LEAST SQUARES QUADRATIC FIT IS MADE
C     TO 5 EU-152 LINES. RESULTS ARE WRITTEN TO CALIBRATION BLOCK
C
C
C LOAD VALUES INTO VHEAD ARRAY
        VHEAD(ZW)=UC.CALB.FW1      !ZERO TERM OF WIDTH EQ.
        VHEAD(LW)=UC.CALB.FW2      !LINEAR TERM OF WIDTH EQ.
        VHEAD(CZ)=UC.CALB.CZERO    !CHANNEL ZERO
        VHEAD(AE)=UC.CALB.A        !A TERM OF ENERGY EQ.
        VHEAD(BE)=UC.CALB.B        !LINEAR TERM OF ENERGY EQ.
        VHEAD(CE)=UC.CALB.Q        !QUAD TERM OF ENERGY EQ.
        VHEAD(HFW)=0.0             !FLAG WORD
        VHEAD(SCA)=UC.CALB.SCA     !SUMMING CORRECTION COEFFICIENT A
        VHEAD(SCB)=UC.CALB.SCB     !SUMMING CORRECTION COEFFICIENT B
        VHEAD(CM1)=UC.CALB.EM(1)    !CALIBRATION ERROR MATRIX COEF.
        VHEAD(CM2)=UC.CALB.EM(2)
        VHEAD(CM3)=UC.CALB.EM(3)
        VHEAD(CM4)=UC.CALB.EM(4)
        VHEAD(CM5)=UC.CALB.EM(5)
        VHEAD(CM6)=UC.CALB.EM(6)
        VHEAD(DPA)=0.0             !PRINT DETAILED FIT ANALYSIS FLAG

C
        INC=VHEAD(NC)              !NUMBER OF CHANNELS

C
C IF PULSERED SPECTRUM LOOK ONLY AS FAR AS GAMMA CUT OFF
        IF(P.NO_PULSER .EQ. 2)THEN
            IGAMMA_CUT=P.PLS_ONE_START-1

```

ELSE

IGAMMA_CUT=INC-20

ENDIF

BIG=0.0

LOOK FOR 1408 Kev LINE AS THE LARGEST PEAK
BEYOND (NUMBER OF CHANNELS)/2+200

DO J=VHEAD(NC)*0.75, IGAMMA_CUT
Y=FGET(J)

IF(BIG.LT.Y) THEN

BIG=Y

PKD(5,1,XV)=J

ENDIF

ENDDO

C5=PKD(5,1,XV)

C1=E(1)/(E(5)/C5)

L=C1

BIG=0.0

M=L-50

N=L+50

LOOK FOR THE 121 Kev PEAK BASED ON THE LOCATION
OF THE 1408 LINE ASSUMING NO ZERO OFFSET.

DO 20 J=M,N

Y=FGET(J)

IF(BIG.GT.Y) GOTO 20

BIG=Y

PKD(1,1,XV)=J

20 CONTINUE

C1=PKD(1,1,XV)

A=(E(5)-E(1))/(C5-C1) !SLOPE OF ENERGY CALIBRATION LINE

B=E(1)-A*C1 !ZERO OF ENERGY CALIBRATION LINE

DO 40 K=2,4

L=(E(K)-B)/A

BIG=0.0

N=L-7

M=L+7

WE CAN NOW FIND THE REST OF THE PEAKS

BASED ON A LINEAR LINE BETWEEN THE 121 AND 1408 LINES.

DO 30 L=N,M

Y=FGET(L)

IF(BIG.GT.Y) GOTO 30

BIG=Y

AY=FGET(L-1)

BY=FGET(L)

CY=FGET(L+1)

```

      IF((AY+BY+CY) .NE. 0.0) THEN
        PKD(K,1,XV)=FLOAT(L)+((CY-AY)/(AY+BY+CY))
      ELSE
        PKD(K,1,XV)=L
      ENDIF
30    CONTINUE
40    CONTINUE
      VHEAD(NP)=5.0
C
C FIT PEAKS AT DETERMINED LOCATIONS
C
      IF(P.NO_PULSER .NE. 2) THEN
        CALL GAUSS !VARYING WIDTH
      ELSE
        P.NO_PULSER=0
        CALL GAUSS
        P.NO_PULSER=2
      ENDIF
C
C CHECK TO SEE IF ALL PEAKS WERE FIT CORRECTLY
C I.E. PKD(I,3,FLG)=0. IF SOME WERE NOT FIT CORRECTLY TELL USER
C WHICH ONES THEN EXIT
      UNFIT=.FALSE.
      DO 6100 I=1,5
        IF(PKD(I,3,FLG).NE.0) THEN
          ICHN=PKD(I,1,XV)
          TYPE *, ' PEAK AT CHANNEL', ICHN, ' WAS NOT FIT'
          UNFIT=.TRUE.
        ENDIF
6100  CONTINUE
      IF(UNFIT)GOTO 5555      !ERROR RETURN
C
      ITB=VHEAD(LTB)
      DO I=1,5
        CC(I)=PKD(I,1,XV)
        W(I)=PKD(I,1,WV)
      ENDDO
      CALL SQLN(CC,E,5,A,B)
      UC.CALB.CZERO=A/B
C
C COMPUTE WEIGHTING FACTOR FOR EACH ENERGY.
C
      DO 65 I=1,5
        IF(DE(I).EQ.0.0)GOTO 64

```

```

      WG(I)=1.0/(DE(I)*DE(I)+(B*B*
      1(D ENERGY(PKD(I,1,XV),PKD(I,1,DER)))**2))
      GOTO 65
      WG(I)=1.0
      CONTINUE
      COMPUTE CORRECTED CHANNELS
      I.E. ZERO AND NON-LINEARITY CORRECTED
      DO I=1,5
      CC(I)=PKD(I,1,XV)+UC.CALB.CZERO
      ENDDO
      FILL IN MATRIX AND INVERT
      MX(2,3)=0.0
      MX(3,3)=0.0
      MX(1,3)=0.0
      MX(1,2)=0.0
      MX(1,1)=0.0
      DO 80 I=1,5
      MX(1,1)=MX(1,1)+WG(I)
      MX(1,2)=MX(1,2)+(CC(I)*WG(I))
      MX(1,3)=MX(1,3)+(CC(I)*CC(I)*WG(I))
      MX(2,3)=MX(2,3)+(CC(I)**3)*WG(I)
      MX(3,3)=MX(3,3)+(CC(I)**4)*WG(I)
      MX(2,2)=MX(1,3)
      V(1)=0.0
      V(2)=0.0
      V(3)=0.0
      DO 90 I=1,5
      V(1)=V(1)+(E(I)*WG(I))
      V(2)=V(2)+(E(I)*CC(I)*WG(I))
      V(3)=V(3)+(E(I)*CC(I)*CC(I)*WG(I))
      CALL INVT(3,MX,V,PP,3)
      UC.CALB.A=PP(1)
      UC.CALB.B=PP(2)
      UC.CALB.Q=PP(3)
      EQSQ=0.0
      COMPUTE QUALITY OF FIT NUMBER
      DO 93 I=1,5
      CCE(I)=CC(I)*UC.CALB.B+(UC.CALB.Q*(CC(I)
      93 EQSQ=EQSQ+WG(I)*((E(I)-CCE(I))**2)
      EQSQ=EQSQ/2

```

```

C      SAVE CALIBRATION RESULTS IN CALIBRATION BLOCK.
C
      UC.CALB.EM(1)=MX(1,1)*EQSQ
      UC.CALB.EM(2)=MX(2,2)*EQSQ
      UC.CALB.EM(3)=MX(3,3)*EQSQ
      UC.CALB.EM(4)=MX(1,2)*EQSQ
      UC.CALB.EM(5)=MX(1,3)*EQSQ
      UC.CALB.EM(6)=MX(2,3)*EQSQ
C
C      DO A LEAST SQUARES OF WIDTHS TO A LINE
C
      CALL SQLN(CC,W,5,A,B)
      UC.CALB.FW1=A
      UC.CALB.FW2=B
      WRITE(2'1)UC.CLB
C PUT NEWLY COMPUTED CALIBRATION VALUES IN SPECTRUM PARAMETER BLOCK
      READ(1'2)P !GET CURRENT VALUES
      P.ZERO_OFFSET=UC.CALB.CZERO
      P.EA_CO=UC.CALB.A
      P.EB_CO=UC.CALB.B
      P.EC_CO=UC.CALB.Q
      P.WA_CO=UC.CALB.FW1
      P.WB_CO=UC.CALB.FW2
      P.CAL_PULSER_WD=UC.CALB.CAL_PULSER_WD
      P.NO_PULSER=UC.CALB.NO_PULSER
      P.PLS_CHANNELS=UC.CALB.PLS_CHANNELS
      P.PLS_ONE_START=UC.CALB.PLS_ONE_START
      P.PLS_TWO_START=UC.CALB.PLS_TWO_START
      P.PLS_ONE_PS=UC.CALB.PLS_ONE_PS
      P.PLS_TWO_PS=UC.CALB.PLS_TWO_PS
      DO I=1, 6
        P.ERR_MX(I)=UC.CALB.EM(I)
      ENDDO
      WRITE(1'2)P
C
C PRINT CALIBRATION RESULTS
C
      OPEN(UNIT=6,NAME='EU_CALB.DAT',STATUS='NEW',
1        CARRIAGECONTROL='FORTRAN')
C GET CURRENT TIME
      STATUS=LIB$DATE_TIME(DATETIME)
      IF(.NOT.STATUS)CALL LIB$SIGNAL(%VAL(STATUS))
      WRITE(6,1010)DATETIME
1010 FORMAT('1EU-152 CALIBRATION ',A23)
      lspecid=len(spec_id) !How long is SPEC_ID (for format 1020).
      WRITE(6,1020)SID,spec_id
1020 FORMAT(/,' DETECTOR SYSTEM: ',A2,5x,'SPECTRUM ID: ',a<lspecid>)

```



```

96  WRITE(6,600)UC.CALB.CZERO,UC.CALB.A,UC.CALB.B,UC.CALB.Q
600  FORMAT('/', ' ZERO=', F8.4, /1H , 'ENERGY=', F8.4, '+', F8.5, ' (X) + '
1   , 1PE12.5, ' (X)**2')
    WRITE(6,650)UC.CALB.FW1,UC.CALB.FW2
650  FORMAT('/', ' WIDTH=', F6.3, '+', 1PE11.4, ' (X) ')
    WRITE(6,675)(UC.CALB.EM(I), I=1, 6)
675  FORMAT('/', ' ERROR MATRIX: ', 6(1PE14.6))
    WRITE(6,700)
700  FORMAT('/', ' CHANNEL', 6X, 'ENERGY', 6X, 'CAL. ENG', 5X, 'D-ENG',
1   4X, 'WIDTH')
    DO 100 I=1, 5
    DFE=E(I)-CCE(I)
    WRITE(6,800)PKD(I, 1, XV), E(I), CCE(I), DFE, W(I)
100  CONTINUE
800  FORMAT(F9.3, 3X, F9.3, 3X, F9.3, 5X, F6.3, 3X, F6.2)
    CLOSE(UNIT=1)
    CLOSE(UNIT=2)
    RETURN
    END

```

B.23 PEQ_TH Subroutine

```

C SUBROUTINE TO COMPUTE THE PULSER EQUIVALENT ENERGIES BASED ON A TH-228
C CALIBRATION SPECTRUM
C "PLS_ONE_KEV" AND "PLS_TWO_KEV" IN THE PARAMETER BLOCK FOR THE SPECTRUM
C SLOT IS UP DATED WITH THE NEW PULSER EQUIVALENT ENERGIES.
C
C          EWK 2/4/92
C
C SUBROUTINE PEQ_TH(SPEC_ID,ERR)
C
C WHERE
C SPEC_ID = THE FILE SPECIFICATION OF THE TH-228 SPECTRUM
C ERR = THE ERROR FLAG LOGICAL
C
C INCLUDE 'GAP$SRC:VHEAD.INC'
C INCLUDE 'GAP$SRC:PKD.INC'
C INCLUDE 'GAP$SRC:CALB.INC'
C INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
C CHARACTER*(*) SPEC_ID
C LOGICAL ERR
C
C REAL*4 PL(512), PH(512), DATA(128), PLUS(512), PA(128), PB(128),
1 PC(128), PD(128)

```

```

      REAL*8 MX(3,3),V(3),PP(3)
      EQUIVALENCE (DATA,PLUS),(PA,PLUS),(PB,PLUS(129)),(PC,PLUS(257)),
1                (PD,PLUS(385))
      CHARACTER*2 SID
      CHARACTER*20      CLBFILE      //'SLOT$SPC: '//
      ERR=.FALSE.
C ENERGY CALIBRATE THE SPECTRUM
      CALL TH_CALB(SPEC_ID,ERR)
      IF(ERR)RETURN
      J=INDEX(SPEC_ID,'.')-11
      SID=SPEC_ID(J:J+1)
      CLBFILE(10:15)=SID(1:2)//'.CLB'
      OPEN(UNIT=2,NAME=CLBFILE,TYPE='OLD',ACCESS='DIRECT',
1 RECORDSIZE=128)
      READ(2'1)UC.CLB
      CALL OPEN$SP(SPEC_ID,IERR)
      VHEAD(ZW)=UC.CALB.FW1
      VHEAD(LW)=UC.CALB.FW2
      RESET=FGET(-1)
      INC=VHEAD(NC)
      IGAMMA_CUT=P.PLS_ONE_START-1
C
C ENERGY SCALE DETERMINATION NOW COMPLETE PROCEED WITH DETERMINING
C THE PULSER EQUIVALENTS. TO DO THAT WE MUST CONSTRUCT A PULSER PEAK ONLY
C SPECTRUM. THIS SPECTRUM MUST BE OPEN ON LUN=1 SO THAT THE PEAKS CAN BE
C FIT.
C
C RETRIVE PULSER DATA
C
      PLC=P.PLS_ONE_START-P.PLS_ONE_PS      !LOW PULSER OFF-SET
      PHC=P.PLS_TWO_START-P.PLS_TWO_PS      !UPPER PULSER OFF-SET
C CONTINUE THE PROCESS WITH THE PULSER INFO WHICH IS IN THE SPECTRUM
      J=P.PLS_ONE_START
      K=P.PLS_TWO_START
      DO I=1,P.PLS_CHANNELS
          PL(I)=FGET(J)
          PH(I)=FGET(K)
          J=J+1
          K=K+1
      ENDDO
C OPEN TEMPORY SPECTRUM FILE IN WHICH TO PUT THE PULSER SPECTRUM
      CLOSE(UNIT=1)
      OPEN(UNIT=1,NAME='SLOT$SPC:PEQ.DAT',TYPE='NEW',
1 ACCESS='DIRECT',RECORDSIZE=128)
      DO I=1,128
          DATA(I)=0.0
      ENDDO

```

```

DO I=7,(INC/128)
  WRITE(1'I)DATA
ENDDO
IL=INT(PLC/128.0)
ILR=INT((PLC-(IL*128))+1)
IH=INT(PHC/128.0)
IHR=INT((PHC-(IH*128))+1)
IL=IL+7
IH=IH+7
DO I=1,P.PLS_CHANNELS
  DATA(ILR)=PL(I)
  ILR=ILR+1
  IF(ILR.LE.128)GOTO 500
  WRITE(1'IL)DATA
  IL=IL+1
  ILR=1
  DO J=1,128
    DATA(J)=0.0
  ENDDO
500 ENDDO
  WRITE(1'IL)DATA
  DO I=1,128
    DATA(I)=0.0
  ENDDO
  DO I=1,P.PLS_CHANNELS
    DATA(IHR)=PH(I)
    IHR=IHR+1
    IF(IHR.LE.128)GOTO 540
    WRITE(1'IH)DATA
    IH=IH+1
    IHR=1
    DO J=1,128
      DATA(J)=0.0
    ENDDO
540 ENDDO
  WRITE(1'IH)DATA
  RESET=FGET(-1)
C FIND LOCATION OF PULSER PEAKS
C
  PKD(1,1,XV)=0.0
  PKD(2,1,XV)=0.0
  YMAX=0.0
  DO I=100,INC/2
    Y=FGET(I)
    IF(Y.GE.YMAX)THEN
      PKD(1,1,XV)=I
      YMAX=Y
    
```

```

        ENDIF
    ENDDO
    YMAX=0.0
    DO I=(INC/2)+1,IGAMMA_CUT
        Y=FGET(I)
        IF(Y.GE.YMAX)THEN
            PKD(2,1,XV)=I
            YMAX=Y
        ENDIF
    ENDDO
    VHEAD(NP)=2.0
    VHEAD(LW)=0.0      !LINEAR TERM OF WIDTH EQ.=0.0 SINCE PULSER
                        !PEAKS ARE THE SAME WIDTH EVERY WHERE
    P.NO_PULSER=0

C
C FIT THE PEAKS TWICE TO REFINE THE POSITIONS AND GET A BETTER FIT
    CALL GAUSS
    CALL GAUSS !VARYING WIDTH
    P.NO_PULSER=2

C
C CHECK RESULTS OF FIT TO DETERMINE IF EVERY THING OK
    DO I=1,2
        IF(PKD(I,3,FLG).NE.0.0)THEN
            TYPE*, ' PULSER AT ',PKD(I,1,XV),
1            ' WAS NOT FIT'
5555    CALL LIB$GET_INPUT(SID(1:1),
1    'Press "Return" key when ready to continue')
            CLOSE(UNIT=1,DISP='DELETE')
            CLOSE(UNIT=2)
            ERR=.TRUE.
            RETURN
        ENDIF
        IF(PKD(I,1,WV).GE.5.0)THEN
            TYPE *, ' PULSER AT ',PKD(I,1,XV), ' IS TOO WIDE'
            TYPE *, ' WITH A WIDTH OF: ',PKD(I,1,WV)
            GOTO 5555
        ENDIF
    ENDDO
    IF(ABS(PKD(1,1,AV)-PKD(2,1,AV)).GT.(0.1*PKD(1,1,AV)))THEN
        TYPE *, 'PULSER AREAS DO NOT AGREE'
        TYPE *, ' LOWER PULSER AREA: ',PKD(1,1,AV)
        TYPE *, ' UPPER PULSER AREA: ',PKD(2,1,AV)
        GOTO 5555
    ENDIF

C
    WRITE(6,9010)PKD(1,1,WV),PKD(1,3,ERA),PKD(2,3,ERA)
9010  FORMAT(/,4X,'LOW PULSER WIDTH: ',F6.2,/,

```

```

      1 4X, 'LOW PULSER ENERGY EQUIVALENT: ', F9.3, /,
      2 4X, 'HIGH PULSER ENERGY EQUIVALENT: ', F9.3)
C
C UPDATE CALIBRATION BLOCK
      UC.CALB.LPEQ=PKD(1,3,ERA)
      UC.CALB.HPEQ=PKD(2,3,ERA)
      UC.CALB.CAL_PULSER_WD=PKD(1,1,WV)
      WRITE(2'1)UC
      CLOSE(UNIT=1,DISP='DELETE')
      CLOSE(UNIT=2)
C
C UPDATE PARAMETER BLOCK WITH PULSER EQUIVALENTS
      CALL OPEN$SP(SPEC_ID,IER)
      READ(1'2)P
      P.PLS_ONE_KEV=PKD(1,3,ERA)
      P.PLS_TWO_KEV=PKD(2,3,ERA)
      P.CAL_PULSER_WD=PKD(1,1,WV)
      WRITE(1'2)P
      CLOSE(UNIT=1)
      RETURN
      END

```

B.24 QUALITY_UPDATE Subroutine

```

C SUBROUTINE TO MAKE AN ENTRY IN THE QUALITY CONTROL DATA FILE.
C THIS ROUTINE UPDATES THE QUALITY CONTROL FILE WITH THE CALIBRATION DATA
C THE DATA IN THE FILE CAN BE PRINTED WITH THE QC_REPORT PROGRAM.
C SPOOL FILE SHOULD BE OPEN ON UNIT 6. THIS ROUTINE CAN HANDLE AT MOST
C 5 NUCLIDES WITH 10 LINES/NUCLIDE. BEFORE ROUTINE IS CALLED THE CALIBRATION
C ENTRIES, CALIBRATION NUCLIDE INFORMATION MUST BE IN THE STRUCTURE
C "QC_QUALITY". THIS ROUTINE WILL RE-ANALYZE THE CALIBRATION SPECTRUM AND
C ENTER THE LINE ENTRY VALUES FOR EACH NUCLIDE IN THE CALIBRATION SPECTRUM
C TEST FOR OUTLIERS AND SAVE THE RESULTS IN THE QUALITY CONTROL FILE.
C      EWK 2/2/94
      SUBROUTINE QUALITY_UPDATE(SPID,LIBID,EFF)
C WHERE: SPID = FULL FILE SPECIFICATION OF CALIBRATION SPECTRUM
C      LIBID= FULL FILE SPECIFICATION OF THE NUCLIDE LIBRARY
C      EFF= EFFICIENCY TABLE TO USE
C
C Expanded the name of the quality data file Q_FILE from X1QC.DAT to X1AAQC.DAT
C to include QNUCID as AA, QNUCID serving as a quality file identifier and here
C it is taken to be the first two letters of the nuclide library file name (see
C also QA_REPORT_SUB.FOR). D. A. Femec, February 22, 1994.
C
C Corrected vector size assignment for AZ (3 to 30) and added a blank line

```

C before the net activities. D. A. Femec, June 2, 1994.
C

```
INCLUDE 'QC_QUALITY.INC'  
INCLUDE 'GAP$SRC:VHEAD.INC'  
INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'  
INCLUDE 'GAP$SRC:PKD.INC'
```

```
CHARACTER*(*) SPID  
CHARACTER*(*) LIBID  
INTEGER*4 EFF
```

```
CHARACTER*1 ANS  
CHARACTER*40 Q_FILE /'SLOT$SPC:X1AAQC.DAT'/  
REAL*4 AZ(30),AEB(30),AWA(30),AWB(30),ANET(5,30),AVNET(5),SDNET(5)  
REAL*4 NETVAL(30)
```

```
REAL*4 QA  
LOGICAL QA_FLAG /.FALSE./
```

```
REAL*4 FNID  
CHARACTER*4 CNID  
EQUIVALENCE (FNID,CNID)
```

C THE FOLLOWING STRUCTURE IS DATA EQUIVALENT TO THE STRUCTURE IN "QC_QUALITY.INC"
C BUT WE WILL USE THIS STRUCTURE TO TEMPORARLY HOLD RECORDS READ FROM THE
C QUALITY DATA FILE.

```
STRUCTURE /QC_QUALITY_TEMP/  
CHARACTER*6 DATE !MMDDYY RECORD CREATED  
CHARACTER*4 TIME !HHMM RECORD CREATED  
CHARACTER*11 SPECTRUM_ID !CALIBRATION SPECTRUM ID  
REAL*4 ZERO !FOLLOWING ARE CALIBRATION CONSTANTS  
REAL*4 A_TERM  
REAL*4 B_TERM  
REAL*4 C_TERM  
REAL*4 W_ZERO  
REAL*4 W_LIN  
REAL*4 LOW_PULSER_EQ  
REAL*4 HIGH_PULSER_EQ  
LOGICAL PULSER !PULSERED SPECTRUM  
INTEGER*4 NONE !NUMBER OF DIFFERENT NUCLIDES IN CALIBA.  
INTEGER*4 NOLINE(5) !NUMBER OF LINES FOR EACH NUCLIDE  
REAL*4 NET_MC(5) !NET MC VALUE FOR EACH NUCLIDE  
REAL*4 NET_ERR(5) !NET ERR IN MC VALUE FOR EACH NULCIDE  
INTEGER*4 NA(5) !NUCLIDE "A"  
INTEGER*4 NZ(5) !NUCLIDE "Z"
```

```

        CHARACTER*4 NID(5)          !NUCLIDE ID

C ENTRIES FOR EACH DIFFERENT NUCLIDE
        REAL*4          CENTROID(5,10) !CENTROID 10 LINES MAX
        REAL*4          WIDTH(5,10)
        REAL*4          LINE_ENERGY(5,10)
        REAL*4          LINE_MC(5,10)
        REAL*4          LINE_ERR(5,10)
        REAL*4          AREA(5,10)
        INTEGER*4       FLAG(5,10)

        END STRUCTURE
        RECORD /QC_QUALITY_TEMP/TQC

C ARRAYS USED TO STORE THE VALUES FOR THE LAST SIX QALITY FILE ENTRIES
C
        REAL*4          QENG(6,5,5),QWIDTH(6,5,5),QMC(6,5,5),QM CER(6,5,5),QZERO(6),
        1 QGAIN(6),QLEQ(6),QHEQ(6),QNET(6,5),QNETER(6,5)
        INTEGER*4       QFLG(6,5,5)
        CHARACTER*6     QDATE(6)
        CHARACTER*4     QTIME(6)

C
C
        CALL OPEN$SP(SPID,IERR)
        P.EFF_TABLE=EFF

C
C BEGIN ANALYSIS
C OPEN ISOTOPE AND LIMIT LIBRARIES
        OPEN(UNIT=2,NAME=LIBID,READONLY,
        1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128)
        OPEN(UNIT=3,NAME='GE$LIMIT$LIB:ZEROLIM.LIB',READONLY,
        1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128)

C
        VHEAD(SSN)=10.0
        VHEAD(NP)=0.0
        VHEAD(HFW)=0.0
        VHEAD(ETB)=1101
        CALL BATCH_CAL_PULSER(SPID)      !PULSER CALIBRATE WITH NEW PULSER EQUIVALENTS
        CALL SEARCH
        CALL GAUSS
        CALL D_FIT
        CLOSE(UNIT=3)
        CALL LIBRARY_TAG
        CALL CHECK_ICODE(NRM)
        CALL RSOLVE

C
C EXTRACT THE FIT RESULTS FROM THE PKD ARRAY AND LOAD THEM INTO THE

```

```

C QC.QUALITY STRUCTURE WHICH IS PARTIALLY FILLED OUT BY THE CALLING ROUTINE
C
C NOTE: THE LIBRARY USED FOR THE ABOVE ANALYSIS CAN NOT HAVE MULTIPLE
C TAGS FOR ANY LINE AND MUST CONTAIN ONLY THE NUCLIDES USED FOR CALIBRATION

```

```

      NPS=VHEAD(NP)
      DO J=1,QC.NONE
        NOL=0
        DO I=1,NPS
          IF(PKD(I,2,NOT) .EQ. 0.0)GOTO 500
          IA=PKD(I,1,IAA)/10.0
          IZ=PKD(I,1,IZA)
          FNID=PKD(I,1,ISA)
          IF((IA .EQ. QC.NA(J)) .AND.
1           (IZ .EQ. QC.NZ(J)) .AND.
2           (CNID .EQ. QC.NID(J)))THEN

            NOL=NOL+1
            QC.CENTROID(J,NOL)=PKD(I,1,XV)
            QC.WIDTH(J,NOL)=PKD(I,1,WV)
            QC.LINE_ENERGY(J,NOL)=PKD(I,3,ERA)
            QC.LINE_MC(J,NOL)=PKD(I,1,MCI)
            QC.LINE_ERR(J,NOL)=PKD(I,1,EMCI)
            QC.AREA(J,NOL)=PKD(I,1,AV)
            QC.FLAG(J,NOL)=PKD(I,3,FLG)
            QC.NET_MC(J)=PKD(I,1,NET)
            QC.NET_ERR(J)=PKD(I,1,NETE)
            QC.NOLINE(J)=NOL
          ENDIF
500      ENDDO
          IF(NOL .EQ. 0)THEN
            TYPE 9100,QC.NID(J),QC.NA(J)
9100    FORMAT(//,2X,'UNABLE TO LOCATE ENTERY FOR ',A4,I3)
            CALL EXIT
          ENDIF
        ENDDO
C
C PRINT THE RESULTS
      WRITE(6,9600)
9600    FORMAT(///,38X,
1      'ANALYSIS OF CALIBRATION LINES USING NEW PULSER EQUIVALENTS',
2      '//,3X,'CENTROID',3X,'WIDTH',3X,'ENERGY',
3      4X,'AREA',6X,'FLAGS',6X,'MICROCURRIES',5X,'%ERROR',
4      5X,'NET MICROCURRIES',3X,'%ERROR',
5      4/,3X,8(' '),3X,5(' '),2X,7(' '),2X,9(' '),3X,5(' '),
6      5 6X,12(' '),5X,6(' '),5X,16(' '),3X,6(' '))

```



```

      NPS=VHEAD(NP)
C
      DO J=1,QC.NONE
        DO I=1,QC.NOLINE(J)
          WRITE(6,9610)QC.CENTROID(J,I),QC.WIDTH(J,I),
1             QC.LINE_ENERGY(J,I),QC.AREA(J,I),
2             QC.FLAG(J,I),QC.LINE_MC(J,I),QC.LINE_ERR(J,I)
9610  FORMAT(1X,F10.2,F7.1,F10.2,F10.0,I8,7X,1PE10.3,5X,0PF7.1)
          ENDDO

          WRITE(6,9620)QC.NID(J),QC.NA(J),QC.NET_MC(J),QC.NET_ERR(J)
9620  FORMAT(81X,A4,I3,1PE10.3,2X,0PF7.1)
          ENDDO

C  UPDATE THE QUALITY FILE
      CLOSE(UNIT=1)
C
C  Find the quality file identifier, first assuming that LIBID contains a
C  logical for the directory specification; if that fails, assume that a full
C  directory specification is given; and if that fails, assume that no
C  directory is specified.
C
      j=index(libid,':')  !First look for a logical.
      if(j.eq.0)j=index(libid,']')  !Then look for a full directory specification.
      Q_FILE(10:13)=QC.SPECTRUM_ID(1:2)//libid((j+1):(j+2))
C
      OPEN(UNIT=1,NAME=Q_FILE,FORM='UNFORMATTED',TYPE='UNKNOWN')
C
C  BACKSPACE 6 RECORDS IF THEY EXIST AND PRINT THOSE VALUES
      REWIND(UNIT=1)
      IREC=0
      IX=0
1000  READ(1,END=2000)TQC
      IREC=IREC+1
      IX=IX+1
      IF(IX.GT. 30)IX=1
      AZ(IX)=TQC.ZERO
      AEB(IX)=TQC.B_TERM
      AWA(IX)=TQC.W_ZERO
      AWB(IX)=TQC.W_LIN
      DO I=1,TQC.NONE
        ANET(I,IX)=TQC.NET_MC(I)
      ENDDO
      GOTO 1000

C  DONE READING THE LAST 30 OR LESS ENTRIES IN THE QUALITY FILE

```

C CHECK CURRENT VALUE TO SEE IF IT IS IN STATISTICAL AGREEMENT

C FIRST COMPUTE AVERAGE OF ALL VALUES

2000 IF(IX .GT. 2)THEN

AVZ=0.0

AVEA=0.0

AVEB=0.0

AVWA=0.0

AVWB=0.0

DO I=1,TQC.NONE

AVNET(I)=0.0

ENDDO

DO I=1,IX

AVZ=AVZ+AZ(I)

AVEB=AVEB+AEB(I)

AVWA=AVWA+AWA(I)

AVWB=AVWB+AWB(I)

DO J=1,TQC.NONE

AVNET(J)=AVNET(J) + ANET(J,I)

ENDDO

ENDDO

AVZ=AVZ/IX

AVEB=AVEB/IX

AVWA=AVWA/IX

AVWB=AVWB/IX

DO J=1,TQC.NONE

AVNET(J)=AVNET(J)/IX

ENDDO

C

C COMPUTE STANDARD DEVIATION

CALL STD_DEV(IX,AZ,SDZ)

CALL STD_DEV(IX,AEB,SDEB)

CALL STD_DEV(IX,AWA,SDWA)

DO I=1,TQC.NONE

DO J=1,IX

NETVAL(J)=ANET(I,J)

ENDDO

CALL STD_DEV(IX,NETVAL,SDNET(I))

ENDDO

C

TEMP=QA(AVZ,QC.ZERO,SDZ)

IF(TEMP .GT. 3.0)THEN

TYPE *, ' '

TYPE *, ' ZERO OF ENERGY EQUATION IS NOT IN AGREEMENT'

TYPE *, ' AVERAGE VALUE=',AVZ,' STD DEV=',SDZ

TYPE *, ' CURENT VALUE=',QC.ZERO,

```

1      ' QUALITY FLAG VALUE=',TEMP
      QA_FLAG=.TRUE.
ENDIF
TEMP=QA(AVEB, QC.B_TERM, SDEB)
IF(TEMP .GT. 3.0) THEN
  TYPE *, ' '
  TYPE *, ' "B" OF ENERGY EQUATION IS NOT IN AGREEMENT'
  TYPE *, ' AVERAGE VALUE=', AVEB, ' STD DEV=', SDEB
  TYPE *, ' CURENT VALUE=', QC.B_TERM,
1      ' QUALITY FLAG VALUE=',TEMP
      QA_FLAG=.TRUE.
ENDIF
TEMP=QA(AVWA, QC.W_ZERO, SDWA)
IF(TEMP .GT. 3.0) THEN
  TYPE *, ' '
  TYPE *, ' "A" OF WIDTH EQUATION IS NOT IN AGREEMENT'
  TYPE *, ' AVERAGE VALUE=', AVWA, ' STD DEV=', SDWA
  TYPE *, ' CURENT VALUE=', QC.W_ZERO,
1      ' QUALITY FLAG VALUE=',TEMP
      QA_FLAG=.TRUE.
ENDIF

DO I=1, TQC.NONE
  TEMP=QA(AVNET(I), QC.NET_MC(I), SDNET(I))
  IF(TEMP .GT. 3.0) THEN
    TYPE 9500, QC.NID(I), QC.NA(I), AVNET(I), SDNET(I),
1      QC.NET_MC(I), TEMP
9500  FORMAT(/, 2X, 'NET MICRO-CURIE FOR ', A4, I3, ' IS NOT IN AGREEMENT' , /,
1 4X, 'AVERAGE VALUE= ', 1PE10.7, ' STD DEV = ', 1PE10.7, /,
2 4X, 'CURRENT VALUE= ', 1PE10.7, /,
3 4X, 'QUALITY FLAG VALUE= ', F8.1)
      QA_FLAG=.TRUE.
  ENDIF
ENDDO

C
C IF ANY VALUE IS NOT IN AGREEMENT TELL OPERATOR
  IF(QA_FLAG) THEN
    TYPE *, ' '
    TYPE *, ' '
    TYPE *, ' ONE OR MORE VALUES FROM THIS CALIBRATION.'
    TYPE *, ' IS NOT IN STATISTICAL AGREEMENT'
    CALL LIB$GET_INPUT(ANS,
1  'DO YOU WANT TO ADD THIS RESULT TO THE QUALITY FILE (N=NO)? ')
    IF(ANS(1:1) .EQ. 'N') CALL EXIT
  ENDIF
ENDIF

```

C

```

C UPDATE QUALITY DATA FILE
  WRITE(1)QC
  IREC=IREC+1
C
  IF(IREC .GT. 6)THEN
    IREC=IREC-6
    REWIND(UNIT=1)
    DO J=1,IREC
      READ(1)TQC
    ENDDO
    NOR=6
  ELSE
    REWIND(UNIT=1)
    NOR=IREC
  ENDIF
C
5000 WRITE(6,9510)
9510 FORMAT('1',///,T50,'CALIBRATION NUCLIDE DATA',///,
  1 1X,'NUCLIDE',T12,'DATE',T19,'TIME',T31,'ENERGY',T40,'WIDTH',
  2 T49,'uCURIES',T61,'% ERROR',T72,'FLAG',T79,'NET uCURIES',
  3 T93,'% ERROR',/,1X,100(' '))
C
C LOAD THE ARRAYS WITH VALUES FROM THE LAST 6 ENTRIES
C
  DO I=1,NOR
    READ(1)TQC
    QDATE(I)=TQC.DATE
    QTIME(I)=TQC.TIME
    QZERO(I)=TQC.ZERO
    QGAIN(I)=TQC.B_TERM
    QLEQ(I)=TQC.LOW_PULSER_EQ
    QHEQ(I)=TQC.HIGH_PULSER_EQ

    DO J=1,TQC.NONE
      QNET(I,J)=TQC.NET_MC(J)
      QNETER(I,J)=TQC.NET_ERR(J)

      DO K=1,TQC.NOLINE(J)
        QENG(I,J,K)=TQC.LINE_ENERGY(J,K)
        QWDTH(I,J,K)=TQC.WIDTH(J,K)
        QMC(I,J,K)=TQC.LINE_MC(J,K)
        QMCER(I,J,K)=TQC.LINE_ERR(J,K)
        QFLG(I,J,K)=TQC.FLAG(J,K)
      ENDDO
    ENDDO
  ENDDO

```

C NOW PRINT THE NUCLIDE INFORMATION
C

```
DO J=1,TQC.NONE
    WRITE(6,9520)TQC.NID(J),TQC.NA(J),QDATE(1),
1 QTIME(1),QENG(1,J,1),QWDTH(1,J,1),QMC(1,J,1),
2 QMCER(1,J,1),QFLG(1,J,1)

9520 FORMAT(/,1X,A4,I3,T11,A6,T19,A4,T30,F7.2,T41,F4.1,T47,1PE11.4,
1 T61,0PF6.1,T72,I4)

    DO I=2,NOR
        WRITE(6,9530)QDATE(I),QTIME(I),QENG(I,J,1),
1 QWDTH(I,J,1),QMC(I,J,1),QMCER(I,J,1),QFLG(I,J,1)

9530 FORMAT(T11,A6,T19,A4,T30,F7.2,T41,F4.1,T47,
1 1PE11.4,T61,0PF6.1,T72,I4)
    ENDDO

    DO K=2,TQC.NOLINE(J)
        WRITE(6,9532)

9532 FORMAT(1X)
        DO I=1,NOR
            WRITE(6,9530)QDATE(I),QTIME(I),QENG(I,J,K),
1 QWDTH(I,J,K),QMC(I,J,K),QMCER(I,J,K),QFLG(I,J,K)
        ENDDO
    ENDDO
    WRITE(6,9532)

C
C PRINT THE NET VALUES
DO I=1,NOR
    WRITE(6,9535)QDATE(I),QTIME(I),QNET(I,J),QNETER(I,J)

9535 FORMAT(T11,A6,T19,A4,T79,1PE11.4,T93,0PF6.1)
    ENDDO

ENDDO

C PRINT THE SPECTROMETER CALIBRATION DATA
WRITE(6,9550)
9550 FORMAT('1',//,T40,'SPECTROMETER CALIBRATION DATA',//,
1 T2,'DATE',T10,'TIME',T20,'ZERO',T32,'GAIN',T43,'LOW PULSER EQ.',
2 T62,'HIGH PULSER EQ.',/,1X,76('_'),/)

DO I=1,NOR

    WRITE(6,9560)QDATE(I),QTIME(I),QZERO(I),QGAIN(I),QLEQ(I),
1 QHEQ(I)
```

```

9560  FORMAT(1X,A6,T10,A4,T19,F7.2,T31,F7.3,T45,F7.2,T65,F7.2)
      ENDDO
      END

```

B.25 QA_REPORT_SUB Subroutine

```

C ROUTINE TO PRINT THE QUALITY CONTROL INFORMATION OUT OF THE QUALITY DATA
C FILE FOR A DETECTOR SYSTEM. THIS ROUTINE USES THE NEW QUALITY FILE
C STRUCTURE DEFINED IN THE QC_QUALITY.INC FILE.
C      EWK 2/9/94
C
C Subroutine version, QA_REPORT_SUB, created. Calling routine passes the
C detector slot id, D_S, and an identifier for the quality data file of
C interest, QNUCID. The name of the quality data file has been expanded from
C X1QC.DAT to X1AAQC.DAT to include QNUCID as AA (see also QUALITY_UPDATE.FOR).
C Also, instead of calling SPOOL, this subroutine opens a file on unit 6; then,
C before returning to the calling routine, if quality records are found, closes
C and prints it; using SPOOL resulted in the file not being printed until the
C user exited from the calling main routine. Renamed subroutine REMOVE to
C QA_REMOVE to avoid conflict with a VAXCTRL symbol named REMOVE. D. A. Femec,
C February 22, 1994.
C
      subroutine qa_report_sub(d_s,qnucid)
      INCLUDE      'QC_QUALITY.INC'
C
      REAL*4      QA
      REAL*4      AREA(5,5,500),LMC(5,5,500),LER(5,5,500),WIDTH(5,5,500)
      REAL*4      ZERO(500),EB(500),LPEQ(500),HPEQ(500),
      1 NET(5,500),NETER(5,500),VALUE(500),WA(500)
      REAL*4      AVA(5,5),AVLMC(5,5),AVW(5,5),TOT(8)
      REAL*4      SDW(5,5),SDA(5,5),SDLMC(5,5),AVNET(5),SDNET(5)
      CHARACTER*1 QAW(5,5,500),QAA(5,5,500),QALMC(5,5,500)
      CHARACTER*1 QAZ,QAEA,QAEB,QAEC,QAWA,QAWB,QANET,QALP,QAHP
      CHARACTER*6 CDATE(500)
      CHARACTER*40 QFILE /'SLOT$SPC:X1AAQC.DAT'/
      CHARACTER*6 START_DATE,END_DATE
      INTEGER*4   SY,DS,SM,EY,ED,EM,CY,CD,CM
      INTEGER*4   LIB$FIND_FILE,LIB$FIND_FILE_END,STATUS
      CHARACTER*2 D_S,qnucid
      LOGICAL     FNFD /.TRUE./
      LOGICAL     PULSER /.FALSE./ !ANY PULSER VALUES
C
100  CALL LIB$GET_INPUT(START_DATE,
      1 'Enter starting date of report (MMDDYY): ',NOC)

```

```

      IF(NOC .NE. 6) THEN
110        TYPE *, ' BAD INPUT--PLEASE REENTER'
          GOTO 100
      ENDIF
C CHECK START DATE FOR VALID INPUT
      READ(UNIT=START_DATE(1:2), FMT=*, ERR=110) SM
      READ(UNIT=START_DATE(3:4), FMT=*, ERR=110) SD
      READ(UNIT=START_DATE(5:6), FMT=*, ERR=110) SY
      IF(SY.LT.90) GOTO 110
      IF(SM .LT. 1 .OR. SM .GT. 12) GOTO 110
      IF(SD .LT. 1 .OR. SD .GT. 31) GOTO 110
C
200    CALL LIB$GET_INPUT(END_DATE,
      1 'Enter ending date of report (MMDDYY): ', NOC)
      IF(NOC .NE. 6) THEN
210        TYPE *, ' BAD INPUT--PLEASE REENTER'
          GOTO 200
      ENDIF
C CHECK END DATE FOR VALID INPUT
      READ(UNIT=END_DATE(1:2), FMT=*, ERR=210) EM
      READ(UNIT=END_DATE(3:4), FMT=*, ERR=210) ED
      READ(UNIT=END_DATE(5:6), FMT=*, ERR=210) EY
      IF(EY.LT.90 .OR. EY .LT. SY) GOTO 210
      IF(EM .LT. 1 .OR. EM .GT. 12) GOTO 210
      IF(ED .LT. 1 .OR. ED .GT. 31) GOTO 210
C
      open(6, file=(d_s//'_ '//qnucid//'_qa_report.lis'), status='new',
        . carriagecontrol='fortran')
      QFILE(10:13)=D_S//qnucid
C
C START READING THE CONTENTS OF THE FILE FIND THOSE ENTRIES WITH IN THE
C SPECIFIED INTERVAL START_DATE TO END_DATE
      NOE=0 !NUMBER OF ENTRIES WITHIN START AND END DATES
      OPEN(UNIT=1, NAME=QFILE, FORM='UNFORMATTED', TYPE='OLD')
      REWIND(UNIT=1)
300    READ(1, END=1000) QC
      READ(UNIT=QC.DATE(1:2), FMT=*) CM
      READ(UNIT=QC.DATE(3:4), FMT=*) CD
      READ(UNIT=QC.DATE(5:6), FMT=*) CY
C
      IF(CY .GT. EY) GOTO 300
      IF(CY .LT. SY) GOTO 300
      IF((CY .EQ. SY) .AND. (CM .LT. SM)) GOTO 300
      IF((CM .EQ. SM) .AND. (CD .LT. SD)) GOTO 300
C
      IF(EY .GT. CY) GOTO 300
      IF((EY .EQ. CY) .AND. (CM .GT. EM)) GOTO 300

```

```

        IF((CM .EQ. EM) .AND. (CD .GT. ED))GOTO 300
C
C FOUND AN ENTRY WHICH FALL WITHIN SPECIFIED DATES
        FNFD=.FALSE.
        NOE=NOE+1
        IF(NOE .GT. 500)THEN
            TYPE *, ' OVER 500 ENTRIES IN PERIOD--REDUCE INTERVAL'
            close(6,status='delete')
            RETURN
        ENDIF
C
DO I=1,QC.NONE
    DO J=1,QC.NOLINE(I)
        AREA(I,J,NOE)=QC.AREA(I,J)
        LMC(I,J,NOE)=QC.LINE_MC(I,J)
        LER(I,J,NOE)=QC.LINE_ERR(I,J)
        WIDTH(I,J,NOE)=QC.WIDTH(I,J)
    ENDDO
    NET(I,NOE)=QC.NET_MC(I)
    NETER(I,NOE)=QC.NET_ERR(I)
ENDDO

ZERO(NOE)=QC.ZERO
EB(NOE)=QC.B_TERM
WA(NOE)=QC.W_ZERO
CDATE(NOE)=QC.DATE

IF(QC.PULSER)THEN
    LPEQ(NOE)=QC.LOW_PULSER_EQ
    HPEQ(NOE)=QC.HIGH_PULSER_EQ
    PULSER=.TRUE.
ENDIF

GOTO 300
C
C ALL ENTRIES IN FILE HAVE BEEN FOUND AND THEIR VALUES LOADED IN THE ARRAYS
C COMPUTE THE AVERAGES OF THE VARIOUS ENTRIES
1000 IF(NOE .EQ. 0)GOTO 5000 !NO ENTRIES FOR THIS SOURCE

DO I=1,QC.NONE
    DO J=1,QC.NOLINE(I)
        TOT(1)=0.0
        TOT(2)=0.0
        TOT(3)=0.0

        DO K=1,NOE
            TOT(1)=TOT(1)+WIDTH(I,J,K)

```



```

                TOT(2)=TOT(2)+AREA(I,J,K)
                TOT(3)=TOT(3)+LMC(I,J,K)
            ENDDO
            AVW(I,J)=TOT(1)/NOE
            AVA(I,J)=TOT(2)/NOE
            AVLMC(I,J)=TOT(3)/NOE
        ENDDO

        AVNET(I)=0.0
        DO J=1,NOE
            AVNET(I)=AVNET(I) + NET(I,J)
        ENDDO
        AVNET(I)=AVNET(I)/NOE
    ENDDO

    AVZ=0.0
    AVEB=0.0
    AVL PQ=0.0
    AVHPQ=0.0
    AVWA=0.0
    DO I=1,NOE
        AVZ=AVZ+ZERO(I)
        AVEB=AVEB+EB(I)
        AVWA=AVWA+WA(I)
        AVL PQ=AVL PQ+LPEQ(I)
        AVHPQ=AVHPQ+HPEQ(I)
    ENDDO
    AVZ=AVZ/NOE
    AVEB=AVEB/NOE
    AVWA=AVWA/NOE
    AVL PQ=AVL PQ/NOE
    AVHPQ=AVHPQ/NOE

C
C COMPUTE STANDARD DEVIATION IF THERE ARE MORE THAN 1 ENTRY
    IF(NOE .GT. 1) THEN
        DO I=1,QC.NONE
            DO J=1,QC.NOLINE(I)
                DO K=1,NOE
                    VALUE(K)=WIDTH(I,J,K)
                ENDDO
                CALL STD_DEV(NOE,VALUE,SDW(I,J))
                DO K=1,NOE
                    VALUE(K)=AREA(I,J,K)
                ENDDO
                CALL STD_DEV(NOE,VALUE,SDA(I,J))
                DO K=1,NOE
                    VALUE(K)=LMC(I,J,K)

```

```

                ENDDO
                CALL STD_DEV(NO, VALUE, SDLMC(I, J))
            ENDDO

            DO K=1, NO
                VALUE(K)=NET(I, K)
            ENDDO
            CALL STD_DEV(NO, VALUE, SDNET(I))
        ENDDO

        CALL STD_DEV(NO, ZERO, SDZ)
        CALL STD_DEV(NO, EB, SDEB)
        CALL STD_DEV(NO, WA, SDWA)

        IF(PULSER) THEN
            CALL STD_DEV(NO, LPEQ, SDLPQ)
            CALL STD_DEV(NO, HPEQ, SDHPQ)
        ENDIF
    ENDIF

C
C COMPUTE QUALITY CONTROL FLAG FOR EACH LINE
DO I=1, QC.NONE
    DO J=1, QC.NOLINE(I)
        DO K=1, NO
            QAW(I, J, K)=' '
            QAA(I, J, K)=' '
            QALMC(I, J, K)=' '
            IF(NO .LE. 2)GOTO 2000
        DO KK=1, NO
            ITMP=(WDTH(I, J, KK)+0.05)*10.0
            VALUE(KK)=ITMP/10.0
        ENDDO
        CALL qa_REMOVE(K, NO, VALUE, TEMP)
        IF(TEMP .GT. 3.0)QAW(I, J, K)='*'
        IF(TEMP .GT. 4.0)QAW(I, J, K)='#'
    ENDDO
ENDDO

C COMPUTE STD DEV INDEPENDANT OF THIS VALUE
DO KK=1, NO
    VALUE(KK)=AREA(I, J, KK)
ENDDO
CALL qa_REMOVE(K, NO, VALUE, TEMP)
IF(TEMP .GT. 3.0)QAA(I, J, K)='*'
IF(TEMP .GT. 4.0)QAA(I, J, K)='#'

```

C COMPUTE STD DEV INDEPENDANT OF THIS VALUE

```

                DO KK=1,NOE
                  VALUE(KK)=LMC(I,J,KK)
                ENDDO
                CALL qa_REMOVE(K,NOE,VALUE,TEMP)
                IF(TEMP .GT. 3.0)QALMC(I,J,K)='*'
                IF(TEMP .GT. 4.0)QALMC(I,J,K)='#'
2000          ENDDO
        ENDDO
    ENDDO
C
C START THE PRINT OF EVERY THING
    WRITE(6,9000)QFILE(10:11),START_DATE,END_DATE
9000    FORMAT('1',////////,T30,'QUALITY CONTROL DATA SHEET',
      1 //,T38,'DETECTOR: ',A2,
      2 //,T39,'FOR PERIOD',/,T36,A6,' TO ',A6,/)

    WRITE(6,9010)
9010    FORMAT(/,2X,'NUCLIDE',T15,'DATE',T24,'ENERGY',T34,'WIDTH',
      1 T46,'AREA',T53,'uCURIE',T65,'%ERR')

    DO I=1,QC.NONE
      DO K=1,QC.NOLINE(I)
        WRITE(6,9015)QC.NID(I),QC.NA(I),CDATE(1),
      1 QC.LINE_ENERGY(I,K),WDTH(I,K,1),QAW(I,K,1),NINT(AREA(I,K,1)),
      2 QAA(I,K,1),LMC(I,K,1),QALMC(I,K,1),LER(I,K,1)
9015    FORMAT(/,1X,A4,I3,T14,A6,T23,F7.2,T34,F5.2,A1,T42,I8,
      1 A1,T52,1PE9.2,A1,T64,0PF5.2)

        DO J=2,NOE
          WRITE(6,9020)CDATE(J),WDTH(I,K,J),
      1 QAW(I,K,J),NINT(AREA(I,K,J)),QAA(I,K,J),LMC(I,K,J),QALMC(I,K,J),
      2 LER(I,K,J)
9020    FORMAT(T14,A6,T34,F5.2,A1,T42,I8,A1,T52,1PE9.2,A1,T64,0PF5.2)
          ENDDO

          WRITE(6,9025)AVW(I,K),NINT(AVA(I,K)),AVLMC(I,K),
      1 SDW(I,K),NINT(SDA(I,K)),SDLMC(I,K)
9025    FORMAT(1X,75('_'),/,2X,'AVERAGES:',T34,F5.2,T42,I8,
      1 T52,1PE9.2,/,2X,'STD DEV:',T34,0PF5.2,T42,I8,T52,1PE9.2)

        ENDDO
      ENDDO
    ENDDO

```

```

C
  WRITE(6,9046)QFILE(10:11),START_DATE,END_DATE
9046  FORMAT('1',///,52X,'QUALITY CONTROL DATA SHEET',
        1 //,T60,'DETECTOR: ',A2,
        2 //,60X,'FOR PERIOD',/,57X,A6,' TO ',A6,/)

C PRINT THE NUCLIDE NET INFORMATION
  WRITE(6,9080)
9080  FORMAT(//,T30,'NUCLIDE',T42,'DATE',T55,'NET_uCURIE',T68,'%ERROR',
        1 //,T30,50('_'))
      DO I=1,QC.NONE
        WRITE(6,9082)QC.NID(I),QC.NA(I)
9082  FORMAT(T30,A4,I3)
      DO J=1,NOE
        QANET=' '
        IF(NOE .LE. 2)GOTO 2900
        DO K=1,NOE
          VALUE(K)=NET(I,K)
        ENDDO
        CALL qa_REMOVE(J,NOE,VALUE,SDTMP)
        IF(SDTMP .GT. 5.0)QANET='*'
        IF(SDTMP .GT. 6.0)QANET='#'
2900      WRITE(6,9085)CDATE(J),NET(I,J),QANET,NETER(I,J)

9085  FORMAT(T42,A6,T55,1PE9.3,A1,T68,0PF7.2)
      ENDDO

      WRITE(6,9087)AVNET(I),SDNET(I)
9087  FORMAT(T30,50('_'),/,T34,'AVERAGE:',T55,1PE9.2,/,T34,'STD DEV:',
        1 T55,1PE9.2,/)

      ENDDO

C WRITE SPECTROMETER INFO
  WRITE(6,9046)QFILE(10:11),START_DATE,END_DATE

  WRITE(6,9050)
9050  FORMAT(//,1X,'DATE',7X,'ENERGY ZERO',2X,'ENERGY_B',
        1 2X,'WIDTH_A',$)
      IF(PULSER)THEN
        WRITE(6,9051)
9051  FORMAT('+',3X,'LOW PEQ',3X,'HIGH PEQ',3X,'COMMENTS')
      ELSE
        WRITE(6,9052)
9052  FORMAT('+',25X,'COMMENTS')
      ENDIF

```

```

DO J=1,NOE
  QAZ=' '
  QAEB=' '
  QALP=' '
  QAHP=' '
  QAWA=' '
  IF(NOE .LE. 2)GOTO 3000
  DO I=1,NOE
    ITMP=(ZERO(I)+0.005)*100.0
    VALUE(I)=ITMP/100.0
  ENDDO
  CALL qa_REMOVE(J,NOE,VALUE,SDTMP)
  IF(SDTMP .GT. 3.0)QAZ='*'
  IF(SDTMP .GT. 4.0)QAZ='#'
  DO I=1,NOE
    ITMP=(EB(I)+0.005)*100.0
    VALUE(I)=ITMP/100.0
  ENDDO
  CALL qa_REMOVE(J,NOE,VALUE,SDTMP)
  IF(SDTMP .GT. 3.0)QAEB='*'
  IF(SDTMP .GT. 4.0)QAEB='#'

  DO I=1,NOE
    ITMP=(WA(I)+0.005)*100.0
    VALUE(I)=ITMP/100.0
  ENDDO
  CALL qa_REMOVE(J,NOE,VALUE,SDTMP)
  IF(SDTMP .GT. 3.0)QAWA='*'
  IF(SDTMP .GT. 4.0)QAWA='#'

  IF(PULSER)THEN
    CALL qa_REMOVE(J,NOE,LPEQ,SDTMP)
    IF(SDTMP .GT. 5.0)QALP='*'
    IF(SDTMP .GT. 6.0)QALP='#'
    CALL qa_REMOVE(J,NOE,HPEQ,SDTMP)
    IF(SDTMP .GT. 5.0)QAHP='*'
    IF(SDTMP .GT. 6.0)QAHP='#'
  ENDIF
3000  WRITE(6,9060)CDATE(J),ZERO(J),QAZ,EB(J),QAEB,
      1 WA(J),QAWA
9060  FORMAT(1X,A6,5X,F8.3,A1,4X,F7.3,A1,
      1 2X,F6.3,A1,$)

  IF(PULSER)THEN
    WRITE(6,9061)LPEQ(J),QALP,HPEQ(J),QAHP
9061  FORMAT('+',2X,F7.2,A1,3X,F7.2,A1,9X,34('_'))
    ELSE

```

```

          WRITE(6,9062)
9062  FORMAT('+',33X,30('_'))
      ENDIF

      ENDDO

      WRITE(6,9070)AVZ,AVEB,AVWA
9070  FORMAT(1X,71('_'),/,1X,'AVERAGES',3X,F8.3,3X,F9.3,3X,
1      F6.3,1X,$)
      IF(PULSER)THEN
          WRITE(6,9071)AVLPQ,AVHPQ
9071  FORMAT('+',3X,F7.2,4X,F7.2)
      ENDIF
      WRITE(6,9072)SDZ,SDEB,SDWA
9072  FORMAT(1X,'STD DEV',4X,0PF8.3,3X,F9.3,
1      3X,F6.3,1X,$)

      IF(PULSER)THEN
          WRITE(6,9071)SDLPQ,SDHPQ
      ELSE
          WRITE(6,9073)
9073  FORMAT('+',' ')
      ENDIF

C
      WRITE(6,9090)
9090  FORMAT(//,' NOTE:'2X,/,
1  3X,'1.) * Indicates a lower control limit (warning) value
2  that is out of statistical agreement with',/,
3  7X,' QA results:',/,
3  24X,' > 3 std. dev. for "Energy zero", "Width_A" and
3  "Energy_B" values.',/,
4  24X,' > 5 std. dev. for "Net Ucuries" and "Pulser" values.',/,
5  3x,'2.) # Indicates and upper control limit (action) value
6  that is out of statistical agreement with',/,
7  7X,' QA results:',/,
8  24X,' > 4 std. dev. for "Energy zero", "Width_A" and
8  "Energy_B" values.',/,
9  24X,' > 6 std. dev. for "Net Ucuries" and "Pulser" values.',/,
9  3x,'3.) The QA results =
9  (average value - current value/std.dev.).',/,
9  3x,'4.) Due to the extremely small variances between these
9  measured values',/,
9  7x,'it was necessary to set the control limit criteria
9  above the usual two and three standard deviation criteria.')

C
C END OF LOOP FOR ONE SOURCE
5000 IF(FNFD)THEN

```

```

        TYPE *, ' NO QUALITY FILE ENTRIES MEET DATES SPECIFIED'
        close(6,status='delete')
        RETURN
    ENDIF
    close(6,status='keep')
    call lib$spawn(('print/delete/noidentify '//d_s//'_ '//qnucid//
. '_qa_report.lis'),,,1)
    return
END

```

B.26 QA_REMOVE Subroutine

```

C ROUTINE TO REMOVE A VALUE FROM A LIST OF NUMBERS, COMPUTE A NEW AVERAGE
C AND STANDARD DEVIATION FROM THE NEW LIST THEN RETURN A QA FLAG VALUE.
C
C      EWK 1/10/90
C
C Renamed to QA_REMOVE to avoid conflict with a VAXCTRL symbol with the name
C REMOVE.  D. A. Femec, February 22, 1994.
C
C      SUBROUTINE qa_REMOVE (X,NE,OLD,QAVAL)
C WHERE:
C      X = INDEX VALUE TO REMOVE FROM ARRAY OLD
C      NE = NUMBER OF ENTRIES IN OLD ARRAY
C      OLD= ARRAY
C      QAVAL = QA FLAG VALUE
C
C      REAL*4 OLD(*),NEW(1000),QAVAL
C      INTEGER*4  X
C
C      K=1
C      DO I=1,NE
C          IF(I .NE. X) THEN
C              NEW(K)=OLD(I)
C              K=K+1
C          ENDIF
C      ENDDO
C
C COMPUTE AVERAGE
C      AVTMP=0.0
C      DO I=1,NE-1
C          AVTMP=AVTMP+NEW(I)
C      ENDDO
C      AVTMP=AVTMP/(NE-1)
C      CALL STD_DEV(NE-1,NEW,SDTMP)
C      QAVAL=QA(AVTMP,OLD(X),SDTMP)

```

RETURN
END

B.27 QUE_PRINT_DEL Subroutine

```
C ROUTINE TO QUE OUTPUT TO THE SYS$PRINT QUE
C          EWK 6/23/94
C ROUTINE TO QUE UP A PRINT JOB TO SYS$PRINT AND DELETE FILE
C
C          SUBROUTINE QUE_PRINT_DEL(FILE)
C
C WHERE: FILE =NAME OF FILE TO PRINT

INCLUDE '($SSDEF)'
INCLUDE '($LNMDEF)'
INCLUDE '($SJCDEF)'
INCLUDE '($JBCMSGDEF)'

STRUCTURE /ITEMLIST/
    INTEGER*2 BUFLTH          !BUFFER LENGTH
    INTEGER*2 ITMCDE          !ITEM CODE
    INTEGER*4 BUFADR          !BUFFER ADDRESS
    INTEGER*4 RTNLTH          !RETURN LENGTH ADDRESS
END STRUCTURE

C
RECORD /ITEMLIST/ SNDITM(7)

INTEGER*4  STATUS,SYS$SNDJBCW
CHARACTER*(*)  FILE !FILE TO PRINT NAME
CHARACTER*8  SETUP_NAME  //'LASER132'/
CHARACTER*9  LQUEUE      //'SYS$PRINT'/
INTEGER*4    IOSB(2)
INTEGER*4    DTIME(2) /-200000000,-1/

C QUE THE OUTPUT

SNDITM(1).BUFLTH = LEN(FILE)
SNDITM(1).ITMCDE = SJC$_FILE_SPECIFICATION
SNDITM(1).BUFADR = %LOC(FILE)
SNDITM(1).RTNLTH = 0

C
SNDITM(2).BUFLTH = LEN(SETUP_NAME)
SNDITM(2).ITMCDE = SJC$_FILE_SETUP_MODULES
SNDITM(2).BUFADR = %LOC(SETUP_NAME)
```



```

C      SNDITM(2).RTNLTH = 0

C      SNDITM(3).BUFLTH = 0
      SNDITM(3).ITMCDE = SJC$_NO_FILE_FLAG
      SNDITM(3).BUFADR = 0
      SNDITM(3).RTNLTH = 0

C      SNDITM(4).BUFLTH = 0
      SNDITM(4).ITMCDE = SJC$_DELETE_FILE
      SNDITM(4).BUFADR = 0
      SNDITM(4).RTNLTH = 0

      SNDITM(5).BUFLTH = LEN(LQUEUE)
      SNDITM(5).ITMCDE = SJC$_QUEUE
      SNDITM(5).BUFADR = %LOC(LQUEUE)
      SNDITM(5).RTNLTH = 0

C      SNDITM(6).BUFLTH = 8
      SNDITM(6).ITMCDE = SJC$_AFTER_TIME
      SNDITM(6).BUFADR = %LOC(DTIME)
      SNDITM(6).RTNLTH = 0

C      SNDITM(7).BUFLTH = 0
      SNDITM(7).ITMCDE = 0

C      STATUS = SYS$SNDJBCW
2      (,%VAL(SJC$_ENTER_FILE),%VAL(0),SNDITM,IOSB,,)
      IF (STATUS.NE.SS$_NORMAL) THEN
        CALL LIB$SIGNAL(%VAL(STATUS))
      ENDIF
      IF (IOSB(1).NE.JBC$_NORMAL) THEN
        CALL LIB$SIGNAL(%VAL(IOSB(1)))
      ENDIF

C      RETURN
      END

```

B.28 PEQ_EU Subroutine

```

C MODIFIED TO PUT THE LOW PULSER WIDTH IN THE CALIBRATION AND PARAMETER BLOCKS
C      EWK 1/11/91
C SUBROUTINE TO COMPUTE THE PULSER EQUIVALENT ENERGIES BASED ON A EU-152
C CALIBRATION SPECTRUM
C "PLS_ONE_KEV" AND "PLS_TWO_KEV" IN THE PARAMETER BLOCK FOR THE SPECTRUM
C SLOT IS UP DATED WITH THE NEW PULSER EQUIVALENT ENERGIES.

```

```

C
C          EWK 10/31/89
C
C Modified to be closer to PEQ_TH: "Press return to continue" at the end of
C the routine removed, parameter block read before the pulser equivalents are
C updated, and pulser equivalents written to the output file via WRITE (rather
C than TYPE) statements. D. A. Femec, June 24, 1993.
C
C          SUBROUTINE PEQ_EU(SPEC_ID,ERR)
C
C WHERE
C          SPEC_ID = THE FILE SPECIFICATION OF THE EU-152 SPECTRUM
C          ERR = THE ERROR FLAG LOGICAL
C
C          INCLUDE 'GAP$SRC:VHEAD.INC'
C          INCLUDE 'GAP$SRC:PKD.INC'
C          INCLUDE 'GAP$SRC:CALB.INC'
C          INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
C          CHARACTER*(*)      SPEC_ID
C          LOGICAL              ERR
C
C          REAL*4              PL(512),PH(512),DATA(128),PLUS(512),PA(128),PB(128),
C          1                    PC(128),PD(128)
C          REAL*8 MX(3,3),V(3),PP(3)
C          EQUIVALENCE (DATA,PLUS),(PA,PLUS),(PB,PLUS(129)),(PC,PLUS(257)),
C          1              (PD,PLUS(385))
C          CHARACTER*2 SID
C          CHARACTER*20        CLBFILE      //'SLOT$SPC: '//
C          ERR=.FALSE.
C ENERGY CALIBRATE THE SPECTRUM
C          CALL EU_CALB(SPEC_ID,ERR)
C          IF(ERR)RETURN
C          J=INDEX(SPEC_ID,'.')-11
C          SID=SPEC_ID(J:J+1)
C          CLBFILE(10:15)=SID(1:2)//'.CLB'
C          OPEN(UNIT=2,NAME=CLBFILE,TYPE='OLD',ACCESS='DIRECT',
C          1 RECORDSIZE=128)
C          READ(2'1)UC.CLB
C          CALL OPEN$SP(SPEC_ID,IERR)
C          VHEAD(ZW)=UC.CALB.FW1
C          VHEAD(LW)=UC.CALB.FW2
C          RESET=FGET(-1)
C          INC=VHEAD(NC)
C          IGAMMA_CUT=P.PLS_ONE_START-1
C
C ENERGY SCALE DETERMINATION NOW COMPLETE PROCEED WITH DETERMINING

```

```

C THE PULSER EQUIVALENTS. TO DO THAT WE MUST CONSTRUCT A PULSER PEAK ONLY
C SPECTRUM. THIS SPECTRUM MUST BE OPEN ON LUN=1 SO THAT THE PEAKS CAN BE
C FIT.
C
C RETRIVE PULSER DATA
C
    PLC=P.PLS_ONE_START-P.PLS_ONE_PS      !LOW PULSER OFF-SET
    PHC=P.PLS_TWO_START-P.PLS_TWO_PS      !UPPER PULSER OFF-SET
C CONTINUE THE PROCESS WITH THE PULSER INFO WHICH IS IN THE SPECTRUM
    J=P.PLS_ONE_START
    K=P.PLS_TWO_START
    DO I=1,P.PLS_CHANNELS
        PL(I)=FGET(J)
        PH(I)=FGET(K)
        J=J+1
        K=K+1
    ENDDO
C OPEN TEMPORY SPECTRUM FILE IN WHICH TO PUT THE PULSER SPECTRUM
    CLOSE(UNIT=1)
    OPEN(UNIT=1,NAME='SLOT$SPC:PEQ.DAT',TYPE='NEW',
    1 ACCESS='DIRECT',RECORDSIZE=128)
    DO I=1,128
        DATA(I)=0.0
    ENDDO
    DO I=7,(INC/128)
        WRITE(1'I)DATA
    ENDDO
    IL=INT(PLC/128.0)
    ILR=INT((PLC-(IL*128))+1)
    IH=INT(PHC/128.0)
    IHR=INT((PHC-(IH*128))+1)
    IL=IL+7
    IH=IH+7
    DO I=1,P.PLS_CHANNELS
        DATA(ILR)=PL(I)
        ILR=ILR+1
        IF(ILR.LE.128)GOTO 500
        WRITE(1'IL)DATA
        IL=IL+1
        ILR=1
        DO J=1,128
            DATA(J)=0.0
        ENDDO
500    ENDDO
        WRITE(1'IL)DATA
    DO I=1,128
        DATA(I)=0.0

```

```

        ENDDO
        DO I=1,P.PLS_CHANNELS
            DATA(IHR)=PH(I)
            IHR=IHR+1
            IF(IHR.LE.128)GOTO 540
            WRITE(1'IH)DATA
            IH=IH+1
            IHR=1
            DO J=1,128
                DATA(J)=0.0
            ENDDO
540    ENDDO
        WRITE(1'IH)DATA
        RESET=FGET(-1)
C FIND LOCATION OF PULSER PEAKS
C
        PKD(1,1,XV)=0.0
        PKD(2,1,XV)=0.0
        YMAX=0.0
        DO I=100,INC/2
            Y=FGET(I)
            IF(Y.GE.YMAX)THEN
                PKD(1,1,XV)=I
                YMAX=Y
            ENDIF
        ENDDO
        YMAX=0.0
        DO I=(INC/2)+1,IGAMMA_CUT
            Y=FGET(I)
            IF(Y.GE.YMAX)THEN
                PKD(2,1,XV)=I
                YMAX=Y
            ENDIF
        ENDDO
        VHEAD(NP)=2.0
        VHEAD(LW)=0.0      !LINEAR TERM OF WIDTH EQ.=0.0 SINCE PULSER
                           !PEAKS ARE THE SAME WIDTH EVERY WHERE
        P.NO_PULSER=0
C
C FIT THE PEAKS TWICE TO REFINE THE POSITIONS AND GET A BETTER FIT
        CALL GAUSS
        CALL GAUSS !VARYING WIDTH
        P.NO_PULSER=2
C
C CHECK RESULTS OF FIT TO DETERMINE IF EVERY THING OK
        DO I=1,2
            IF(PKD(I,3,FLG).NE.0.0)THEN

```

```

        TYPE*, ' PULSER AT ', PKD(I,1,XV),
1      ' WAS NOT FIT'
5555   CALL LIB$GET_INPUT(SID(1:1),
1     'Press "Return" key when ready to continue')
        CLOSE(UNIT=1, DISP='DELETE')
        CLOSE(UNIT=2)
        ERR=.TRUE.
        RETURN
      ENDIF
      IF(PKD(I,1,WV).GE.5.0) THEN
        TYPE *, ' PULSER AT ', PKD(I,1,XV), ' IS TOO WIDE'
        TYPE *, ' WITH A WIDTH OF: ', PKD(I,1,WV)
        GOTO 5555
      ENDIF
    ENDDO
    IF(ABS(PKD(1,1,AV)-PKD(2,1,AV)).GT.(0.1*PKD(1,1,AV))) THEN
      TYPE *, 'PULSER AREAS DO NOT AGREE'
      TYPE *, ' LOWER PULSER AREA: ', PKD(1,1,AV)
      TYPE *, ' UPPER PULSER AREA: ', PKD(2,1,AV)
      GOTO 5555
    ENDIF
  C
  WRITE(6,9010) PKD(1,1,WV), PKD(1,3,ERA), PKD(2,3,ERA)
9010  FORMAT(/,4X,'LOW PULSER WIDTH: ',F6.2,/,
1     4X,'LOW PULSER ENERGY EQUIVALENT: ',F9.3,/,
2     4X,'HIGH PULSER ENERGY EQUIVALENT: ',F9.3)
  C
  C UPDATE CALIBRATION BLOCK
    UC.CALB.LPEQ=PKD(1,3,ERA)
    UC.CALB.HPEQ=PKD(2,3,ERA)
    UC.CALB.CAL_PULSER_WD=PKD(1,1,WV)
    WRITE(2'1) UC
    CLOSE(UNIT=1, DISP='DELETE')
    CLOSE(UNIT=2)
  C
  C UPDATE PARAMETER BLOCK WITH PULSER EQUIVALENTS
    CALL OPEN$SP(SPEC_ID, IER)
    READ(1'2) P
    P.PLS_ONE_KEV=PKD(1,3,ERA)
    P.PLS_TWO_KEV=PKD(2,3,ERA)
    P.CAL_PULSER_WD=PKD(1,1,WV)
    WRITE(1'2) P
    CLOSE(UNIT=1)
    RETURN
  END

```

B.29 TYPE_FILE Subroutine

```

C ROUTINE TO TYPE THE CONTENTS OF A FILE
C
      SUBROUTINE TYPE_FILE(FILE)
C
C WHERE: FILE IS THE FILE ID TO TYPE
C
      CHARACTER*(*)      FILE
      CHARACTER*132 LINE
      INTEGER*4    TLUN

      INTEGER*4 LIB$GET_LUN,LIB$FREE_LUN

      CALL LIB$GET_LUN(LUN)
      OPEN(UNIT=LUN,NAME=FILE,IOSTAT=IOS,STATUS='OLD')
100    READ(LUN,9000,IOSTAT=IOS)N,LINE
9000   FORMAT(Q,A132)
      IF(IOS .NE. 0)THEN
          CLOSE(UNIT=LUN)
          CALL LIB$FREE_LUN(LUN)
          RETURN
      ENDIF
      IF(N .EQ. 0)THEN
          TYPE *, ' '
      ELSE
          TYPE 9100,LINE(1:N)
      ENDIF
9100   FORMAT(1X,A<N>)
      GOTO 100
      END

```

B.30 SUBMIT Subroutine

```
C ROUTINE TO SUBMIT A JOB TO SYS$BATCH USING THE JBC SYSTEM CALL
C                               EWK 7/5/94
C
C      SUBROUTINE SUBMIT(P1,P2,P3,P4,P5,P6)
C
C WHERE P1...P6 ARE THE PARAMETERS TO SUBMIT
C
C      INCLUDE '($JBCMSGDEF)'
C      INCLUDE '($SJCDEF)'
```

CHARACTER*(*) P1,P2,P3,P4,P5,P6

C STRUCTURE OF ITEMS

STRUCTURE /ITEMLIST/

INTEGER*2 BL

INTEGER*2 ITM

INTEGER*4 BFA

INTEGER*4 RTN

END STRUCTURE

RECORD /ITEMLIST/IL(20)

INTEGER*4 DTIME(2)/-20000000,-1/

INTEGER*4 STATUS

INTEGER*4 IOSB(2)

INTEGER*4 SYS\$SNDJBCW

C

C POPULATE ITEM LIST

IL(1).BL=LEN('SYS\$BATCH')

IL(1).ITM=SJC\$_QUEUE

IL(1).BFA=%LOC('SYS\$BATCH')

IL(1).RTN=0

IL(2).BL=LEN('INEL\$EXE:BATCH_GIDXC_ANALYSIS.COM')

IL(2).ITM=SJC\$_FILE_SPECIFICATION

IL(2).BFA=%LOC('INEL\$EXE:BATCH_GIDXC_ANALYSIS.COM')

IL(2).RTN=0

IL(3).BL=0

IL(3).ITM=SJC\$_NO_NOTIFY

IL(3).BFA=0

IL(3).RTN=0

IL(4).BL=LEN('\$USER:[OPERATOR]BATCH_GIDXC_ANALYSIS.LOG')

IL(4).ITM=SJC\$_LOG_SPECIFICATION

IL(4).BFA=%LOC('\$USER:[OPERATOR]BATCH_GIDXC_ANALYSIS.LOG')

IL(4).RTN=0

IL(5).BL=8

IL(5).ITM=SJC\$_AFTER_TIME

IL(5).BFA=%LOC(DTIME)

IL(5).RTN=0

IL(6).BL=0

IL(6).ITM=SJC\$_NO_LOG_SPOOL

IL(6).BFA=0
IL(6).RTN=0

IL(7).BL=0
IL(7).ITM=SJC\$_RESTART
IL(7).BFA=0
IL(7).RTN=0

C NOW LOAD THE PARAMTER VALUES

ILX=8

IF(P1(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P1)
IL(ILX).ITM=SJC\$_PARAMETER_1
IL(ILX).BFA=%LOC(P1)
IL(ILX).RTN=0
ILX=ILX + 1

ENDIF

IF(P2(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P2)
IL(ILX).ITM=SJC\$_PARAMETER_2
IL(ILX).BFA=%LOC(P2)
IL(ILX).RTN=0
ILX=ILX + 1

ENDIF

IF(P3(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P3)
IL(ILX).ITM=SJC\$_PARAMETER_3
IL(ILX).BFA=%LOC(P3)
IL(ILX).RTN=0
ILX=ILX + 1

ENDIF

IF(P4(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P4)
IL(ILX).ITM=SJC\$_PARAMETER_4
IL(ILX).BFA=%LOC(P4)
IL(ILX).RTN=0
ILX=ILX + 1

ENDIF

IF(P5(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P5)
IL(ILX).ITM=SJC\$_PARAMETER_5
IL(ILX).BFA=%LOC(P5)
IL(ILX).RTN=0
ILX=ILX + 1

ENDIF

IF(P6(1:1) .NE. ' ') THEN

IL(ILX).BL=LEN(P6)


```

        IL(ILX).ITM=SJC$_PARAMETER_6
        IL(ILX).BFA=%LOC(P6)
        IL(ILX).RTN=0
        ILX=ILX + 1
    ENDIF

    IL(ILX).BL=0
    IL(ILX).ITM=0
    IL(ILX).BFA=0
    IL(ILX).RTN=0

    STATUS=SYS$SNDJBCW(,%VAL(SJC$_ENTER_FILE),%VAL(0),IL,IOSB,,)
    IF(STATUS)STATUS=IOSB(1)
    IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))

    RETURN
    END

```

B.31 QUE_PRINT Subroutine

```

C ROUTINE TO QUE OUTPUT TO THE SYS$PRINT QUE
C     EWK 6/23/94
C ROUTINE TO QUE UP A PRINT JOB TO SYS$PRINT
C
    SUBROUTINE QUE_PRINT(FILE)
C
C WHERE: FILE =NAME OF FILE TO PRINT

    INCLUDE '($SSDEF)'
    INCLUDE '($LNMDEF)'
    INCLUDE '($SJCDEF)'
    INCLUDE '($JBCMSGDEF)'

    STRUCTURE /ITEMLIST/
        INTEGER*2 BUFLTH          !BUFFER LENGTH
        INTEGER*2 ITMCDE          !ITEM CODE
        INTEGER*4 BUFADR          !BUFFER ADDRESS
        INTEGER*4 RTNLTH          !RETURN LENGTH ADDRESS
    END STRUCTURE

C
    RECORD /ITEMLIST/ SNDITM(7)

    INTEGER*4 STATUS,SYS$SNDJBCW
    CHARACTER*(*) FILE !FILE TO PRINT NAME

```

```

CHARACTER*8 SETUP_NAME  /'LASER132'/
CHARACTER*9 LQUEUE      /'SYS$PRINT'/
INTEGER*4   IOSB(2)
INTEGER*4   DTIME(2) /-20000000,-1/

```

C QUE THE OUTPUT

```

SNDITM(1).BUFLTH = LEN(FILE)
SNDITM(1).ITMCDE = SJC$_FILE_SPECIFICATION
SNDITM(1).BUFADR = %LOC(FILE)
SNDITM(1).RTNLTH = 0

```

C

```

SNDITM(2).BUFLTH = LEN(SETUP_NAME)
SNDITM(2).ITMCDE = SJC$_FILE_SETUP_MODULES
SNDITM(2).BUFADR = %LOC(SETUP_NAME)
SNDITM(2).RTNLTH = 0

```

C

```

SNDITM(3).BUFLTH = 0
SNDITM(3).ITMCDE = SJC$_NO_FILE_FLAG
SNDITM(3).BUFADR = 0
SNDITM(3).RTNLTH = 0

```

C

```

SNDITM(4).BUFLTH = LEN(LQUEUE)
SNDITM(4).ITMCDE = SJC$_QUEUE
SNDITM(4).BUFADR = %LOC(LQUEUE)
SNDITM(4).RTNLTH = 0

```

C

```

SNDITM(5).BUFLTH = 8
SNDITM(5).ITMCDE = SJC$_AFTER_TIME
SNDITM(5).BUFADR = %LOC(DTIME)
SNDITM(5).RTNLTH = 0

```

C

```

SNDITM(6).BUFLTH = 0
SNDITM(6).ITMCDE = 0

```

C

```

STATUS = SYS$SNDJBCW
2      (, %VAL(SJC$_ENTER_FILE), %VAL(0), SNDITM, IOSB,,)
      IF (STATUS .NE. SS$_NORMAL) THEN
          CALL LIB$SIGNAL(%VAL(STATUS))
      ENDIF
      IF (IOSB(1) .NE. JBC$_NORMAL) THEN
          CALL LIB$SIGNAL(%VAL(IOSB(1)))
      ENDIF

```

C

```

RETURN
END

```

B.32 DCL Command File for Linking GC

```
$LINK/EXE=INEL$EXE GC, ENTER_PAR, GCMENU, STOP_DISPLAY, START_DISPLAY, -  
  EU_CALB, PEQ_EU, READ_SPECTRUM, ADCAST_GC, READ_STATUS, diagln, counts_mca, -  
  slosyn, qa_report_sub, qa_remove, quality_update, ASTIME, START_COUNTING, -  
  STOP_COUNTING, READTIB, SET_ERR_MSG, PLOT_SP, PEQ_TH, TH_CALB, -  
  QIOWL, QUE_PRINT, QUE_PRINT_DEL, SUBMIT, TYPE_FILE, -  
  GAP$SRC:GAUSS, OPEN$SP, FGET, SQLN, WEIGHT, ADCLINK/OPT, -  
  GAP$SRC:DAS_SUB, DAS_ID_SUB, DAS_MARK_SUB, DAS_PLOT_SUB, DAS_SIZE, COLOR, -  
  DEFINE_COLOR, MESSAGE, IDENTIFY_SUB, LIBRARY_LOOK, CLRMESS, -  
  CAL_PULSER, SEARCH, batch_cal_pulser, check_icode, d_fit, -  
  library_tag, qa, rsolve, std_dev, d_fit_dp, eff, lib_check, mcuries, -  
  ransum, libr_lookup, true_positive, -  
  GAP$LIB/LIB, VT100$LIB/LIB, LIB_CAM:NDLIB/LIB, OPT/OPT, -  
  SYS$INPUT/OPT  
  SYS$SHARE:DECW$XLIBSHR/SHARE  
$EXIT
```

B.33 ADC_STATUS Include File

```
C  
C STRUCTURE TO BE PUT IN A GLOBAL SECTION. TO BE USED FOR INTER PROCESS  
C COMUNICATION BETWEEN ROUTINES USED TO CONTROL THE ADC'S  
C  
  C  
    PARAMETER NOR=6          !NUMBER OF REMOTE ADC'S IN THE SYSTEM  
  STRUCTURE /ADC_STATUS/  
    INTEGER*4 REAL_TIME(6)    !COUNT REAL TIME  
    INTEGER*4 LIVE_TIME(6)    !COUNT LIVE TIME  
    INTEGER*4 AIM_NO(6)       !AIM MODULE NUMBER  
    INTEGER*4 ADC_NO(6)       !ADC NUMBER  
    INTEGER*4 COUNT_INTERVAL(6)!LENGTH OF COUNTING INTERVAL  
    INTEGER*4 START_RETRY(6)  !RETRY COUNTER START PULSER CHECKS  
    LOGICAL*1 MESSAGE_AVAILABLE(10) !MESSAGE AVAILABLE  
    LOGICAL*1 ON_OFF(6)       !ADC ON OR OFF TRUE=ON  
    LOGICAL*1 CHECK_PULSER(6)  !CHECK PULSERS DURING COUNT INTERVAL  
    LOGICAL*1 AUTO_COUNT(6)    !AUTO COUNT IN PROGRESS FLAG  
    CHARACTER*24 DUMMY         !NOT USED FILLS SECTION TO 3 PAGES  
    CHARACTER*4 START_TIME(6)  !COLLECTION START TIME MMHH  
    CHARACTER*6 START_DATE(6)  !COLLECTION START DATE MMDDYY  
    CHARACTER*40 SPECTRUM_ID(6) !FILE SPECIFICATION OF LAST SPECTRUM  
                                !WRITTEN TO DISK  
    CHARACTER*40 SAVE_DIR(6)   !LOGICAL POINTING TO SAVE AREA
```

```

        CHARACTER*80      MESSAGE(10)      !ERROR MESSAGE
END STRUCTURE
RECORD /ADC_STATUS/AS

C
COMMON /ADC_STATUS_COMMON/AS

```

B.34 AIM_GSS Include File

```

C STRUCTURE CONTAINS THE ON-LINE INFORMATION USED IN A GLOBAL SECTION FOR THE
C AIM MCA'S CONNECTED TO A COMPUTER.
C          EWK 5/7/91
C ARRAY INDEX ARRAY(I,J,K)
C   WHERE: I=1 FOR A AIMS
C          I=2 FOR B AIMS
C          I=3 FOR D AIMS
C          I=4 FOR R AIMS
C          J=1-4 FOR MCA'S
C          K=1 FOR ADC #0
C          K=2 FOR ADC #1
C

```

```

STRUCTURE /AIM_GSS/

```

```

    BYTE  STARTGSS

```

```

    BYTE  UNUSED(254) !UNUSED TO FILL OUT 17 BLOCKS
    LOGICAL    AIM_BEGIN(4,4,2) !TRUE IF MCA JUST STARTED
    LOGICAL    AIM_CHECK(4,4,2) !TRUE IF PULSER CHECK IS TO BE MADE
    LOGICAL ERROR(4,4,2)      !SHIFT OR OTHER ERROR

```

```

    CHARACTER*40      MESSAGE(4,4,2)      !IF PULSER CHECK FAILED MESSAGE HERE

```

```

    INTEGER*4  TIME(4,4,2) !TIME COUNT STARTED
    INTEGER*4  DATE(4,4,2) !DATE COUNT STARTED
    INTEGER*4  PRESET_TIME(4,4,2) !PRESET TIME IN SECONDS

```

```

C PULSER CHECK INFORMATION AND DATA

```

```

    INTEGER*4  NO_PUL_CHANNELS(4,4,2) !NUMBER OF CHANNELS IN
                                         !PULSER BUFFER

```

```

    INTEGER*4  START_LOW_CHANNEL(4,4,2) !STARTING CHANNEL OF
                                         !LOW PULSER BUFFER

```

```

    INTEGER*4  START_HIGH_CHANNEL(4,4,2) !STARTING CHANNEL OF

```

```

                                !HIGH PULSER BUFFER

INTEGER*4   CHECK_RETRY(4,4,2) !RETRY COUNTER IN CASE OF
                                !FAILURE

REAL*4      LOW_LAST(4,4,2,128) !LAST LOW PULSER BUFFER

REAL*4      HIGH_LAST(4,4,2,128) !LAST HIGH PULSER BUFFER

REAL*4      LOW_CENTER(4,4,2)    !CENTRIOD OF LOW PULSER AT
                                !START OF COUNT

REAL*4      HIGH_CENTER(4,4,2)   !CENTRIOD OF HIGH PULSER AT
                                !START OF COUNT

BYTE  ENDGSS

END STRUCTURE

RECORD /AIM_GSS/ AIM

COMMON /AIM_D_STATUS/ AIM

```

B.35 ANALYSIS_PARAMETER Include File

C STRUCTURE TO CONTAIN THE ANALYSIS PARAMETERS FOR AN GAMMA-RAY ANALYSIS
C IN THE WASTE RECOVERY SYSTEM.

```

C
  STRUCTURE /ANALYSIS_PARAMETER/
    CHARACTER*40  ISOTOPE_LIB  //'MASTER'/
    CHARACTER*40  REQUIRED_LIB   //'ZEROLIM'/
    CHARACTER*40  INTERFERENCE_LIB  //'NOINTLIB'/
    LOGICAL       BACK_CORRECT  /.FALSE./
    LOGICAL       PRINT_RESULTS  /.TRUE./
    LOGICAL       SAVE_RESULTS   /.TRUE./
    CHARACTER*40  SAVE_DIR      //'GE$SAVE'/
  END STRUCTURE

C
  RECORD        /ANALYSIS_PARAMETER/ AP
C

```

B.36 ASTTRM Include File

```
C
C Increased the length of TILINE to allow a "BA" command explicitly specifying
C up to six spectrum ids to be entered. D. A. Femec, January 31, 1994.
C
      INTEGER*4   TIITERM,TILNGTH
      INTEGER*4   IKEYTBL,KEY
      CHARACTER*80 TILINE
C
C COMMON USED TO PASS INPUT TO THIS PROGRAM BETWEEN INPUT AST ROUTINES
COMMON/RDINFO/TIITERM,TILNGTH,TILINE,KEY,IKEYTBL
```

B.37 CALIBRATION Include File

```
C STRUCTURE HOLDING THE CALIBRATION COEFF. FOR EACH OF THE DETECTORS
      STRUCTURE /GC_CLB/
      REAL CZERO(6)           !ENERGY ZERO IN CHANNELS
      REAL A(6)               !A TERM OF ENERGY EQU.
      REAL B(6)               !B TERM OF ENERGY EQU.
      REAL Q(6)               !C TERM OF ENERGY EQU.
      REAL FW1(6)             !ZERO TERM OF WIDTH EQU.
      REAL FW2(6)             !LINEAR TERM OF WIDTH EQU.
      REAL LPEQ(6)            !LOW PULSER EQUIVALENT
      REAL HPEQ(6)            !HIGH PULSER EQUIVALENT
      REAL NO_PULSER(6)
      REAL PULSER_RATE(6)      !PULSER RATE
      REAL PLS_CHANNELS(6)     !NO OF CHANNELS IN PULSER BUFFER
      REAL PLS_ONE_START(6)    !START CHANNEL OF FIRST PULSER
      REAL PLS_TWO_START(6)    !START CHANNEL OF SECOND PULSER
      REAL PLS_ONE_PS(6)       !FIRST PULSER OFFSET
      REAL PLS_TWO_PS(6)       !SECOND PULSER OFFSET
      REAL CAL_PULSER_WD(6)    !PULSER WIDTH AT CALIBRATION
      END STRUCTURE
      RECORD /GC_CLB/GCB
      COMMON/GCCAL/GCB
```

B.38 PULSER_CONFIG Include File

```
C
C STRUCTURE TO HOLD INFORMATION ABOUT THE PULSERS ON EACH OF THE DETECTOR
```

C SYSTEMS

C

```
STRUCTURE /PULSER_CONFIG/
  INTEGER*4 NO_PUL_CHANNELS(6)
  INTEGER*4 START_LOW_CHANNEL(6)
  INTEGER*4 START_HIGH_CHANNEL(6)
END STRUCTURE
RECORD /PULSER_CONFIG/PC
```

C

```
COMMON /PULSER_CONFIG_COMMON/PC
```

C

B.39 QC_QUALITY Include File

C STRUCTURE TO HOLD ONE RECORD OF DATA FROM THE QUALITY CONTROL FILE USED TO

C SAVE THE QUALITY CONTROL INFORMATION. THIS STRUCTURE CAN HOLD UP TO

C 5 DIFFERENT NUCLIDES USED FOR CALIBRATION WITH 10 LINES/NUCLIDE.

C

EWK 2/2/94

```
STRUCTURE /QC_QUALITY/
  CHARACTER*6 DATE !MMDDYY RECORD CREATED
  CHARACTER*4 TIME !HHMM RECORD CREATED
  CHARACTER*11 SPECTRUM_ID !CALIBRATION SPECTRUM ID
  REAL*4 ZERO !FOLLOWING ARE CALIBRATION CONSTANTS
  REAL*4 A_TERM
  REAL*4 B_TERM
  REAL*4 C_TERM
  REAL*4 W_ZERO
  REAL*4 W_LIN
  REAL*4 LOW_PULSER_EQ
  REAL*4 HIGH_PULSER_EQ
  LOGICAL PULSER !PULSERED SPECTRUM
  INTEGER*4 NONE !NUMBER OF DIFFERENT NUCLIDES IN CALIBA.
  INTEGER*4 NOLINE(5) !NUMBER OF LINES FOR EACH NUCLIDE
  REAL*4 NET_MC(5) !NET MC VALUE FOR EACH NUCLIDE
  REAL*4 NET_ERR(5) !NET ERR IN MC VALUE FOR EACH NULCIDE
  INTEGER*4 NA(5) !NUCLIDE "A"
  INTEGER*4 NZ(5) !NUCLIDE "Z"
  CHARACTER*4 NID(5) !NUCLIDE ID
```

C ENTRIES FOR EACH DIFFERENT NUCLIDE

```
REAL*4 CENTROID(5,10) !CENTROID 10 LINES MAX
REAL*4 WIDTH(5,10)
REAL*4 LINE_ENERGY(5,10)
REAL*4 LINE_MC(5,10)
```

```

REAL*4          LINE_ERR(5,10)
REAL*4          AREA(5,10)
INTEGER*4       FLAG(5,10)

END STRUCTURE
RECORD /QC_QUALITY/QC
COMMON         /QC_COMMON/QC

```

B.40 THOR_QUALITY Include File

C THIS STRUCTURE DESCRIBES THE RECORDS IN THE THORIUM QUALITY FILE. AS EACH
C THORIUM IS RUN A NEW RECORD WILL BE MADE IN THE FILE.

C EWK 12/27/89

C

```

STRUCTURE /THOR_QUALITY/
  CHARACTER*6 DATE !MDDYY
  CHARACTER*4 TIME !HHMM
  CHARACTER*4 SOURCE_NUMBER !THORIUM SOURCE NUMBER
  CHARACTER*11 SPECTRUM_ID !THORIUM SPECTRUM ID
  REAL*4 ZERO !ENERGY EQUATION COEFFICIENTS
  REAL*4 A_TERM
  REAL*4 B_TERM
  REAL*4 C_TERM
  REAL*4 W_ZERO !WIDTH EQUATION COEFFICIENTS
  REAL*4 W_LIN
  LOGICAL PULSER !PULSERED SYSTEM
  REAL*4 LOW_PULSER_EQ !LOW PULSER ENERGY EQUIVALENT
  REAL*4 HIGH_PULSER_EQ !HIGH PULSER ENERGY EQUIVALENT
  REAL*4 CENTRIOD(5) !CENTROID OF TH LINES
  REAL*4 WIDTH(5) !WIDTHS OF TH LINES
  REAL*4 LINE_MC(5) !MICRO-CURIE VALUES FOR EACH LINE
  REAL*4 LINE_ERR(5) !ERROR IN MICRO-CURIE VALUE
  REAL*4 AREA(5) !AREA OF TH LINES
  INTEGER*4 FLAGS(5) !FLAGS ASSOCIATED WITH TH LINES
  REAL*4 NET_MC !NET MICRO-CURIE VALUE
  REAL*4 NET_ERR !ERROR IN NET MICRO-CURIE
END STRUCTURE

```

C

RECORD /THOR_QUALITY/TQ

C

B.41 GC_USER_INITIALIZATION DCL Command File


```

$ ! Routine to prepare a user's root directory for running GC by creating a
$ ! number of needed communication files.
$ !
$ ! Coded by D. A. Femec, August 1, 1994.
$ !
$ ! Make sure the rest of this routine is executed from the user's root
$ ! directory, pointed to by the logical SYS$LOGIN.
$ !
$ set default sys$login
$ !
$ ! Create the needed communication files (more can be added to this list as
$ ! needed).
$ !
$ create a1.input /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a1.output /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a2.input /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a2.output /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a3.input /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a3.output /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a4.input /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create a4.output /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create gidxc.input /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create gidxc.output /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create slosyn.dat /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create slosyn_spawn_in.dat /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ create slosyn_spawn_ot.dat /protection=(s:rwed,o:rwed,g:rwed,w:re)
$ !
$ ! Display a confirmatory message before exiting.
$ !
$ write sys$output " "
$ write sys$output "User's root directory successfully prepared for executing GC."
$ exit

```

Appendix C. GIDXC-Related Source Code and Other Listings

The following listings include the main routines BATCH_GIDXC_ANALYSIS and BATCH_GIDXC_SUMMARY and their developer-written subroutines and functions.

C.1 BATCH_GIDXC_ANALYSIS Main Routine

```
      program batch_gidxc_analysis
C
C DO A BATCH ANALYSIS FOR THE A1 DETECTOR SYSTEM FOR THE SWEPP GC ROUTINE
C      EWK 4/9/92
C MODIFIED TO ACCOMODATE THE NEW WIDTH CALCULATION USING THE PULSERS
C      EWK 1/14/91
C
C THIS PROGRAM IS MEANT TO BE RUN IN BATCH MODE USING THE CONCURRENT
C BACKGROUND ANALYSIS TECHNIQUE.
C THE SPECTRUM ID WILL BE PASSED IN LOGICAL A1$SPECTRUM$ID
C
C THE CBAT ALGORITHM WILL BE USED FOR BACKGROUND SUBTRACTION
C THE BACKGROUNDS A LOCATED IN DIRECTORIES POINTED TO BY THE LOGICALS
C      A<n>$BKG
C THE THREE MOST RECENT SPECTRA IN THAT DETECTORS DIRECTORY WILL BE USED
C
C      THE ANALYSIS PARAMETERS WILL BE LOCATED IN FILE
C      SLOT$SPC:A<n> .ANP
C
C      EWK 4/2/90
C
C BATCH_GIDXC_ANALYSIS created, based on E. W. Killian's BATCH_GI_ANALYSIS.
C Modified to perform double matrix correction based on Pu-239 and apply it to
C Pu-239, -240, and -241, Am-241, and U-233, -235, and -238 accordingly as per
C DXC (E. W. Killian's program to perform double matrix correction for Pu and
C Am radionuclides); some additional comments about the algorithm have been
C added. The external matrix attenuation factors are obtained via MU_LOOK
C where P.SYS_VOLUME_UNITS contains the applicable matrix, while the self-
C attenuation factors (using the Pu table) are obtained via the new derivative
C routine MU_SELF_LOOK. Also calls XTP_REGION to generate desired XTP-like
C plots of spectral regions. Changed name of print routine from CBAT_PRTWR to
C CBAT_PRGIDXC (a derivative of CBAT_PRTWR). Added the Pu-238:Pu-241 ratio to
C the summary report; the ratio is based on the Pu-238 152-keV line and the Pu-
C 241 148-keV line, as per L. V. East's NOTEGRAM of July 6, 1993. Eliminated
C the repetitious coding for calculating ratios via an array-based construct.
C Modified so that in the case of a lead-lined barrel, no isotopic mass ratios
C are calculated. Modified so that in the case of excessive U-235, the Pu-
```

C 241:Pu-239 mass isotopic ratio will be based on the 161.5-keV line for Pu-
C 239; also, the Pu-240:Pu-239, Am-241:Pu-239, and U-235:Pu-239 isotopic mass
C ratios will be calculated. Modified to calculate the matrix attenuation
C correction factor x-effective only for Eu-152 and to apply it to the
C appropriate radionuclides. Corrected an error by which the branching ratio
C for a desired radionuclide was actually the ratio for the first radionuclide
C tagged for a given line and not necessarily the correct one. D. A. Femec,
C November 29, 1993.

C
C Added a fourth line to the x- and t-effective calculations for Pu-239, namely
C the 345.01-keV line (which is interfered with by an Eu-152 line and so the
C appropriate addition was made to the interference library). A linear least
C squares method is now used for the calculation of x- and t-effective (see the
C routine LINFIT) so that uncertainties in these factors can be generated and
C the number of lines used in the calculations extended; accordingly, the
C matrices used in calculating these factors have been extended to contain at
C most ten elements. Standard error propagation has been included for the mass
C isotopic ratios (which will be presented as percents rather than as ratios)
C and the point source assay value; the coding for calculating the distributed
C source assay value has been disabled because the methodology is still under
C review. Added isotopic ratio structure (IR), which is also written to the
C BATCH\$RESULTS file; this structure includes a three-character flag indicating
C the source and any fitting warnings relevant to the specific ratio. Reworked
C how ratios-to-be-calculated are stored to ease adding new ones (the method to
C be used will specify which second-dimension element to access from the
C appropriate arrays). Added propagation of the errors in x- and t-effective
C into the line microCurie errors after interference decontamination and
C calculation of the isotopic mass ratios but before calculating net activities
C so that the propagated errors are automatically taken into account when
C calculating the assay values); because of this, the call to RSOLVE has been
C moved to after the ratios are calculated, therefore the "line microCurie"
C values have not been calculated and the appropriate conversions for the
C values and there errors are performed after the calls to FIND_LINE. Modified
C to include the attenuation table used in the sample description sections.
C Replaced detection-limit test with true-positive test when determining
C ratios; also eliminated the linear-fit test. Modified to generate printed
C output only if the user_mode logical IF\$PRINT is set to true; XTP plots will
C be generated independent of IF\$PRINT. Added the program version number
C (starting at 01.00) and the date the program was run to the banner.
C D. A. Femec, June 22, 1994.

C
C Refer to Engineering Design File #RWM-533, July 23, 1992, "Relative Isotopic
C Mass Ratio Gamma Measurements," L. O. Johnson, for information regarding the
C isotopic mass ratio and Pu-239 mass calculations.

C
C

C This program, modified as per the above, constitutes version 01.00 of the

```

C SWEPP version of BATCH_GIDXC_ANALYSIS. D. A. Femec, July 28, 1994.
C
C
C Declaration statements.
C
C INCLUDE ANALYSIS COMMONS
    INCLUDE '($LNMDEF)'
    INCLUDE 'GAP$SRC:VHEAD.INC'
    INCLUDE 'GAP$SRC:PKD.INC'
    INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
    INCLUDE 'GAP$SRC:BACK_FIT.INC'
    INCLUDE 'GAP$SRC:BACKGROUND_RESULTS.INC'
    INCLUDE 'ANALYSIS_PARAMETER.INC/LIST'
    INCLUDE 'FIND_LINE_STRUC.INC'
C
C STRUCTURE USE TO SAVE THE RESULTS OF A SPECTRUM ANALYSIS
C     EWK 5/16/90
    STRUCTURE /TWR_RESULTS/
        INTEGER*4  NPS      !NUMBER OF PEAKS ANALYZED
C
C RAW PEAK DATA
    REAL*4
        PEAK(200,4) !PEAK ANALYSIS RESULTS
        !PEAK(I,1)=ENERGY
        !PEAK(I,2)=COUNTS/SEC IN PEAK
        !PEAK(I,3)=% ERROR
        !PEAK(I,4)=EFFICIENCY @ ENERGY
C NUCLIDE DATA
    INTEGER*4  NOE      !NUMBER OF NUCLIDES IDENTIFIED
    INTEGER*4  Z(50)    !NUCLIDE Z
    INTEGER*4  A(50)    !NUCLIDE A
    CHARACTER*4 ID(50)  !NUCLIDE ALPHA NUMERIC ID
    REAL*4     MC(50)    !NET NUCLIDE MICRO-CURIES
    REAL*4     ERR(50)   !ERROR IN MICRO-CURIES
    END STRUCTURE
    RECORD      /TWR_RESULTS/TR
C
C Structure for recording isotopic ratios from an individual detector.
C
    structure /isorat/
        integer*4 nratio !Number of ratios calculated.
        integer*4 rat_al(100) !For the 1st isotope of the ratio: nuclide A
        character*4 rat_il(100) ! and ID.
        integer*4 rat_a2(100) !For the 2nd isotope of the ratio: nuclide A
        character*4 rat_i2(100) ! and ID.
        real*4 ratios(100) !Ratios (as percentages)
        real*4 uncrat(100) ! and uncertainties in the ratios (also as percentages).
C

```

```

C      Ratio source and fitting warning flag:  first character of S=standard,
C      U=excessive uranium, L=lead liner present; second character=which line
C      was fit using linear methods; third character=which line was below
C      detection limits.
C
      character*4 ratflg(100)
      end structure
record /isorat/ ir

CHARACTER*85      COMMENT

INTEGER*4  SDATE,STIME,CDATE,CTIME
CHARACTER*4 STM,CTM

INTEGER*4 LIB$GET_LUN,LIB$FREE_LUN,STATUS
REAL      DATA(128)
CHARACTER*1 NULL
CHARACTER*10 ACU
CHARACTER*512  HEADER
CHARACTER*80   ERR_MESS      !ERROR MESSAGE BUFFER
CHARACTER*40   PFILE  /'SLOT$SPC:A1.ANP'/ !ANALYSIS PARAMETER FILE
CHARACTER*2 DS  !DETECTOR SYSTEM
CHARACTER*4 SID
REAL*4          R
EQUIVALENCE (R,SID)
C
EQUIVALENCE (INULL,NULL)
C
COMMON /TABLES/ NEFF,EFFID,NLIN,LINID
DATA INULL /0/
C
C STRUCTURE USED TO RUN SYSTEM CALL $TRNLNM
STRUCTURE /ITEM_LIST/
      INTEGER*2  BL
      INTEGER*2  ITEM
      INTEGER*4  ADRESS
      INTEGER*4  RET
      INTEGER*4  END    /0/
END STRUCTURE
RECORD /ITEM_LIST/IL
C
CHARACTER*30      SP_NAME
INTEGER*4  SP_NAME_LN,SYS$TRNLNM,lifprt
CHARACTER*80      SP_FILE
character*9 curdat  !Date program was run.
character*1 cifprt  !Should printed output be generated (default=Y)?

```

C CONSTANTS AND VARIABLES USED TO PERFORM DOUBLE MATRIX CORRECTIONS

```
C
  character*24 sigfig,cx_efc,ct_efc,cerati,cPu239  !SIGFIG versions for x- and t-effective.
  LOGICAL      TRUE_POSITIVE
  REAL*8       T1,T2,T3,T4,T5
  REAL*4       XIN(128),XC(128)
```

```
  integer*2 numefc  !Number of sets of X- and T-effective to be calculated.
```

```
C
C ENERGY OF LINES USED TO DETERMINE X-EFFECTIVE AND T-EFFECTIVE (be sure to
C leave a zero as the last value in each row of required energies).
```

```
C
  REAL*4       REQ_ENG(10,10) /129.30, 203.55, 345.01, 413.71, 6*0.00,  !Based on Pu-239.
  1            778.90,1408.02,8*0.00,  !Based on Eu-152.
  2            143.77, 163.36, 185.72, 205.32, 76*0.00/  !Based on U-235.
```

```
  REAL*4       KV(10,10)  !KV(1,I)=BR*EFF(REQ_ENG(1,I)), . . . .
```

```
  REAL*4       BR(10,10)  !RADIONUCLIDE'S BRANCHING RATIO @ REQ_ENG(1,I), . . . .
```

```
  REAL*4       S(10,10)   !S(1,I)=MATRIX MU @ REQ_ENG(1,I), . . . .
```

```
  REAL*4       PV(10,10)  !PV(1,I)=SELF MU @ REQ_ENG(1,I), . . . .
```

```
  REAL*4       CS(10,10)  !CS(1,I)=CPS @ REQ_ENG(1,I), . . . .
```

```
  real*8 lngps(10),  !Ln of gammas-per-second (LN(CS/KV)) - the y data.
.  ulngps(10)/10*1.0/,  !The uncertainty in the above.
.  xdata(3,10),  !The x data.
.  xtefcs(3),xtstdv(3),  !For returning x- (2) and t-effective (3) from LSTSQR.
.  stdevY,  !The standard deviation between observed and predicted Ys.
.  x_efc(3),ux_efc(3),t_efc(3),ut_efc(3),  !X- and T-effectives and uncertainties.
.  pcdgps(3)  !Goodness-of-fit for corrected values.
```

```
C FOLLOWING IS A LIST OF THE NUCLIDES FOR WHICH THE DOUBLE MATRIX CORRECTION IS
C GOING TO BE PERFORMED. The first radionuclide in each list is the one on
C which the double matrix correction is based.
```

```
C
  CHARACTER*4 dxc_ID(8,3) /'PU ','PU ','PU ','PU ',
  1            'AM ','U ','U ','U ',
  2            'EU ',' ',' ',' ',
  3            ' ',' ',' ',' ',
  4            'U ',' ',' ',' ',
  5            ' ',' ',' ',' '
  INTEGER*4   dxc_A(8,3) / 239, 238, 240, 241,
  1            241, 233, 235, 238,
  2            152, 0, 0, 0,
```

3		0,	0,	0,	0,
4		235,	0,	0,	0,
5		0,	0,	0,	0/
INTEGER*4	dxz_Z(8,3)	/	94,	94,	94,
1			95,	92,	92,
2			63,	0,	0,
3			0,	0,	0,
4			92,	0,	0,
5			0,	0,	0/

Spectral region(s) for plotting via DPY_REGION.

```
real*4 estart(1)/120.0/,eend(1)/200.0/
integer*2 numxtp/1/
```

Arrays for isotopic ratios (nuclide 1:nuclide 2) to be calculated:

Standard set:

Pu-238 (152.72 keV) : Pu-241 (148.54 keV),
 Pu-241 (148.54 keV) : Pu-239 (144.20 keV),
 Am-241 (146.55 keV) : Pu-239 (144.20 keV), and
 Pu-240 (160.31 keV) : Pu-239 (161.48 keV).

Excess U-235 set:

Pu-240 (160.31 keV) : Pu-239 (161.48 keV),
 Pu-241 (148.54 keV) : Pu-239 (161.48 keV),
 Am-241 (125.29 keV) : Pu-239 (129.30 keV), and
 U-235 (163.36 keV) : Pu-239 (161.48 keV).

Excess Am-241 set:

Pu-240 (160.31 keV) : Pu-239 (129.30 keV),
 Pu-241 (148.54 keV) : Pu-239 (129.30 keV),
 Am-241 (125.29 keV) : Pu-239 (129.30 keV),
 U-235 (185.72 keV) : Pu-239 (129.30 keV), and
 U-235 (185.72 keV) : Pu-239 (203.55 keV).

Extras for testing purposes:

Pu-238 (152.72 keV) : Pu-239 (144.20 keV),
 Pu-238 (152.72 keV) : Pu-239 (129.30 keV),
 Am-241 (165.92 keV) : U-235 (163.36 keV),
 Am-241 (169.56 keV) : U-235 (163.36 keV),
 Am-241 (169.56 keV) : U-235 (185.72 keV), and
 Am-241 (125.29 keV) : U-235 (185.72 keV).

```
character*4 r_id_1(10,4)/
```

```
. 'PU', 'PU', 'AM', 'PU', 6* ' ', !Standard set of ratios.
. 'PU', 'PU', 'AM', 'U', 6* ' ', !Excess U-235 set.
. 'PU', 'PU', 'AM', 'U', 'U', 5* ' ', !Excess Am-241 set.
. 'PU', 'PU', 'AM', 'AM', 'AM', 'AM', 4* ' ', !Extras.
. r_id_2(10,4)/
. 'PU', 'PU', 'PU', 'PU', 6* ' ', !Standard set of ratios.
```

```

. 'PU', 'PU', 'PU', 'PU', 6* ' ', !Excess U-235 set.
. 'PU', 'PU', 'PU', 'PU', 'PU', 5* ' ', !Excess Am-241 set.
. 'PU', 'PU', 'U', 'U', 'U', 'U', 4* ' ', !Extras.
. r_id_3 !A spare for use as needed.
real*4 r_en_1(10,4)/
. 152.72,148.54,146.55,160.31, 6*0.0, !Standard set of ratios.
. 160.31,148.54,125.29,163.36, 6*0.0, !Excess U-235 set.
. 160.31,148.54,125.29,185.72,185.72, 5*0.0, !Excess Am-241 set.
. 152.72,152.72,165.92,169.56,169.56,125.29, 4*0.0/, !Extras.
. r_en_2(10,4)/
. 148.54,144.20,144.20,161.48, 6*0.0, !Standard set of ratios.
. 161.48,161.48,129.30,161.48, 6*0.0, !Excess U-235 set.
. 129.30,129.30,129.30,129.30,203.55, 5*0.0, !Excess Am-241 set.
. 144.20,129.30,163.36,163.36,185.72,185.72, 4*0.0/ !Extras.
integer*4 r_z_1(10,4)/
. 94, 94, 95, 94, 6*0, !Standard set of ratios.
. 94, 94, 95, 92, 6*0, !Excess U-235 set.
. 94, 94, 95, 92, 92, 5*0, !Excess Am-241 set.
. 94, 94, 95, 95, 95, 95, 4*0/, !Extras.
. r_z_2(10,4)/
. 94, 94, 94, 94, 6*0, !Standard set of ratios.
. 94, 94, 94, 94, 6*0, !Excess U-235 set.
. 94, 94, 94, 94, 94, 5*0, !Excess Am-241 set.
. 94, 94, 92, 92, 92, 92, 4*0/, !Extras.
. r_a_1(10,4)/
. 238, 241, 241, 240, 6*0, !Standard set of ratios.
. 240, 241, 241, 235, 6*0, !Excess U-235 set.
. 240, 241, 241, 235, 235, 5*0, !Excess Am-241 set.
. 238, 238, 241, 241, 241, 241, 4*0/, !Extras.
. r_a_2(10,4)/
. 241, 239, 239, 239, 6*0, !Standard set of ratios.
. 239, 239, 239, 239, 6*0, !Excess U-235 set.
. 239, 239, 239, 239, 239, 5*0, !Excess Am-241 set.
. 239, 239, 235, 235, 235, 235, 4*0/ !Extras.

```

```

C logical iftest, !Can the lead-liner or excessive-uranium test be performed?
. ifmtch !Was an energy pair match found?

```

```

C Declaration related to weighted averages with outlier testing via WAVG_O.
C

```

```

real*4 values(6),errors(6), !Data points passed to WAVG_O.
. mean,stdev,wmean,wstdev,rchisq !Returned from WAVG_O.
integer*4 ndatum !Number of data points passed to WAVG_O.
integer*2 indexes(6) !To keep track of I indexes of passed values.
logical ifvald(6) !Flag whether a value was used by WAVG_O.

```

```

C
C For L. V. East's test version request . . .

```



```

C      real*4 r_pi_1(10,4),r_pe_1(10,4),    !Pre-interference microCuries decontamination ratio numerators . .
.
. r_pi_2(10,4),r_pe_2(10,4),    ! . . . and denominator.
. r_xi_1(10,4),r_xe_1(10,4),r_xi_2(10,4),r_xe_2(10,4),    !Pre-XT GPS . . .
. r_ai_1(10,4),r_ae_1(10,4),r_ai_2(10,4),r_ae_2(10,4),    !After-interference GPS . . .
. r_bi_1(10,4),r_bi_2(10,4)    !After-interference microCuries. . .

```

```

C      Arrays for isotopic ratios (nuclide 1:nuclide 2) to be calculated:

```

```

C      Standard set:

```

```

C      Pu-241 (148.54 keV) : Pu-239 (144.20 keV),
C      Am-241 (146.55 keV) : Pu-239 (144.20 keV) and
C      (125.29 keV) : (129.30 keV),
C      Pu-240 (160.31 keV) : Pu-239 (161.48 keV),
C      Pu-238 (152.72 keV) : Pu-239 (144.20 keV),
C      U-235 (185.72 keV) : Pu-239 (203.55 keV) and
C      (185.72 keV) : (129.30 keV) and
C      (163.36 keV) : (161.48 keV), and
C      Am-241 (169.56 keV) : U-235 (163.36 keV).

```

```

C      Excess U-235 set (i.e., when U-235:Pu-239 > 4.4%):

```

```

C      Pu-241 (148.54 keV) : Pu-239 (161.48 keV) and
C      (148.54 keV) : (129.30 keV),
C      Am-241 (125.29 keV) : Pu-239 (129.30 keV),
C      Pu-240 (160.31 keV) : Pu-239 (161.48 keV) and
C      (160.31 keV) : (129.30 keV),
C      Pu-238 (152.72 keV) : Pu-239 (129.30 keV),
C      U-235 (185.72 keV) : Pu-239 (203.55 keV) and
C      (185.72 keV) : (129.30 keV) and
C      (163.36 keV) : (161.48 keV), and
C      Am-241 (125.29 keV) : U-235 (185.72 keV) and
C      (169.56 keV) : (185.72 keV).

```

```

C      Excess Am-241 set (i.e., when Am-241:Pu-239 > 1.3%):

```

```

C      Pu-241 (148.54 keV) : Pu-239 (129.30 keV),
C      Am-241 (125.29 keV) : Pu-239 (129.30 keV),
C      Pu-240 (160.31 keV) : Pu-239 (129.30 keV),
C      Pu-238 (152.72 keV) : Pu-239 (129.30 keV),
C      U-235 (185.72 keV) : Pu-239 (203.55 keV) and
C      (185.72 keV) : (129.30 keV), and
C      Am-241 (125.29 keV) : U-235 (185.72 keV) and
C      (169.56 keV) : (185.72 keV).

```

```

C      The RATID<n> arrays are packed as follows: for multiple line pairs for
C      the same ratio, subsequent elements are left blank, this being used as
C      a flag for when to start reaccumulating terms for a weighted average.

```

```

C      character*4 ratid1(20,3)/

```

```

.      'PU', 'AM', ' ', 'PU', 'PU', 'U', ' ', ' ', 'AM',

```

```

. 11* ' ', !Standard set of ratios.
. 'PU', ' ', 'AM', 'PU', ' ', 'PU', 'U', ' ', ' ',
. 'AM', ' ', 9* ' ', !Excess U-235 set.
. 'PU', 'AM', 'PU', 'PU', 'U', ' ', 'AM', ' ', 12* ' ', !Excess Am-241 set.
. ratid2(20,3)/
. 'PU', 'PU', ' ', 'PU', 'PU', 'PU', ' ', ' ', 'U',
. 11* ' ', !Standard set of ratios.
. 'PU', ' ', 'PU', 'PU', ' ', 'PU', 'PU', ' ', ' ',
. 'U', ' ', 9* ' ', !Excess U-235 set.
. 'PU', 'PU', 'PU', 'PU', 'PU', ' ', 'U', ' ', 12* ' ', !Excess Am-241 set.
. ratid3 !A spare for use as needed.

```

The RATEN<n> arrays are packed as follows: non-zero values indicate that a ratio is to be calculated.

```

real*4 raten1(20,3)/
. 148.54,146.55,125.29,160.31,152.72,185.72,185.72,163.36,169.56,
. 11*0.0, !Standard set of ratios.
. 148.54,148.54,125.29,160.31,160.31,152.72,185.72,185.72,163.36,
. 125.29,169.56, 9*0.0, !Excess U-235 set.
. 148.54,125.29,160.31,152.72,185.72,185.72,125.29,169.56,12*0.0/, !Excess Am-241 set.
. raten2(20,3)/
. 144.20,144.20,129.30,161.48,144.20,203.55,129.30,161.48,163.36,
. 11*0.0, !Standard set of ratios.
. 161.48,129.30,129.30,161.48,129.30,129.30,203.55,129.30,161.48,
. 185.72,185.72, 9*0.0, !Excess U-235 set.
. 129.30,129.30,129.30,129.30,203.55,129.30,185.72,185.72,12*0.0/ !Excess Am-241 set.

```

The RATZ<n> arrays are packed as follows: for multiple line pairs for the same ratio, subsequent elements are set to zero.

```

integer*4 ratz1(20,3)/
. 94, 95, 0, 94, 94, 92, 0, 0, 95,
. 11*0, !Standard set of ratios.
. 94, 0, 95, 94, 0, 94, 92, 0, 0,
. 95, 0, 9*0, !Excess U-235 set.
. 94, 95, 94, 94, 92, 0, 95, 0, 12*0/, !Excess Am-241 set.
. ratz2(20,3)/
. 94, 94, 0, 94, 94, 94, 0, 0, 92,
. 11*0, !Standard set of ratios.
. 94, 0, 94, 94, 0, 94, 94, 0, 0,
. 92, 0, 9*0, !Excess U-235 set.
. 94, 94, 94, 94, 94, 0, 92, 0, 12*0/, !Excess Am-241 set.

```

The RATA<n> arrays are packed as follows: for multiple line pairs for the same ratio, subsequent elements are set to zero.

```

. rata1(20,3)/
.   241, 241, 0, 240, 238, 235, 0, 0, 241,
. 11*0, !Standard set of ratios.
.   241, 0, 241, 240, 0, 238, 235, 0, 0,
.   241, 0, 9*0, !Excess U-235 set.
.   241, 241, 240, 238, 235, 0, 241, 0, 12*0/, !Excess Am-241 set.
. rata2(20,3)/
.   239, 239, 0, 239, 239, 239, 0, 0, 235,
. 11*0, !Standard set of ratios.
.   239, 0, 239, 239, 0, 239, 239, 0, 0,
.   235, 0, 9*0, !Excess U-235 set.
.   239, 239, 239, 239, 239, 0, 235, 0, 12*0/ !Excess Am-241 set.

```

C
C For determining, in cases of interference decontamination, the clean line
C used - this is needed for error propagation.

C
C include 'gap\$src:intlib.inc/list'

C
C For obtaining the uncertainties in branching ratios via the function BR_UC.

C
C include 'branch.inc'
C common /bruc/ nob,bru
C open(1,name='ge\$isotope\$lib:swepp.br_uc',type='old',readonly,
C . shared,form='unformatted')
C read(1,nob,(bru.e(i),i=1,nob)
C close(1,status='keep')

C
C Format statements.

C
8000 FORMAT('1',///,' ',13('*'),
1 'SWEPP BARREL MONITOR GAMMA-RAY SPECTROMETERY SUMMARY REPORT ',
2 'V 01.00 - ',a11,' - GENERATED ',a9,13('*'))
8010 FORMAT(///,' SAMPLE NAME: ',A100)
8020 FORMAT(/,' SAMPLE ID: ',A40,29(' '), 'ANALYST:',32('_'))
8030 FORMAT(/,' SAMPLE COLLECTION (DATE/TIME):',1(' '),
1 I6,'/',A4,38(' '), 'APPROVED BY: ',27('_'))
8040 FORMAT(/,' SAMPLE COUNTED (DATE/TIME):',4(' '),I6,
1 '/',A4,38(' '), 'COMMENTS: ',30('_'))
8050 FORMAT(/,' SAMPLE QUANTITY:',1PE12.5,' QUANTITY UNITS: ',
1 A10,34(' '),30('_'),///,' ATTENUATION TABLE USED: ',a8,
2 58(' '),30('_')//91x,30('_'))
8100 FORMAT(///,' RADIONUCLIDE',4(' '), 'SAMPLE ACTIVITY (',
1 A3,'/',A<IXU>,')',6(' '), '% ERROR (1-SIGMA)')
8101 FORMAT(///,
2 ' ***** NO PHOTOPEAKS TAGGED IN THIS ANALYSIS *****')
8110 FORMAT(I5,A4,I3,10X,1PE11.3,12X,0PF8.2)
9005 FORMAT(50X,I8,' TAGS REMOVED--NO ICODE 1 LINE')

```

9200 format(//10X,'ISOTOPIC MASS RATIOS (%)'/10x,'NONE CAN BE ',
. 'CALCULATED - A LEAD LINER IS LIKELY PRESENT')
9202 format(//10X,'ISOTOPIC MASS RATIOS (%)'/10x,'(ONLY A ',
. 'RESTRICTED SET CAN BE CALCULATED - EXCESSIVE URANIUM IS ',
. 'LIKELY PRESENT)')
9204 format(//10X,'ISOTOPIC MASS RATIOS (%)'/10x,'(THE STANDARD ',
. 'SET CAN BE CALCULATED)')
9206 format(//10X,'ISOTOPIC MASS RATIOS (%)'/10x,'(ONLY A ',
. 'RESTRICTED SET CAN BE CALCULATED - EXCESSIVE AMERICIUM IS ',
. 'LIKELY PRESENT)')
9208 format(//10X,'ISOTOPIC MASS RATIOS (%)'/10x,'(THIS IS A TEST ',
. 'SET)')
9210 format(2x,a2,'-',i3,' (',f6.2,' keV) to ',a2,'-',i3,' (',f6.2,
. ' keV): ',a24/47x,a85/10x,'x,t corrected numerator gps= ',e10.4,
. ' error=',e10.4/12x,'exp(mu*x)=' ,e10.4,' exp(smu*t)=' ,e10.4,
. ' INTCOR subtraction=' ,e10.4/12x,
. ' uncorrected numerator gps= ',e10.4,' error=' ,e10.4/10x,
. ' x,t corrected denominator gps=' ,e10.4,' error=' ,e10.4/12x,
. ' exp(mu*x)=' ,e10.4,' exp(smu*t)=' ,e10.4,
. ' INTCOR subtraction=' ,e10.4/
. 12x,'uncorrected denominator gps=' ,e10.4,' error=' ,e10.4/)
9220 format(2x,a2,'-',i3,' (',f6.2,' keV) to ',a2,'-',i3,' (',f6.2,
. ' keV): ',a24/47x,a85)
9222 format(2x,a2,'-',i3,' (',f6.2,' keV) to ',a2,'-',i3,' (',f6.2,
. ' keV): ',a24/47x,a85/) !Single value version.
9224 format(2x,a2,'-',i3,' (wgthd aver) to ',a2,'-',i3,' (wgthd ',
. ' aver): ',a24/) !Weighted average version.
9226 format(2x,' Because of poor precision (>20%), will try a more ',
. ' restrictive set of energy pairs for the above ratio . . .')
9228 format(2x,' . . . there is not a more restrictive set of energy ',
. ' pairs to try for this ratio.')
9230 format(2x,' . . . returning to the less restrictive sets of ',
. ' energy pairs for the remaining ratios.')
9299 FORMAT('1',/)
9300 FORMAT (' THE MAXIMUM NUMBER
1 OF ISOTOPIC TAGS IS GREATER THAN 50. ',/,
1 ' I WILL SAVE THE FIRST 50 VALUES')
9301 FORMAT(// ' FOR NUCLIDE ',a2,'-',i3,' :',/,
1 6X,'X-EFFECTIVE = 0 CM (I.E., NO CORRECTION)',/,
2 6X,'T-EFFECTIVE = 0 CM (I.E., NO CORRECTION)',/,
3 ' BECAUSE OF THE NON-TRUE-POSITIVE LINE AREA NOTED ABOVE')
9302 FORMAT(// ' FOR NUCLIDE ',a2,'-',i3,' :',/,
1 6X,'X-EFFECTIVE = ',a24,' CM',/,
2 6X,'T-EFFECTIVE = ',a24,' CM')
9303 FORMAT(6X,'OBSERVED GAMMAS-PER-SECOND(' ,0PF8.2,' Kev)=' ,1PE11.4,
2 ' ,', 'MATRIX MU=' ,1PE10.3,' CM-1, SELF MU=' ,1PE10.3,' CM-1')
9304 FORMAT(6X,'AVERAGE STANDARD DEVIATION OF OBSERVED GAMMAS-PER-',

```

```

4 'SECOND FROM THE PREDICTED VALUE = ',1PE9.2)
9305 FORMAT(// ' FOR NUCLIDE ',a2,'-',i3,':',/,
1 6X,'X-EFFECTIVE = ',a24,' CM',/,
1 2(6X,'OBSERVED GAMMAS-PER-SECOND(',0PF8.2,' Kev)=',1PE11.4,', ',
2 'MATRIX MU=',1PE10.3,' CM-1, SELF MU=',1PE10.3,' CM-1'/),6X,
3 'AVERAGE PERCENT ABSOLUTE DEVIATION OF CORRECTED GAMMAS-PER-',
4 'SECOND FROM MEAN = ',1PE9.2)
9310 FORMAT(1X,10('*'),/,
1 ' THE NUMBER OF PHOTOPEAKS EXCEED 200. ONLY THE RESULTS FROM THE
2 FIRST 200 WILL BE SAVED',/,1X,10('*'))
9314 FORMAT(/,3X,'PU-239(161 Kev) WAS NOT FOUND DURING THE ANALYSIS')
9316 FORMAT(/,3X,'PU-239 MASS ASSUMING "POINT SOURCE": ',a24,' Grams')
9318 FORMAT(/,3X,'PU-239 MASS ASSUMING "DISTRIBUTED SOURCE": ',
1 F8.1,' Grams')
9319 FORMAT(/,3X,
1 ' PU-239 (LINE 413 Kev) WAS NOT FOUND DURING THE ANALYSIS')
9500 FORMAT(//,3X,' COMPUTED AREA FOR LINE AT: ',F6.1,
1 ' IS NOT TRUE-POSITIVE'//3X,' X- AND T-EFFECTIVE FOR ',
2 a2,'-',i3,' WILL BE SET TO ZERO')
9501 FORMAT(//,3X,' COMPUTED AREA FOR LINE AT: ',F6.1,
1 ' IS NOT TRUE-POSITIVE'//3X,' X-EFFECTIVE FOR ',
2 a2,'-',i3,' WILL BE SET TO ZERO')
9502 FORMAT(//,3X,' LINE @ ENERGY: ',F6.1,' WAS NOT FOUND'//
1 3X,' X- AND T-EFFECTIVE FOR ',
2 a2,'-',i3,' WILL BE SET TO ZERO')
9503 FORMAT(//,3X,' CANNOT SOLVE FOR X- AND T-EFFECTIVE (DETERMINANT ',
1 'EQUALS ZERO)'//3X,' X- AND T-EFFECTIVE FOR ',
2 a2,'-',i3,' WILL BE SET TO ZERO')

```

C

C Beginning of the routine.

C

C Get the current date for use in generating output.

C

call date(curdatt)

C

C Obtain whether printed output should be generated and the spectrum id.

C

```

IL.ADRRESS=%LOC(cifprt)
IL.ITEM=LNM$_STRING
IL.BL=30
IL.RET=%LOC(lifprt)
STATUS=SYSTRNLNM('LNM$PROCESS_TABLE','IF$PRINT',,IL)
IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))
IL.ADRRESS=%LOC(SP_NAME)
IL.ITEM=LNM$_STRING
IL.BL=30
IL.RET=%LOC(SP_NAME_LN)

```

```

        STATUS=SYS$TRNLNM(, 'LNM$PROCESS_TABLE', 'A1$SPECTRUM$ID', , IL)
        IF(.NOT. STATUS)CALL LIB$STOP(%VAL(STATUS))
C
C OPEN AN READ IN ANALYSIS PARAMETER FILE
150   DS=SP_NAME(1:2)
        PFILE(10:11)=DS
        OPEN(UNIT=1, NAME=PFILE, TYPE='OLD', FORM='UNFORMATTED', READONLY,
        1 SHARED)
        READ(1)AP
        CLOSE(UNIT=1)
C
C OPEN LIBRARY FILES
C ISOTOPE LIBRARY LUN=2
C
        J=INDEX(AP.ISOTOPE_LIB, ' ')-1
        HEADER='GE$ISOTOPE$LIB: '//AP.ISOTOPE_LIB(1:J)//'.LIB'
        OPEN(UNIT=2, NAME=HEADER(1:J+19),
        1 IOSTAT=IOS, READONLY,
        1 TYPE='OLD', ACCESS='DIRECT', RECORDSIZE=128)
        IF(IOS.NE.0)THEN
            ERR_MESS='UNABLE TO OPEN NUCLIDE LIBRARY: '//
            1 AP.ISOTOPE_LIB(1:J)
C
C PRINT ERROR MESSAGE AND EXIT
200   CALL PR_EXIT(ERR_MESS, ' ', ' ')
        ENDIF
C
C
C
C OPEN REQUIRED ISOTOPE LIBRARY
        J=INDEX(AP.REQUIRED_LIB, ' ')-1
        HEADER='GE$LIMIT$LIB: '//AP.REQUIRED_LIB(1:J)//'.LIB'
        OPEN(UNIT=3, NAME=HEADER(1:J+17),
        1 IOSTAT=IOS, READONLY,
        1 TYPE='OLD', ACCESS='DIRECT', RECORDSIZE=128)
        IF(IOS.NE.0)THEN
            ERR_MESS='UNABLE TO OPEN REQUIRED ENERGY LIBRARY '
            1 //HEADER(1:J+13)
            CALL PR_EXIT(ERR_MESS, ' ', ' ')
        ENDIF
C
C GET THREE MOST RESENT BACKGROUND SPECTRA IN THE BACKGROUND DIRECTORY
C
        IF(.NOT.AP.BACK_CORRECT)GOTO 600      !BACKGROUND CORRECT?
        CALL FIND_BKG_FILES(DS, 'BK', 2)
        IF(BK.NO_BKG.EQ.0)GOTO 600
        DO J=1, BK.NO_BKG

```

```

        CALL OPEN$SP(BK.SP_ID(J), IERR)
        IF(IERR) THEN
            ERR_MESS='UNABLE TO OPEN BACKGROUND FILE '
1 //BK.SP_ID(J)
            CALL PR_EXIT(ERR_MESS, ' ', ' ')
        ENDIF
        CALL BATCH_CAL_PULSER(BK.SP_ID(J))
        CALL LIB$GET_LUN(ILUN)
        BF.E(J).LUN=ILUN
        BF.E(J).FW1=VHEAD(ZW)
        BF.E(J).FW2=VHEAD(LW)
        BF.E(J).EZ=VHEAD(CZ)
        BF.E(J).EA=VHEAD(AE)
        BF.E(J).EB=VHEAD(BE)
        BF.E(J).EC=VHEAD(CE)
        BF.E(J).LIVE=VHEAD(LVT)
        BF.E(J).REAL=VHEAD(RLT)
        BF.E(J).SCB=VHEAD(SCB)
        BF.E(J).SCA=VHEAD(SCA)
        BF.E(J).SCC=P.SC_SUM
        BF.E(J).RATE=P.PULSER_RATE
        BF.E(J).CAL_PULSER_WD=P.CAL_PULSER_WD
        BF.E(J).LOW_PULSER_WD=P.LOW_PULSER_WD
        IF(P.NO_PULSER.NE.2) THEN
            BF.E(J).PULSER=.FALSE.
        ELSE
            BF.E(J).PULSER=.TRUE.
        ENDIF
        CLOSE(UNIT=1)
        OPEN(UNIT=ILUN, NAME=BK.SP_ID(J),
1 TYPE='OLD', ACCESS='DIRECT', RECORDSIZE=128, READONLY, SHARED)
        ENDDO
C
600 SP_FILE=' '
    SP_FILE=AP.SAVE_DIR(1:(INDEX(AP.SAVE_DIR, ' ') - 1))//': '
    1 SP_NAME//'.DGE'
    CALL OPEN$SP(SP_FILE, IERR)
    IF(IERR) THEN
        ERR_MESS='UNABLE TO OPEN SPECTRUM '//SP_FILE
        CALL PR_EXIT(ERR_MESS, ' ', ' ')
    ENDIF
C
650 CF =1.0                !UCI CONVERSION
    ACU = 'uCi'
C
C LOAD VALUES INTO VHEAD ARRAY
100 VHEAD(SSN)=8.0          !SEARCH SENSITIVITY

```

```

      VHEAD(NP)=0.0          !NUMBER OF PEAKS FOUND
      VHEAD(HFW)=0.0        !ELIMINATE TAGS WITH EXCESSIVE HALF LIFES
      NEFF=VHEAD(ETB)       !EFFICIENCY TABLE NUMBER
C
C OPEN FILE TO CONTAIN THE PRINTED OUTPUT
      if(cifprt.eq.'N')then  !Opened as a scratch file so it is deleted when execution completes.
        open(6,file='gidxc_temp.output',status='scratch')
        else                !So that the output is printed.
          CALL SPOOL
        endif
C BEGIN PROCESSING DATA
      CALL BATCH_CAL_PULSER(SP_FILE)
C
C IF PULSERS ARE IN SYSTEM SET THE NUMBER OF CHANNELS TO ANALYZE TO
C NUMBER OF CHANNEL - 2*PLS_CHANNELS
      IF(P.NO_PULSER.NE.0)THEN
        VHEAD(NC)=P.NO_CHANNELS-2*P.PLS_CHANNELS
      ENDIF
      CALL NOR_SEARCH      !FIND PEAKS
C
C OPEN AND READ THE FILE CONTAINING THE PEAKS TO INSERT
      NPF=VHEAD(NP)        !NUMBER OF PEAKS FOUND BY SEARCH

      CALL LIB$GET_LUN(IUN)
      OPEN(UNIT=IUN,NAME='GE$LIMIT$LIB:INSERT.LIB',READONLY,SHARED,
        1 TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,IOSTAT=IOS)

      IF(IOS .NE. 0 )THEN  !IF FILE IS NOT THERE DONT INSERT PEAKS
        CALL LIB$FREE_LUN(IUN)
      ELSE
        READ(IUN'1)XIN      !GET ENERGIES

        CLOSE(UNIT=IUN)
        CALL LIB$FREE_LUN(IUN)

        T1=VHEAD(AE)
        T2=VHEAD(BE)
        T3=VHEAD(CE)

        IX=0

        DO I=1,128  !CONVERT ENERGIES TO CHANNEL NUMBERS
          IF(XIN(i) .EQ. 0.0)GOTO 2000

```



```

        IF(T3 .NE. 0.0) THEN
            T4=T1-XIN(i)
            T5=DSQRT(T2*T2 -4.0*T3*T4)
            IX=IX+1
            XC(IX)=(-T2+T5)/(2.0*T3)
            XC(IX)=XC(IX) - VHEAD(CZ) !ADJUST FOR ZERO OFFSET
        ELSE
            IX=IX+1
            XC(IX)=((XIN(i)-T1)/T2) - VHEAD(CZ)
        ENDIF
    ENDDO
2000    IF(IX .EQ. 0) GOTO 2900 !NO PEAKS TO INSERT
C
C FIND THE CHANNEL WITH THE MAXIMUM NUMBER OF COUNTS IN THE RANGE
C OF +- 1 CHANNELS
        DO I=1, IX
C
            IF((XIN(i) .EQ. 160.28) .OR. (XIN(i) .EQ. 161.45)) GOTO 2050
            K=NINT(XC(i))-1
            CMAX=0.0
            DO J=K, K+2
                IF(FGET(J) .GT. CMAX) THEN
                    IM=J
                    CMAX=FGET(J)
                ENDIF
            ENDDO
            XC(i)=IM
2050    ENDDO

C PEAKS TO BE INSERTED HAVE BEEN DETERMINED ELIMINATE ALL PEAKS FOUND WHICH
C ARE WITHIN 1 WIDTH OF THE PEAKS TO BE INSERTED

2100    DO I=1, NPF
        DO J=1, IX
C
            Was * 1.0 in DXC . . .
            IF(ABS(PKD(I,1,XV)-XC(J)) .LT. (F_WIDTH(XC(J))*0.75)) THEN
                NPF=NPF-1
                IF(NPF .EQ. 0) THEN
                    NPF=IX
                    DO K=1, IX
                        PKD(K,1,XV)=XC(K)
                    ENDDO
                ENDIF
            ENDIF
        ENDDO
    ENDDO

```

```

                                GOTO 2900
                            ENDIF

                            DO K=I,NPF
                                PKD(K,1,XV)=PKD(K+1,1,XV)
                            ENDDO
                            GOTO 2100

                        ENDIF
                    ENDDO
                ENDDO
            C
            C ALL PEAKS FOUND HAVE BEEN ELIMINATED IF THEY ARE WITH IN ONE WIDTH OF THE
            C INSERT PEAKS. INSERT THE PEAKS FROM THE ENERGY LIST

                DO I=1,IX
                    IF(NPF .EQ. 300)GOTO 2900
                    NPF=NPF+1
                    PKD(NPF,1,XV)=XC(i)
                ENDDO

            C
            C SORT REMAINING PEAK LOCATIONS IN ASSENDING ORDER

                IF(NPF .EQ. 1)GOTO 2900

2200                DO I=1,NPF-1
                    IF(PKD(I,1,XV) .GT. PKD(I+1,1,XV))THEN
                        TXV=PKD(I,1,XV)
                        PKD(I,1,XV)=PKD(I+1,1,XV)
                        PKD(I+1,1,XV)=TXV
                        GOTO 2200
                    ENDIF
                ENDDO

            ENDIF
        C
        2900 VHEAD(NC)=P.NO_CHANNELS
            IF(VHEAD(NP).EQ.0.0)THEN !ANY PEAKS FOUND?
                READ(3'1)DATA !NO IS THERE ANY LIMIT LINES?
                IF(DATA(1).EQ.0.0)THEN !YES THEN DO LIMIT
                    CALL CBAT_PRGIDXC(AP.BACK_CORRECT,AP.ISOTOPE_LIB,
1                    AP.REQUIRED_LIB,AP.INTERFERENCE_LIB,AP.PRINT_RESULTS)
                    CALL EXIT
                ELSE
                    GOTO 700
                ENDIF

```

```

ENDIF
VHEAD(NP)=NPF
CALL DPY_CBAT_GAUSS
700 CALL DPY_CBAT_LINFIT
CLOSE(UNIT=3) !CLOSE LIMIT LIBRARY FILE
CALL LIBRARY_TAG !LIBRARY LOOKUP ASSIGN TAGS

C
C SUBTRACT BACKGROUND?
IF(AP.BACK_CORRECT)CALL SUB_BKG
C
NPS=VHEAD(NP)

C
C GET DATA FROM HEADER BLOCK
C
READ(1:1)HEADER !READ HEADER BLOCK
DO I = 61, 160 !CHECK FOR END BYTE OF TITLE
IF (ICHAR(HEADER(I:I)) .EQ. 0) GOTO 50
ENDDO
I = 160
50 SDATE=VHEAD(DSD) !SAMPLE DATE
STIME=VHEAD(TSD) !SAMPLE TIME
WRITE(UNIT=STM(1:4),FMT='(I4)')STIME
C ELIMINATE LEADING SPACES
IF(STM(1:3).EQ.' ')STM(1:3)='000'
IF(STM(1:2).EQ.' ')STM(1:2)='00'
IF(STM(1:1).EQ.' ')STM(1:1)='0'
CDATE=VHEAD(DSC) !COUNT DATE
CTIME=VHEAD(TSC) !COUNT TIME
WRITE(UNIT=CTM(1:4),FMT='(I4)')CTIME
C ELIMINATE LEADING SPACES
IF(CTM(1:3).EQ.' ')CTM(1:3)='000'
IF(CTM(1:2).EQ.' ')CTM(1:2)='00'
IF(CTM(1:1).EQ.' ')CTM(1:1)='0'
C
C BEGIN PRINTING OF SUMMARY SHEETS
C
WRITE(6,8000)'MASS RATIOS',curdat

WRITE(6,8010)HEADER(61:160)

WRITE(6,8020)HEADER(1:40)

WRITE(6,8030)SDATE,STM

WRITE(6,8040)CDATE,CTM

```

```

C
C Pre-XT LVE test values.
C
      do 103 j=1,4
        do i=1,10
          if(r_a_1(i,j).eq.0)goto 103
          call find_line(r_z_1(i,j),r_a_1(i,j),r_id_1(i,j),
            r_en_1(i,j))    !Line #1.
          r_xi_1(i,j)=fl.cps/(fl.br*fl.eff/100.0)
          r_xe_1(i,j)=(fl.err*(fl.cps/fl.lmc)/fl.br)/    !Remove the FIND_LINE-added branching ratio component.
            (fl.br*fl.eff/100.0)
          if(r_a_2(i,j).eq.0)goto 103
          call find_line(r_z_2(i,j),r_a_2(i,j),r_id_2(i,j),
            r_en_2(i,j))    !Line #2.
          r_xi_2(i,j)=fl.cps/(fl.br*fl.eff/100.0)
          r_xe_2(i,j)=(fl.err*(fl.cps/fl.lmc)/fl.br)/    !Remove the FIND_LINE-added branching ratio component.
            (fl.br*fl.eff/100.0)
        enddo
      103 enddo

```

[illegible]

```
C      gps* = gps(i) * exp(x-efc * matrix-mu(i)) * exp(t-efc * self-mu(i))
```

```
C
do i=1,10
  if(req_eng(1,i).ne.0.0)numefc=i      !Number of sets of X- and T-effective to be calculated.
enddo
do 6100 iefc=1,numefc      !For all NUMEFC sets of X- and T-effective.
  do i=1,10      !Find JLAST, i.e., the number of lines to be used.
    if(req_eng(i,iefc).ne.0.0)jlast=i
  enddo
enddo
```

```

C ARE THE REQUIRED LINES PRESENT (AND TRUE POSITIVE) IN THE SAMPLE
DO 5100 J=1,jlast
  CS(J,iefc)=0.0
  DO I=1,NPS
    IF(PKD(I,2,NOT) .EQ. 0.0)GOTO 5000
    itag=0    !Find the proper tag.
    do ii=1,pkd(i,2,not)
      r=pkd(i,ii,isa)
      if(((pkd(i,ii,iaa)/10.0).eq.dxc_a(1,iefc)).and.
        (pkd(i,ii,iza).eq.dxc_z(1,iefc)).and.
        (sid.eq.dxc_id(1,iefc)))
        itag=ii    !Match found!
      enddo
    IF((ABS(PKD(I,3,ERA)-REQ_ENG(J,iefc)) .LE. 0.5).and.
      (itag.ne.0))THEN
      IF(.NOT. TRUE_POSITIVE(PKD(I,1,AV),PKD(I,2,DA),2.0))THEN
        if(jlast.ge.3)then
          WRITE(6,9500)REQ_ENG(J,iefc),dxc_id(1,iefc)(1:2),
1          dxc_a(1,iefc)
          else
1          WRITE(6,9501)REQ_ENG(J,iefc),dxc_id(1,iefc)(1:2),
            dxc_a(1,iefc)
          endif
          x_efc(iefc)=0.0
          t_efc(iefc)=0.0
          goto 6100    !No correction to be applied.
        ENDIF

        CALL EFF(NEFF,REQ_ENG(J,iefc),EFC)
        BR(J,iefc)=PKD(I,itag,BRA)/100.0
        KV(J,iefc)=BR(J,iefc)*EFC
        CALL MU_LOOK(REQ_ENG(J,iefc),S(J,iefc))    !For external matrix attenuation.
        if(jlast.ge.3)CALL MU_self_LOOK(dxc_id(1,iefc),
          REQ_ENG(J,iefc),PV(J,iefc))    !For self attenuation.
        CS(J,iefc)=PKD(I,2,CPS)
        goto 5100    !No need to look any further.
      ENDIF
    ENDDO
5000  ENDDO
5100  ENDDO
C
C Did we find all the necessary lines?  If not, print an appropriate error
C message and set x- and t-effective to zero.
C
  DO J=1,jlast
    IF(CS(J,iefc) .EQ. 0.0)THEN
      x_efc(iefc)=0.0
      t_efc(iefc)=0.0

```

```

        WRITE(6,9502)REQ_ENG(J,iefc),dxc_id(1,iefc)(1:2),
1          dxc_a(1,iefc)
          goto 6100
      ENDIF
  ENDDO

C
C AT THIS POINT ALL LINES REQUIRED FOR MATRIX CORRECTION ARE PRESENT AND
C TRUE POSITIVE
C
csim      if(jlast.eq.3)then    !X- and T-effective are to be calculated.
C
C      NOW SOLVE FOR T & X EFFECTIVE
C
C      Solving for x- and t-effective is based on simultaneously solving three
C      equations of the form  $gps^* = gps(i) * \exp(x-efc * matrix-mu(i)) * \exp(t-efc * self-mu(i))$ , where  $gps^*$  is the corrected activity and is to be
C      the same for all three lines,  $gps(i)$  is the uncorrected activity for line
C      i,  $x-efc$  is the matrix attenuation-related correction factor,  $matrix-mu(i)$ 
C      is the matrix attenuation at the energy of line i,  $t-efc$  is the self
C      attenuation-related correction factor, and  $self-mu(i)$  is the self
C      attenuation at the energy of line i.
C
C      To solve these simultaneous equations using determinants, the first step
C      is to linearize them, in this case by taking the logarithm of both sides
C      of the equation, to give  $\ln(gps^*) = \ln(gps(i)) + x-efc * matrix-mu(i) + t-efc * self-mu(i)$ . For purposes of forming the proper determinant, this
C      equation needs to be rearranged by swapping the  $gps^*$  and  $gps(i)$  terms to
C      give  $\ln(gps(i)) = \ln(gps^*) - x-efc * matrix-mu(i) - t-efc * self-mu(i)$ . For
C      ease of reading and coding, let  $a = \ln[gps(i)]$  and  $a^* = \ln(gps^*)$ . The
C      principal determinant D, then, is given by
C
C      D =
C      | 1 -matrix-mu(1) -self-mu(1) |
C      | 1 -matrix-mu(2) -self-mu(2) |
C      | 1 -matrix-mu(3) -self-mu(3) |
C
C      = [ matrix-mu(2)*self-mu(3) - matrix-mu(3)*self-mu(2) ]
C      - [ -matrix-mu(1) ] * [ -self-mu(3) + self-mu(2) ]
C      + [ -self-mu(1) ] * [ -matrix-mu(3) + matrix-mu(2) ].
C
C       $\ln(gps^*)$ , or  $a^*$ , is given by the determinant  $D(a^*)$ , obtained by
C      substituting the 1s in D with  $\ln[gps(i)]$ s (or  $a(i)$ s), divided by D, where
C
C      D(a*) =
C      | a(1) -matrix-mu(1) -self-mu(1) |
C      | a(2) -matrix-mu(2) -self-mu(2) |
C      | a(3) -matrix-mu(3) -self-mu(3) |
C
C      = a(1) * [ matrix-mu(2)*self-mu(3) - matrix-mu(3)*self-mu(2) ]

```

```

C      - [ -matrix-mu(1) ] * [ -a(2)*self-mu(3) + a(3)*self-mu(2) ]
C      + [ -self-mu(1) ] * [ -a(2)*matrix-mu(3) + a(3)*matrix-mu(2) ],
C

```

taking care of all of the minus signs appropriately. gps* (GPSTAR) will be calculated at this point for purposes of determining a goodness-of-fit (here to be reflected in the average percent absolute deviation of the corrected gps values from gps*), yet these corrections will also be applied to the individual lines later in this code.

Similarly, x-efc is given by the determinant D(x-efc), obtained by substituting the matrix-mu(i)s in D with ln[gps(i)]s (or a(i)s), divided by D, where

$$\begin{aligned}
 D(x\text{-efc}) &= \begin{vmatrix} 1 & a(1) & -\text{self-mu}(1) \\ 1 & a(2) & -\text{self-mu}(2) \\ 1 & a(3) & -\text{self-mu}(3) \end{vmatrix} \\
 &= \begin{bmatrix} -a(2)*\text{self-mu}(3) + a(3)*\text{self-mu}(2) \\ -a(1) * [-\text{self-mu}(3) + \text{self-mu}(2)] \\ + [-\text{self-mu}(1)] * [a(3) - a(2)] \end{bmatrix}
 \end{aligned}$$

Lastly, t-efc is given by the determinant D(t-efc), obtained by substituting the self-mu(i)s in D with ln[gps(i)]s (or a(i)s), divided by D, where

$$\begin{aligned}
 D(t\text{-efc}) &= \begin{vmatrix} 1 & -\text{matrix-mu}(1) & a(1) \\ 1 & -\text{matrix-mu}(2) & a(2) \\ 1 & -\text{matrix-mu}(3) & a(3) \end{vmatrix} \\
 &= \begin{bmatrix} -\text{matrix-mu}(2)*a(3) + \text{matrix-mu}(3)*a(2) \\ - [-\text{matrix-mu}(1)] * [a(3) - a(2)] \\ + a(1) * [-\text{matrix-mu}(3) + \text{matrix-mu}(2)] \end{bmatrix}
 \end{aligned}$$

Note that for a solution to exist, D cannot equal zero, and so this will be tested for and flagged as appropriate; if D equals zero, both x- and t-efc will be set to zero so that no correction is performed. Also, in the following, D will be coded as DPRINC, while, as noted above, gps (or exp(a)) is contained in CS/KV, matrix-mu is contained in S, and self-mu is contained in PV. gps* is contained in GPSTAR, and the associated average percent absolute deviation of the corrected gps values from gps* is contained in PDCGPS.

```

C
csim      dprinc=(s(2,iefc)*pv(3,iefc)-s(3,iefc)*pv(2,iefc))
csim      .      -(-s(1,iefc))*(-pv(3,iefc)+pv(2,iefc))
csim      .      +(-pv(1,iefc))*(-s(3,iefc)+s(2,iefc))
csim      if(dprinc.eq.0.0)then      !Cannot solve for x- and t-efc.
csim      x_efc(iefc)=0.0

```

```

csim          t_efc(iefc)=0.0
csim          write(6,9503) dxc_id(1,iefc)(1:2), dxc_a(1,iefc)
csim          goto 6100
csim          endif
csim          a(1)=log(cs(1,iefc)/kv(1,iefc))      !Log = ln (log10 = log).
csim          a(2)=log(cs(2,iefc)/kv(2,iefc))
csim          a(3)=log(cs(3,iefc)/kv(3,iefc))
csim          x_efc(iefc)=((-a(2)*pv(3,iefc)+a(3)*pv(2,iefc))
csim          .      -a(1)*(-pv(3,iefc)+pv(2,iefc))+(-pv(1,iefc))
csim          .      *(a(3)-a(2)))/dprinc
csim          t_efc(iefc)=((-s(2,iefc)*a(3)+s(3,iefc)*a(2))
csim          .      -(-s(1,iefc))*(a(3)-a(2))+a(1)*(-s(3,iefc)+s(2,iefc)))
csim          .      /dprinc
csim          gpstar=exp((a(1)*(s(2,iefc)*pv(3,iefc)
csim          .      -s(3,iefc)*pv(2,iefc))-(-s(1,iefc))*(-a(2)*pv(3,iefc)+
csim          .      a(3)*pv(2,iefc))+(-pv(1,iefc))*(-a(2)*s(3,iefc)+
csim          .      a(3)*s(2,iefc)))/dprinc)
csim          pcdgps(iefc)=0.0
csim          do i=1,3
csim          .      pcdgps(iefc)=pcdgps(iefc)+100.0
csim          .      *abs(((cs(i,iefc)/kv(i,iefc))
csim          .      *exp(x_efc(iefc)*s(i,iefc))
csim          .      *exp(t_efc(iefc)*pv(i,iefc)))-gpstar)/gpstar
csim          .      enddo
csim          pcdgps(iefc)=pcdgps(iefc)/3.0      !Take the average of the above summation.
C
C      (E. W. Killian's original, condensed computational scheme.)
C
cewk  F1=LOG(CS(1,iefc)/CS(3,iefc))
cewk  F2=LOG(KV(1,iefc)/KV(3,iefc))
cewk  F3=(S(3,iefc)-S(1,iefc))/(S(2,iefc)-S(1,iefc))
cewk
cewk  F1=F1-F2-F3*(LOG(CS(1,iefc)/CS(2,iefc))-LOG(KV(1,iefc)/KV(2,iefc)))
cewk  F2=(PV(3,iefc)-PV(1,iefc))-((PV(2,iefc)-PV(1,iefc))*F3)
cewk
cewk  t_efc(iefc)=F1/F2
cewk
cewk  F1=LOG(CS(1,iefc)/CS(2,iefc))-LOG(KV(1,iefc)/KV(2,iefc))-
cewk  1      (PV(2,iefc)-PV(1,iefc))*t_efc(iefc)
cewk
cewk  x_efc(iefc)=F1/(S(2,iefc)-S(1,iefc))

csim          else      !X-effective alone is to be calculated.
C
C      Solving for x-effective is based on simultaneously solving two equations
C      of the form  $gps^* = gps(i) * \exp[x\_efc * matrix\_mu(i)]$ , where  $gps^*$  is the
C      corrected activity and is to be the same for both lines,  $gps(i)$  is the

```



```

C      uncorrected activity for line i, x-efc is the correction factor, and
C      matrix-mu(i) is the matrix attenuation at the energy of line i. Solving
C      for x-efc by setting the right-hand side of the above equation for i=1
C      equal to the same for i=2 and algebraically rearranging things, one
C      obtains x-efc = ln{ [gps(1)/gps(2)] * exp[matrix-mu(1) -matrix-mu(2)] }.
C      In this case PCDGPS will be calculated based on the spread between the two
C      corrected gps values, the average of the two values being taken as GPSTAR.
C
csim          t_efc(iefc)=0.0      !Set T-effective to zero so no correction is applied.
csim          x_efc(iefc)=log(exp(s(1,iefc)-s(2,iefc))*      !Log = ln (log10 = log).
csim          (cs(1,iefc)/kv(1,iefc))/(cs(2,iefc)/kv(2,iefc)))
csim          a(1)=(cs(1,iefc)/kv(1,iefc))*exp(x_efc(iefc)*s(1,iefc))
csim          a(2)=(cs(2,iefc)/kv(2,iefc))*exp(x_efc(iefc)*s(2,iefc))
csim          pcdgps(iefc)=100.0*abs((a(1)-a(2))/2.0)/((a(1)+a(2))/2.0)
csim      endif

C
C      Solve for x- and t-effective using a linear least squares method based on
C      a linear form of the attenuation correction formula, namely
C
C      ln(gps(i)) = ln(gps*) - x-efc*matrix-mu(i) - t-efc*self-mu(i),
C
C      where the above terms will be packed as follows:
C      xdata(1,i) = 1.0, xdata(2,i) = -matrix-mu(i), xdata(3,i) = -self-mu(i),
C      and lngps(i) = ln(gps(i)).
C
      pcdgps(iefc)=0.0
      do i=1,jlast      !Pack the data arrays.
          xdata(1,i)=1.0
          xdata(2,i)=-s(i,iefc)
          xdata(3,i)=-pv(i,iefc)
          lngps(i)=log(cs(i,iefc)/kv(i,iefc))      !Log = ln (log10 = log).
          pcdgps(iefc)=pcdgps(iefc)+cs(i,iefc)/kv(i,iefc)
      enddo
      call lstsq(xdata,lngps,ulngps,jlast,min(jlast,3),xtefcs,
          stdevY,xtstdv)
      x_efc(iefc)=xtefcs(2)      !Unpack the results.
      ux_efc(iefc)=xtstdv(2)
      t_efc(iefc)=xtefcs(3)
      ut_efc(iefc)=xtstdv(3)
      pcdgps(iefc)=abs(stdevY*pcdgps(iefc)/real(jlast))      !Calculate the average standard deviation for gps.
      if(jlast.eq.2)then      !T-effective cannot be determined in this case,
          t_efc(iefc)=0.0      !so force it to zero.
          ut_efc(iefc)=0.0
      endif

      if(iefc.gt.2)goto 6100      !No corrections to be made for the latter set(s).

```

C FIND ALL LINES ASSOCIATED WITH NUCLIDES IN ARRAY ID

```
DO 6000 I=1,NPS
  NTG=PKD(I,2,NOT)
  IF(NTG .EQ. 0)GOTO 6000
```

```
DO II=1,NTG
  DO J=1,8
```

```
    if(dxc_a(j,iefc).ne.0)then
      IF(PKD(I,II,IAA)/10.0 .EQ. dxc_A(J,iefc))THEN
        IF(PKD(I,II,IZA) .EQ. dxc_Z(J,iefc))THEN
          R=PKD(I,II,ISA)
          IF(SID .EQ. dxc_ID(J,iefc))THEN
```

C A REQUIRED NUCLIDE HAS BEEN FOUND CORRECT IT (only the activity is corrected;
c no error propagation is being performed at this point, and so the error in
C the activity within PKD is the error associated with the uncorrected activity)

```
CALL EFF(NEFF,PKD(I,3,ERA),EFC)
CALL MU_LOOK(PKD(I,3,ERA),XEM) !For external matrix attenuation.
if(jlast.ge.3)CALL MU_self_LOOK(dxc_id(1,iefc),PKD(I,3,ERA),TPU) !For self (i.e., Pu) attenuation.
PKD(I,II,MCI)=PKD(I,II,MCI)/((EXP(-XEM*x_efc(iefc)))*
1 (EXP(-TPU*t_efc(iefc))))
```

ENDIF

ENDIF

ENDIF

endif

ENDDO

ENDDO

6000 ENDDO

6100 enddo !Done with all sets of X- and T-effective.

CC

C

C Pre-INTCOR LVE test values.

C

```
do 101 j=1,4
  do i=1,10
    if(r_a_1(i,j).eq.0)goto 101
    call find_line(r_z_1(i,j),r_a_1(i,j),r_id_1(i,j),
      r_en_1(i,j)) !Line #1.
    r_pi_1(i,j)=fl.lmc/(fl.br*fl.eff/100.0)
    r_pe_1(i,j)=(fl.err/fl.br)/ !Remove the FIND_LINE-added branching ratio component.
    (fl.br*fl.eff/100.0)
    if(r_a_2(i,j).eq.0)goto 101
    call find_line(r_z_2(i,j),r_a_2(i,j),r_id_2(i,j),
      r_en_2(i,j)) !Line #2.
```

```

        r_pi_2(i,j)=fl.lmc/(fl.br*fl.eff/100.0)
        r_pe_2(i,j)=(fl.err/fl.br)/      !Remove the FIND_LINE-added branching ratio component.
        (fl.br*fl.eff/100.0)
    enddo
101    enddo

C DO INTERFERNCE DECONTAMINATION
C
    J=INDEX(AP.INTERFERENCE_LIB,' ')-1
    HEADER='GE$INTERF$LIB: '//AP.INTERFERENCE_LIB(1:J)//'.LIB'
    OPEN(UNIT=3,NAME=HEADER(1:J+18),IOSTAT=IOS,READONLY,
    1 TYPE='OLD',FORM='UNFORMATTED')
    IF(IOS.NE. 0)THEN
        ERR_MESS='UNABLE TO OPEN INTERFERNCE LIBRARY '
    1    //AP.INTERFERENCE_LIB(1:INDEX(AP.INTERFERENCE_LIB,' ')-1)
        CALL PR_EXIT(ERR_MESS,' ',' ')
    ENDIF
    REWIND(UNIT=3)
    READ(3)INF
    CLOSE(UNIT=3)
    CALL INTCOR(INF)

C
C After-INTCOR LVE test values.
C
    do 102 j=1,4
        do i=1,10
            if(r_a_1(i,j).eq.0)goto 102
            call find_line(r_z_1(i,j),r_a_1(i,j),r_id_1(i,j),
            r_en_1(i,j))      !Line #1.
            r_ai_1(i,j)=fl.cps/(fl.br*fl.eff/100.0)
            r_ae_1(i,j)=(fl.err*(fl.cps/fl.lmc)/fl.br)/      !Remove the FIND_LINE-added branching ratio component.
            (fl.br*fl.eff/100.0)
            r_bi_1(i,j)=fl.lmc/(fl.br*fl.eff/100.0)
            if(r_a_2(i,j).eq.0)goto 102
            call find_line(r_z_2(i,j),r_a_2(i,j),r_id_2(i,j),
            r_en_2(i,j))      !Line #2.
            r_ai_2(i,j)=fl.cps/(fl.br*fl.eff/100.0)
            r_ae_2(i,j)=(fl.err*(fl.cps/fl.lmc)/fl.br)/      !Remove the FIND_LINE-added branching ratio component.
            (fl.br*fl.eff/100.0)
            r_bi_2(i,j)=fl.lmc/(fl.br*fl.eff/100.0)
        enddo
    102    enddo

C
C PRINT THE ATTENUATION INFORMATION
    do i=1,numefc
        if((x_efc(i).eq.0.0).and.(t_efc(i).eq.0.0))then

```

```

        write(6,9301)dxc_id(1,i)(1:2),dxc_a(1,i)
    else
        cx_efc=sigfig(x_efc(i),ux_efc(i))
        if(t_efc(i).ne.0.0)then !X- and T-effective were calculated.
            ct_efc=sigfig(t_efc(i),ut_efc(i))
            write(6,9302)dxc_id(1,i)(1:2),dxc_a(1,i),cx_efc,
            ct_efc
            do j=1,10 !JLAST has been reused by now.
                if(req_eng(j,i).ne.0.0)write(6,9303)
                req_eng(j,i),(cs(j,i)/kv(j,i)),s(j,i),pv(j,i)
            enddo
            write(6,9304)pcdgps(i)
        else !X-effective alone was calculated.
            write(6,9305)dxc_id(1,i)(1:2),dxc_a(1,i),cx_efc,
            req_eng(1,i),(cs(1,i)/kv(1,i)),s(1,i),pv(1,i),
            req_eng(2,i),(cs(2,i)/kv(2,i)),s(2,i),pv(2,i),
            pc dgps(i)
        endif
    endif
enddo

```

C
C Compute and print the appropriate ratio values. The valid ratios that are
C normally calculated are Pu-238 (152.72 keV) : Pu-241 (148.54 keV),
C Pu-240 (160.31 keV) : Pu-239 (161.48 keV),
C Pu-241 (148.54 keV) : Pu-239 (144.20 keV), and
C Am-241 (146.55 keV) - Pu-239 contribution : Pu-239 (144.20 keV).
C

C In case there is a lead liner present, as indicated by the Pu-239 413.71-
C keV line being more than ten times as intense as its 129.30-keV line
C (absorption by the lead altering the otherwise expected ratio), no valid
C ratio values can be calculated.
C

```

1900 iftest=.true. !Initially assume that the test can be performed.
    call find_line(94,239,'PU',413.71)
    r_cp_1=fl.cps
    if(fl.err.eq.0.0)then
        write(6,*)' ERROR!!'
        write(6,*)' DID NOT FIND A VALID 413.71-keV LINE FOR Pu-239'
        write(6,*)' THE TEST FOR A LEAD LINER CANNOT BE PERFORMED'
        write(6,*)' IT WILL BE ASSUMED THAT NO LEAD LINER EXISTS'
        iftest=.false. !The test cannot be performed.
    endif
    call find_line(94,239,'PU',129.30)
    r_cp_2=fl.cps
    if(fl.err.eq.0.0)then
        write(6,*)' ERROR!!'
        write(6,*)' DID NOT FIND A VALID 129.30-keV LINE FOR Pu-239'
    endif

```

```

write(6,*)' THE TEST FOR A LEAD LINER CANNOT BE PERFORMED'
write(6,*)' IT WILL BE ASSUMED THAT NO LEAD LINER EXISTS'
iftest=.false.  !The test cannot be performed.
endif
if(iftest.and.(r_cp_1.gt.(10.0*r_cp_2)))then
write(6,9200)
ir.ratflg(1)='L  '  !Lead liner present flag.
ir.nratio=1  !To ensure that the above flag is read.
goto 7000
endif

```

```

C
C Check for excessive U-235, as indicated by its 185.72-keV line being more
C intense than the its 143.77-keV line and the Pu-239 144.20-keV line
C combined.

```

```

C
C iftest=.true.  !Initially assume that the test can be performed.
C call find_line(92,235,'U',185.72)
C r_cp_1=fl.cps
C if(fl.err.eq.0.0)then
C   write(6,*)' ERROR!!'
C   write(6,*)' DID NOT FIND A VALID 185.72-keV LINE FOR U-235'
C   write(6,*)' THE TEST FOR EXCESSIVE URANIUM CANNOT BE PERFORMED'
C   write(6,*)' IT WILL BE ASSUMED THAT THERE IS NOT EXCESSIVE '//
C   'URANIUM PRESENT'
C   iftest=.false.  !The test cannot be performed.
C   endif

```

```

C call find_line(92,235,'U',143.77)
C r_cp_2=fl.cps
C if(fl.err.eq.0.0)then
C   write(6,*)' ERROR!!'
C   write(6,*)' DID NOT FIND A VALID 143.77-keV LINE FOR U-235'
C   write(6,*)' THE TEST FOR EXCESSIVE URANIUM CANNOT BE PERFORMED'
C   write(6,*)' IT WILL BE ASSUMED THAT THERE IS NOT EXCESSIVE '//
C   'URANIUM PRESENT'
C   iftest=.false.  !The test cannot be performed.
C   endif

```

```

C call find_line(94,239,'PU',144.20)
C r_cp_3=fl.cps
C if(fl.err.eq.0.0)then
C   write(6,*)' ERROR!!'
C   write(6,*)' DID NOT FIND A VALID 144.20-keV LINE FOR Pu-239'
C   write(6,*)' THE TEST FOR EXCESSIVE URANIUM CANNOT BE PERFORMED'
C   write(6,*)' IT WILL BE ASSUMED THAT THERE IS NOT EXCESSIVE '//
C   'URANIUM PRESENT'
C   iftest=.false.  !The test cannot be performed.
C   endif

```

```

C if(iftest.and.((r_cp_1**2).gt.(r_cp_2+r_cp_3)))then  !Print a warning.

```

```

C      write(6,9202)
C      ir.ratflg(1)(1:1)='U'    !Excessive uranium flag.
C      else    !Calculate the regular set of ratios (i.e., the first four in the above list).
C      write(6,9204)
C      ir.ratflg(1)(1:1)='S'    !Standard set flag.
C      endif
C
C Calculate the isotopic ratios. Note that these ratios are calculated after
C interference decontamination, and so corrections to the Pu-239-contaminated
C Am-241 peak intensities, for example, have already been performed, but before
C RSOLVE and so the line intensities and errors are not yet in microCuries.
C
      jndex=3    !Start with the most restrictive set for testing purposes.
      jndexn=1   !Initially assume that the unrestricted set will be calculated.
      jndexo=3
      iftest=.true.    !This is the test pass, that is, checking for which JNDEX should really be used.
      iefc=1    !For now assume that the only ratios involve Pu-239-related isotopes.
      do i=1,10    !Find JLAST, i.e., the number of lines to be used.
        if(req_eng(i,iefc).ne.0.0)jlast=i
      enddo
1920  ir.nratio=0
      i=0
1925  i=i+1    !Cannot use a do-loop structure because the set in use can change.
      if(i.le.20)then
        if(raten1(i,jndex).eq.0.0)goto 1925    !No more ratios to calculate in this set.
        if(iftest.and.(.not.
          .   (((ratid1(i,jndex).eq.'U').and.(rata1(i,jndex).eq.235)).and.
          .   ((ratid2(i,jndex).eq.'PU').and.(rata2(i,jndex).eq.239))).or.
          .   (((ratid1(i,jndex).eq.'AM').and.(rata1(i,jndex).eq.241)).and.
          .   ((ratid2(i,jndex).eq.'PU').and.(rata2(i,jndex).eq.239))))))
          goto 1925    !Not one that needs to be calculated at this point.
        ir.nratio=ir.nratio+1
        if(jndex.eq.1)ir.ratflg(ir.nratio)='S'    !Reset the warnings flag.
        if(jndex.eq.2)ir.ratflg(ir.nratio)='U'
        if(jndex.eq.3)ir.ratflg(ir.nratio)='A'
C
C If RATID1 is blank, use the last non-blank radionuclide identification.
C
      if(ratid1(i,jndex).eq.' ')then
        ratid1(i,jndex)=ratid1((i-1),jndex)
        rata1(i,jndex)=rata1((i-1),jndex)
        ratz1(i,jndex)=ratz1((i-1),jndex)
        ratid2(i,jndex)=ratid2((i-1),jndex)
        rata2(i,jndex)=rata2((i-1),jndex)
        ratz2(i,jndex)=ratz2((i-1),jndex)
      else    !Reset NDATUM for weighted averages.
        ndatum=0

```

```

endif
C
C Find both of the lines.
C
call find_line(ratz1(i,jindex),rata1(i,jindex),ratid1(i,jindex),
. raten1(i,jindex)) !Line #1.
if(fl.err.eq.0.0)then
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE FIRST LINE USED IS INVALID'
  ir.ratflg(ir.nratio)(4:4)='1'
  ratio=0.0
  eratio=0.0
  call find_line(ratz2(i,jindex),rata2(i,jindex),
. ratid2(i,jindex),raten2(i,jindex)) !Line #2.
  if(fl.err.eq.0.0)then !Is the second line also invalid?
    comment=' - BOTH LINES USED ARE INVALID'
    ir.ratflg(ir.nratio)(4:4)='B'
  endif
  goto 1935 !Do not attempt to calculate a ratio.
endif
ir_f_1=fl.flag
c if(btest(ir_f_1,0))then
c   cerati='(RATIO NOT DETERMINABLE)'
c   comment=' - THE FIRST LINE WAS FIT BY LINEAR METHODS'
c   ir.ratflg(ir.nratio)(2:2)='1'
c   ratio=0.0
c   eratio=0.0
c   call find_line(ratz2(i,jindex),rata2(i,jindex),
c   . ratid2(i,jindex),raten2(i,jindex)) !Line #2.
c   ir_f_2=fl.flag
c   if(btest(ir_f_2,0))then
c     comment=' - BOTH LINES WERE FIT BY LINEAR METHODS'
c     ir.ratflg(ir.nratio)(2:2)='B'
c   endif
c   goto 1935 !Do not attempt to calculate a ratio.
c   endif
r_mc_1=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
r_er_1=(fl.err*100.0*fl.dca/(fl.eff*fl.br*3.7E4))/fl.br !Remove the FIND_LINE-added branching ratio
component.
comment=' '
if(.not.true_positive(r_mc_1,(100.0*r_er_1/r_mc_1),2.0))then !Ratios from non-true-positive
activities are invalid.
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE FIRST LINE USED HAS A NON-TRUE-POSITIVE '//'
  'ACTIVITY'
  ir.ratflg(ir.nratio)(3:3)='1'
  ratio=0.0

```

```

eratio=0.0
call find_line(ratz2(i,jndex),rata2(i,jndex),
  ratid2(i,jndex),raten2(i,jndex)) !Line #2.
r_mc_2=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
r_er_2=(fl.err*100.0*fl.dca/(fl.eff*fl.br*3.7E4))/fl.br !Remove the FIND_LINE-added branching
ratio component.
if(.not.true_positive(r_mc_2,(100.0*r_er_2/r_mc_2),2.0))then !Ratios from non-true-positive
activities are invalid.
  comment=' - BOTH LINES USED HAVE NON-TRUE-POSITIVE '//
  'ACTIVITIES'
  ir.ratflg(ir.nratio)(3:3)='B'
endif
goto 1935 !Do not attempt to calculate a ratio.
endif
r_br_1=fl.br/100.0 !Convert from a percent to a fraction.
r_eb_1=br_uc(ratal(i,jndex),ratz1(i,jndex),ratid1(i,jndex),
  raten1(i,jndex))/100.0 !Convert from a percent to a fraction.
if(r_eb_1.le.0.0)r_eb_1=0.0 !Not found by BR_UC? Set the error to zero.
r_hl_1=fl.hl
r_cp_1=fl.cps
r_ec_1=(r_er_1/r_mc_1)*r_cp_1 !Revert to fractional for MCI, then absolute for CPS.
r_em_1=r_mc_1*sqrt((r_ec_1/r_cp_1)**2+(r_eb_1/r_br_1)**2)
if(btest(ir_f_1,5).or.btest(ir_f_1,6))then !Interference decontaminated.
  do j=1,inf.noentry !Look for the match.
    write(ratid3,fmt='(a4)')inf.conid(j)
    if((abs(inf.econ(j)-raten1(i,jndex)).le.1.0).and.
      (ratid3.eq.ratid1(i,jndex)).and.
      (inf.conz(j).eq.ratz1(i,jndex)).and.
      ((inf.cona(j)/10).eq.ratal(i,jndex)))then !Match found!
      write(ratid3,fmt='(a4)')inf.cleanid(j)
      call find_line(inf.cleanz(j),inf.cleana(j),ratid3,
        inf.eclean(j))
      r_eb_3=br_uc(inf.cleanz(j),inf.cleana(j),ratid3,
        inf.eclean(j))
      r_mc_3=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
      r_em_1=sqrt(r_em_1**2+(r_mc_3*r_br_1/(fl.br/100.0))**2
        *((r_eb_1/r_br_1)**2+(r_eb_3/fl.br)**2)) !Add the extra components.
    endif
  enddo
endif
call mu_look(raten1(i,jndex),r_um_1) !For x-effective.
if(jlast.ge.3)then !For t-effective.
  call mu_self_look(dxc_id(1,iefc),raten1(i,jndex),r_us_1)
else !No t-effective, so set the attenuation factor to zero.
  r_us_1=0.0
endif
call find_line(ratz2(i,jndex),rata2(i,jndex),ratid2(i,jndex),

```



```

      raten2(i,jndex))      !Line #2.
if(fl.err.eq.0.0)then
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE SECOND LINE USED IS INVALID'
  ir.ratflg(ir.nratio)(4:4)='2'
  ratio=0.0
  eratio=0.0
  goto 1935      !Do not attempt to calculate a ratio.
endif
ir_f_2=fl.flag
c   if(btest(ir_f_2,0))then
c     cerati='(RATIO NOT DETERMINABLE)'
c     comment=' - THE SECOND LINE WAS FIT BY LINEAR METHODS'
c     ir.ratflg(ir.nratio)(2:2)='2'
c     ratio=0.0
c     eratio=0.0
c     goto 1935      !Do not attempt to calculate a ratio.
c   endif
r_mc_2=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
r_er_2=(fl.err*100.0*fl.dca/(fl.eff*fl.br*3.7E4))/fl.br      !Remove the FIND_LINE-added branching ratio
component.
if(.not.true_positive(r_mc_2,(100.0*r_er_2/r_mc_2),2.0))then      !Ratios from non-true-positive
activities are invalid.
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE SECOND LINE USED HAS A NON-TRUE-POSITIVE '//
  'ACTIVITY'
  ir.ratflg(ir.nratio)(3:3)='2'
  ratio=0.0
  eratio=0.0
  goto 1935      !Do not attempt to calculate a ratio.
endif
r_br_2=fl.br/100.0      !Convert from a percent to a fraction.
r_eb_2=br_uc(rata2(i,jndex),ratz2(i,jndex),ratid2(i,jndex),
  raten2(i,jndex))/100.0      !Convert from a percent to a fraction.
if(r_eb_2.le.0.0)r_eb_2=0.0      !Not found by BR_UC? Set the error to zero.
r_hl_2=fl.hl
r_cp_2=fl.cps
r_ec_2=(r_er_2/r_mc_2)*r_cp_2      !Revert to fractional for MCI, then absolute for CPS.
r_em_2=r_mc_2*sqrt((r_ec_2/r_cp_2)**2+(r_eb_2/r_br_2)**2)
if(btest(ir_f_2,5).or.btest(ir_f_2,6))then      !Interference decontaminated.
  do j=1,inf.noentry      !Look for the match.
    write(ratid3,fmt='(a4)')inf.conid(j)
    if((abs(inf.econ(j)-raten2(i,jndex)).le.1.0).and.
      (ratid3.eq.ratid2(i,jndex)).and.
      (inf.conz(j).eq.ratz2(i,jndex)).and.
      ((inf.cona(j)/10).eq.rata2(i,jndex)))then      !Match found!
      write(ratid3,fmt='(a4)')inf.cleanid(j)
    enddo

```

```

        call find_line(Inf.cleanz(j),Inf.cleana(j),ratid3,
            Inf.eclean(j))
        r_eb_3=br_uc(Inf.cleanz(j),Inf.cleana(j),ratid3,
            Inf.eclean(j))
        r_mc_3=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
        r_em_2=sqrt(r_em_2**2+(r_mc_3*r_br_2/(fl.br/100.0))**2
            *(r_eb_2/r_br_2)**2+(r_eb_3/fl.br)**2)) !Add the extra components.
        endif
    enddo
endif
call mu_look(raten2(i,jindex),r_um_2) !For x-effective.
if(jlast.ge.3)then !For t-effective.
    call mu_self_look(dxc_id(1,iefc),raten2(i,jindex),r_us_2)
else !No t-effective, so set the attenuation factor to zero.
    r_us_2=0.0
endif

```

C
C Calculate the ratio.

```

        ratio=(r_mc_1*r_hl_1)/(r_mc_2*r_hl_2)

```

C
C Propagate the error in the ratio, neglecting contributions from the half
C lives, attenuation factors, and efficiencies.

```

        eratio=ratio*sqrt((r_em_1/r_mc_1)**2+(r_em_2/r_mc_2)**2+
            (r_eb_1/r_br_1)**2+(r_eb_2/r_br_2)**2+
            (ux_efc(iefc)*(r_um_1-r_um_2))**2+
            (ut_efc(iefc)*(r_us_1-r_us_2))**2)

```

C
C Add the above to the list of values and errors to be used in obtaining a
C weighted average, unless this is the test run.

```

        if(.not.iftest)then
            ndatum=ndatum+1
            values(ndatum)=ratio
            errors(ndatum)=eratio
            indexs(ndatum)=i

```

C
C Also perform a weighted average with outlier testing if needed and
C print the results if this is the last ratio in a set.

```

        if((ratid1((i+1),jindex).ne.' ').or.
            (raten1((i+1),jindex).eq.0.0))then
1930      comment=' '
            call wavg_o(values,errors,ndatum,.true.,mean,stdev,
                wmean,wstdev,rchisq,ifvald)

```

```

C      Print the results.
C
      if(ndatum.ne.1)then      !Print the ratios for more than one point.
        do j=1,ndatum
          cerati=sigfig((values(j)*100.0),
            (errors(j)*100.0))      !Convert to percentages.
          if(ifvald(j))then      !Note this ratio as having been used.
            comment=' - USED IN THE FOLLOWING '//
              'WEIGHTED AVERAGE'
          else      !Note this ratio as having been an outlier.
            comment=' - NOT USED IN THE FOLLOWING '//
              'WEIGHTED AVERAGE; DETERMINED TO BE '//
              'AN OUTLIER'
          endif
          write(6,9220)ratid1((indexs(j)),jindex)(1:2),
            rata1((indexs(j)),jindex),
            raten1((indexs(j)),jindex),
            ratid2((indexs(j)),jindex)(1:2),
            rata2((indexs(j)),jindex),
            raten2((indexs(j)),jindex),cerati,comment
        enddo
        cerati=sigfig((wmean*100.0),(wstdev*100.0))      !Convert to percentages.
        write(6,9224)ratid1(i,jindex)(1:2),rata1(i,jindex),
          ratid2(i,jindex)(1:2),rata2(i,jindex),cerati
      else      !For only one data point.
        cerati=sigfig((wmean*100.0),(wstdev*100.0))      !Convert to percentages.
        write(6,9222)ratid1(indexs(1),jindex)(1:2),
          rata1(indexs(1),jindex),raten1(indexs(1),jindex),
          ratid2(indexs(1),jindex)(1:2),
          rata2(indexs(1),jindex),raten2(indexs(1),jindex),
          cerati,comment
      endif

C
C      Check if the precision is better than twenty percent.  If it is not,
C      use the next more restrictive set of lines for this ratio (if one is
C      available and and it contains different energies).
C
      if((wstdev/wmean).gt.0.20)then
        write(6,9226)
1932      if(jindex.ne.3)then
          jindex=jindex+1      !Use the next more restrictive set.
          if((jindex-1).eq.jindexo)old=i      !Retain the current I for resetting to later.
          do j=1,20      !Find the new I for this ratio in the new set.
            if((ratid1(i,(jindex-1)).eq.ratid1(j,jindex)).and.
              (rata1(i,(jindex-1)).eq.rata1(j,jindex)).and.
              (ratz1(i,(jindex-1)).eq.ratz1(j,jindex)).and.
              (ratid2(i,(jindex-1)).eq.ratid2(j,jindex)).and.

```

```

.      (rata2(i,(jindex-1)).eq.rata2(j,jindex)).and.
.      (ratz2(i,(jindex-1)).eq.ratz2(j,jindex)))then
.      ilast=i
.      do k=(i-1),1,-1    !Find the first pair for I.
.      if((ratid1(k,(jindex-1)).ne.
.      ratid1((k+1),(jindex-1))).or.
.      (rata1(k,(jindex-1)).ne.
.      rata1((k+1),(jindex-1))).or.
.      (ratz1(k,(jindex-1)).ne.
.      ratz1((k+1),(jindex-1))).or.
.      (ratid2(k,(jindex-1)).ne.
.      ratid2((k+1),(jindex-1))).or.
.      (rata2(k,(jindex-1)).ne.
.      rata2((k+1),(jindex-1))).or.
.      (ratz2(k,(jindex-1)).ne.
.      ratz2((k+1),(jindex-1))))then
.      i=k+1
.      goto 1933
.      endif
.      enddo
1933  do k=(j+1),20    !Find the last pair for J.
.      if((ratid1(k,jindex).ne.' ').or.
.      (raten1(k,jindex).eq.0.0))then
.      jlast=k-1
.      goto 1934
.      endif
.      enddo
1934  if((jlast-j).ne.(ilast-i))then    !Different set, so try it.
.      i=j-1    !Since I will be incremented by one.
.      goto 1925
.      endif
.      do k=i,ilast    !Check for energy pair mismatch.
.      ifmtch=.true.    !Assume no match is found.
.      do l=j,jlast
.      if((raten1(k,(jindex-1)).eq.
.      raten1(l,jindex)).and.
.      (raten2(k,(jindex-1)).eq.
.      raten2(l,jindex)))ifmtch=.false.    !Match found.
.      enddo
.      if(ifmtch)then    !Mismatch, so use this next different set.
.      i=j-1    !Since I will be incremented by one.
.      goto 1925
.      endif
.      enddo
.      i=j    !Try the next set, if there is one.
.      goto 1932    !Try the next set, if there is one.
.      endif

```

```

        enddo
        goto 1932    !Try the next set, if there is one.
    endif
    write(6,9228)    !There is not a more restrictive set to try.
c      jndex=jndexo
c      i=iold
    endif
    ratio=wmean      !Store the results.
    eratio=wstdev
    goto 1940
    endif
    goto 1925    !Do the next ratio.
    endif

C
C  If this is the test run, check for excessive U-235 or Am-241.
C
1935  if(iftest)then
        if((ratid1(i,jndex).eq.'U').and.(ratio.gt.0.044))    !Excessive U-235.
            jndexn=max(jndexn,2)    !Excessive Am-241 takes precedence over excessive U-235.
            if((ratid1(i,jndex).eq.'AM').and.(ratio.gt.0.013))    !Excessive Am-241.
                jndexn=max(jndexn,3)
            goto 1925    !No printing at this point.
        endif
        if((ratid1((i+1),jndex).eq.' ').and.
            (raten1((i+1),jndex).ne.0.0))then    !For invalid ratios.
            write(6,9220)ratid1(i,jndex)(1:2),ratal(i,jndex),
            raten1(i,jndex),ratid2(i,jndex)(1:2),rata2(i,jndex),
            raten2(i,jndex),cerati,comment
            goto 1925
        else    !For invalid ratios that are the last of a set.
            if(ndatum.ne.0)then    !In case there are valid ratios to process in this set.
                write(6,9220)ratid1(i,jndex)(1:2),ratal(i,jndex),
                raten1(i,jndex),ratid2(i,jndex)(1:2),
                rata2(i,jndex),raten2(i,jndex),cerati,comment
                goto 1930
            else
                write(6,9222)ratid1(i,jndex)(1:2),ratal(i,jndex),
                raten1(i,jndex),ratid2(i,jndex)(1:2),
                rata2(i,jndex),raten2(i,jndex),cerati,comment
            endif
        endif
    endif

C
C  Store the results in the structure.
C
1940  ir.ratios(ir.nratio)=ratio*100.0    !Convert to percentages.
      ir.uncrat(ir.nratio)=eratio*100.0
      ir.rat_il(ir.nratio)=ratid1(i,jndex)

```

```

ir.rat_a1(ir.nratio)=ratal(i,jindex)
ir.rat_i2(ir.nratio)=ratid2(i,jindex)
ir.rat_a2(ir.nratio)=rata2(i,jindex)

```

C
C Before doing the next ratio, has a more restrictive set been used for the
C last ratio? If it has, reset I and JNDEX appropriately.
C

```

      if(jindex.ne.jindexo)then
        do j=(iold+1),20  !Find the next ratio in the prior set.
          if(((ratid1(j,jindexo).ne.ratid1(i,jindex)).or.
            (ratal(j,jindexo).ne.ratal(i,jindex)).or.
            (ratz1(j,jindexo).ne.ratz1(i,jindex)).or.
            (ratid2(j,jindexo).ne.ratid2(i,jindex)).or.
            (rata2(j,jindexo).ne.rata2(i,jindex)).or.
            (ratz2(j,jindexo).ne.ratz2(i,jindex))).and.
            (ratid1(j,jindexo).ne.' ').and.
            (raten1(j,jindexo).ne.0.0)))then
            write(6,9230)
            i=j-1  !Since I will be incremented by one.
            jindex=jindexo
            goto 1925
          endif
        enddo
        goto 1945  !All done - no more to be found.
      endif
      goto 1925
    endif

```

C
C If this was the test pass, print an appropriate header based on what JNDEX
C was determined to be, then proceed to calculate and print the ratios.
C

```

      if(iftest)then
        iftest=.false.  !No longer testing.
        jindex=jindexn  !Assign the new JNDEX.
        jindexo=jindex  !Retain the current JNDEX for future use.
        if(jindex.eq.1)write(6,9204)
        if(jindex.eq.2)write(6,9202)
        if(jindex.eq.3)write(6,9206)
        goto 1920
      endif

```

C
C To separate the following test section from the rest of the output . . .
C
1945 write(6,9299)

C
C Calculate the isotopic ratios. Note that these ratios are calculated after
C interference decontamination, and so corrections to the Pu-239-contaminated

C Am-241 peak intensities, for example, have already been performed, but before
 C RSOLVE and so the line intensities and errors are not yet in microCuries.

```

C
  jindex=1
  iefc=1 !For now assume that the only ratios involve Pu-239-related isotopes.
  do i=1,10 !Find JLAST, i.e., the number of lines to be used.
    if(req_eng(i,iefc).ne.0.0)jlast=i
  enddo
1902 if(jindex.eq.1)write(6,9204)
  if(jindex.eq.2)write(6,9202)
  if(jindex.eq.3)write(6,9206)
  if(jindex.eq.4)write(6,9208)
  do i=1,10
    if(r_a_1(i,jindex).eq.0)goto 1910 !No more ratios to calculate.
C
C Find both of the lines.
C
    call find_line(r_z_1(i,jindex),r_a_1(i,jindex),r_id_1(i,jindex),
      r_en_1(i,jindex)) !Line #1.
    if(fl.err.eq.0.0)then
      cerati='(RATIO NOT DETERMINABLE)'
      comment=' - THE FIRST LINE USED IS INVALID'
      ratio=0.0
      eratio=0.0
      call find_line(r_z_2(i,jindex),r_a_2(i,jindex),
        r_id_2(i,jindex),r_en_2(i,jindex)) !Line #2.
      if(fl.err.eq.0.0)then !Is the second line also invalid?
        comment=' - BOTH LINES USED ARE INVALID'
      endif
      goto 1905 !Do not attempt to calculate a ratio.
    endif
    r_er_1=(fl.err*100.0*fl.dca/(fl.eff*fl.br*3.7E4))/fl.br !Remove the FIND_LINE-added branching ratio
component.
    r_mc_1=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
    r_br_1=fl.br/100.0 !Convert from a percent to a fraction.
    r_eb_1=br_uc(r_a_1(i,jindex),r_z_1(i,jindex),r_id_1(i,jindex),
      r_en_1(i,jindex))/100.0 !Convert from a percent to a fraction.
    if(r_eb_1.le.0.0)r_eb_1=0.0 !Not found by BR_UC? Set the error to zero.
    r_hl_1=fl.hl
    r_cp_1=fl.cps
    r_ec_1=(r_er_1/r_mc_1)*r_cp_1 !Revert to fractional for MCI, then absolute for CPS.
    r_em_1=r_mc_1*sqrt((r_ec_1/r_cp_1)**2+(r_eb_1/r_br_1)**2)
    ir_f_1=fl.flag
    if(btest(ir_f_1,5).or.btest(ir_f_1,6))then !Interference decontaminated.
      do j=1,inf.noentry !Look for the match.
        write(r_id_3,fmt='(a4)')inf.conid(j)
        if((abs(inf.econ(j)-r_en_1(i,jindex)).le.1.0).and.

```

```

        (r_id_3.eq.r_id_1(i,jindex)).and.
        (inf.conz(j).eq.r_z_1(i,jindex)).and.
        ((inf.cona(j)/10).eq.r_a_1(i,jindex)))then    !Match found!
            write(r_id_3,fmt='(a4)')inf.cleanid(j)
            call find_line(inf.cleanz(j),inf.cleana(j),r_id_3,
                inf.eclean(j))
            r_eb_3=br_uc(inf.cleanz(j),inf.cleana(j),r_id_3,
                inf.eclean(j))
            r_mc_3=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
            r_em_1=sqrt(r_em_1**2+(r_mc_3*r_br_1/(fl.br/100.0))**2
                *((r_eb_1/r_br_1)**2+(r_eb_3/fl.br)**2))    !Add the extra components.
        endif
    enddo
endif
call mu_look(r_en_1(i,jindex),r_um_1)    !For x-effective.
if(jlast.ge.3)then    !For t-effective.
    call mu_self_look(dxc_id(1,iefc),r_en_1(i,jindex),r_us_1)
else    !No t-effective, so set the attenuation factor to zero.
    r_us_1=0.0
endif
call find_line(r_z_2(i,jindex),r_a_2(i,jindex),r_id_2(i,jindex),
    r_en_2(i,jindex))    !Line #2.
if(fl.err.eq.0.0)then
    cerati='(RATIO NOT DETERMINABLE)'
    comment=' - THE SECOND LINE USED IS INVALID'
    ratio=0.0
    eratio=0.0
    goto 1905    !Do not attempt to calculate a ratio.
endif
r_er_2=(fl.err*100.0*fl.dca/(fl.eff*fl.br*3.7E4))/fl.br    !Remove the FIND_LINE-added branching ratio
component.
r_mc_2=fl.lmc*100.0*fl.dca/(fl.eff*fl.br*3.7E4)
r_br_2=fl.br/100.0    !Convert from a percent to a fraction.
r_eb_2=br_uc(r_a_2(i,jindex),r_z_2(i,jindex),r_id_2(i,jindex),
    r_en_2(i,jindex))/100.0    !Convert from a percent to a fraction.
if(r_eb_2.le.0.0)r_eb_2=0.0    !Not found by BR_UC? Set the error to zero.
r_hl_2=fl.hl
r_cp_2=fl.cps
r_ec_2=(r_er_2/r_mc_2)*r_cp_2    !Revert to fractional for MCI, then absolute for CPS.
r_em_2=r_mc_2*sqrt((r_ec_2/r_cp_2)**2+(r_eb_2/r_br_2)**2)
ir_f_2=fl.flag
if(btest(ir_f_2,5).or.btest(ir_f_2,6))then    !Interference decontaminated.
    do j=1,inf.noentry    !Look for the match.
        write(r_id_3,fmt='(a4)')inf.conid(j)
        if((abs(inf.econ(j)-r_en_2(i,jindex)).le.1.0).and.
            (r_id_3.eq.r_id_2(i,jindex)).and.
            (inf.conz(j).eq.r_z_2(i,jindex)).and.

```



```

.      ((inf.cona(j)/10).eq.r_a_2(i,jindex)))then      !Match found!
      write(r_id_3,fmt='(a4')inf.cleanid(j)
      call find_line(inf.cleanz(j),inf.cleana(j),r_id_3,
.      inf.eclean(j))
      r_eb_3=br_uc(inf.cleanz(j),inf.cleana(j),r_id_3,
.      inf.eclean(j))
      r_mc_3=fl.lmc*100.0*fl.dca/(fl. eff*fl.br*3.7E4)
      r_em_2=sqrt(r_em_2**2+(r_mc_3*r_br_2/(fl.br/100.0))**2
.      *((r_eb_2/r_br_2)**2+(r_eb_3/fl.br)**2))      !Add the extra components.
      endif
    enddo
  endif
  call mu_look(r_en_2(i,jindex),r_um_2)      !For x-effective.
  if(jlast.ge.3)then      !For t-effective.
    call mu_self_look(dxc_id(1,iefc),r_en_2(i,jindex),r_us_2)
    else      !No t-effective, so set the attenuation factor to zero.
      r_us_2=0.0
    endif

```

C
C
C Calculate the ratio.

```

      ratio=(r_mc_1*r_hl_1)/(r_mc_2*r_hl_2)

```

C
C Propagate the error in the ratio, neglecting contributions from the half
C lives, attenuation factors, and efficiencies.

```

      eratio=ratio*sqrt((r_em_1/r_mc_1)**2+(r_em_2/r_mc_2)**2+
.      (r_eb_1/r_br_1)**2+(r_eb_2/r_br_2)**2+
.      (ux_efc(iefc)*(r_um_1-r_um_2))**2+
.      (ut_efc(iefc)*(r_us_1-r_us_2))**2)
      cerati=sigfig((ratio*100.0),(eratio*100.0))      !Convert to percentages.
      r_frer=sqrt((r_er_1/r_mc_1)**2+(r_er_2/r_mc_2)**2)*100.0

```

C
C Print the results.

```

      comment=' '
      if(btest(ir_f_1,0))then
        comment=' - THE FIRST LINE USED '//
.        'WAS FIT BY LINEAR METHODS - THE RATIO MAY BE IN ERROR'
        endif
      if(btest(ir_f_2,0))then
        comment=' - THE SECOND LINE USED '//
.        'WERE FIT BY LINEAR METHODS - THE RATIO MAY BE IN ERROR'
        endif
      if(btest((ir_f_1.and.ir_f_2),0))then
        comment=' - BOTH LINES USED '//
.        'WERE FIT BY LINEAR METHODS - THE RATIO MAY BE IN ERROR'

```

```

endif
if(r_mc_1.lt.0.0)then !Ratios from negative activities are invalid.
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE FIRST LINE USED HAS AN ACTIVITY '//
  'BELOW DETECTION LIMITS'
endif
if(r_mc_2.lt.0.0)then !Ratios from negative activities are invalid.
  cerati='(RATIO NOT DETERMINABLE)'
  comment=' - THE SECOND LINE USED HAS AN ACTIVITY '//
  'BELOW DETECTION LIMITS'
endif
if((r_mc_1.lt.0.0).and.(r_mc_2.lt.0.0))then !Ratios from negative activities are invalid.
  comment=' - BOTH LINES USED HAVE ACTIVITIES '//
  'BELOW DETECTION LIMITS'
endif
1905 write(6,9210)r_id_1(i,jindex)(1:2),r_a_1(i,jindex),
  . r_en_1(i,jindex),r_id_2(i,jindex)(1:2),r_a_2(i,jindex),
  . r_en_2(i,jindex),cerati,comment,
  . (r_ai_1(i,jindex)*exp(x_efc(iefc)*r_um_1)*
  . exp(t_efc(iefc)*r_us_1)), !Corrected numerator . . .
  . sqrt(exp(x_efc(iefc)*r_um_1)**2*exp(t_efc(iefc)*r_us_1)**2*
  . r_ae_1(i,jindex)**2+(r_um_1*r_ai_1(i,jindex))**2*
  . ux_efc(iefc)**2+(r_us_1*r_ai_1(i,jindex))**2*ut_efc(iefc)**2), ! and error, . . .
  . exp(x_efc(iefc)*r_um_1), ! . . . matrix correction, . . .
  . exp(t_efc(iefc)*r_us_1), ! . . . self correction, . . .
  . (r_pi_1(i,jindex)-r_bi_1(i,jindex))*
  . (r_ai_1(i,jindex)/r_pi_1(i,jindex)), ! . . . INTCOR subtraction, . . .
  . r_xi_1(i,jindex),r_xe_1(i,jindex), ! and the uncorrected numerator and error.
  . (r_ai_2(i,jindex)*exp(x_efc(iefc)*r_um_2)*
  . exp(t_efc(iefc)*r_us_2)), !Corrected denominator . . .
  . sqrt(exp(x_efc(iefc)*r_um_2)**2*exp(t_efc(iefc)*r_us_2)**2*
  . r_ae_2(i,jindex)**2+(r_um_2*r_ai_2(i,jindex))**2*
  . ux_efc(iefc)**2+(r_us_2*r_ai_2(i,jindex))**2*ut_efc(iefc)**2), ! and error, . . .
  . exp(x_efc(iefc)*r_um_2), ! . . . matrix correction, . . .
  . exp(t_efc(iefc)*r_us_2), ! . . . self correction, . . .
  . (r_pi_2(i,jindex)-r_bi_2(i,jindex))*
  . (r_ai_2(i,jindex)/r_pi_2(i,jindex)), ! . . . INTCOR subtraction, . . .
  . r_xi_2(i,jindex),r_xe_2(i,jindex) ! and the uncorrected numerator and error.
1910 enddo
  jindex=jindex+1
  if(jindex.le.4)goto 1902
C
C PRINT RESULTS SANS-MICRO CURRIES
  CALL CBAT_PRGIDXC(AP.BACK_CORRECT,AP.ISOTOPE_LIB,
  1 AP.REQUIRED_LIB,AP.INTERFERENCE_LIB,AP.PRINT_RESULTS)
C
  CALL CHECK_ICODE(NRM) !ELIMINATE LINES WITHOUT ICODE 1 LINES

```

```

        IF(AP.PRINT_RESULTS)THEN
            WRITE(6,9005)NRM
        ENDIF
C
C FIND ALL LINES ASSOCIATED WITH NUCLIDES IN ARRAY ID

        DO 6010 I=1,NPS
            NTG=PKD(I,2,NOT)
            IF(NTG .EQ. 0)GOTO 6010

            DO II=1,NTG
                DO J=1,8
                    if(dxc_a(j,iefc).ne.0)then
                        IF(PKD(I,II,IAA)/10.0 .EQ. dxc_A(J,iefc))THEN
                            IF(PKD(I,II,IZA) .EQ. dxc_Z(J,iefc))THEN
                                R=PKD(I,II,ISA)
                                IF(SID .EQ. dxc_ID(J,iefc))THEN
C
C Error propagation is being performed at this point.
C
                                CALL EFF(NEFF,PKD(I,3,ERA),EFC)
                                CALL MU_LOOK(PKD(I,3,ERA),XEM)      !For external matrix attenuation.
                                if(jlast.ge.3)CALL MU_self_LOOK(dxc_id(1,iefc),PKD(I,3,ERA),TPU)    !For self (i.e., Pu) attenuation.
                                PKD(I,II,EMCI)=100.0*sqrt((PKD(I,II,EMCI)/100.0)**2+
                                2 (xem*ux_efc(iefc))**2+(tpu*ut_efc(iefc))**2)
                                    ENDIF
                                ENDIF
                            endif
                        ENDIF
                    ENDIF
                ENDIF
            ENDDO
        ENDDO
6010  ENDDO
        CALL RSOLVE
        IF(AP.PRINT_RESULTS)CALL PR_NC_SUMMARY

C NET VALUE FOR PU-239
        PU239NET=0.0
C
C MOVE THE PEAK INFORMATION INTO THE SAVE DATA STRUCTURE 'TWR_RESULTS'. STRUCTURE
C DEFINITIONS ARE IN TWR_RESULTS.INC
        TR.NPS=VHEAD(NP)
        IF(TR.NPS .GT. 200)THEN
            WRITE(6,9310)
            TR.NPS=100
        ENDIF
        DO I=1,TR.NPS

```

```

        TR.PEAK(I,1)=PKD(I,3,ERA)
        TR.PEAK(I,2)=PKD(I,2,CPS)
        TR.PEAK(I,3)=PKD(I,2,DA)
        TR.PEAK(I,4)=PKD(I,3,EFA)
    ENDDO
C
C MOVE NUCLIDE INFO TO THE STRUCTURE
C
    DO I=1,TR.NPS
        NTG=PKD(I,2,NOT)
        IF(NTG.NE.0)THEN
            DO II=1,NTG
                PKD(I,II,HLA)=ABS(PKD(I,II,HLA))
            ENDDO
        ENDIF
    ENDDO
C
C PASS THORUGH THE PKD ARRAY EXTRACTING EACH DIFFERENT ISOTOPE VALUE
C
    TR.NOE=0
    DO I=1,TR.NPS
        NTAG=PKD(I,2,NOT)
        IF(NTAG.NE.0)THEN
            DO II=1,NTAG
                IF(PKD(I,II,HLA).LT.0.0)GOTO 1700
                TR.NOE=TR.NOE+1
                IF(TR.NOE.GT.50)THEN
                    TR.NOE=50
                    WRITE(6,9300)
                    GOTO 1725
                ENDIF
                TR.Z(TR.NOE)=PKD(I,II,IZA)
                TR.A(TR.NOE)=PKD(I,II,IAA)/10
                WRITE(TR.ID(TR.NOE),FMT='(A4)')PKD(I,II,ISA)
            IF(TR.ID(TR.NOE)(1:1).EQ.' ')TR.ID(TR.NOE)=TR.ID(TR.NOE)(2:4)
        ENDIF
    ENDDO
C
C SAVE THE NET VALUE AND ERROR FOR PU-239 WHEN WE COME ACRCROSS IT
    IF((TR.Z(TR.NOE).EQ.94).AND.(TR.A(TR.NOE).EQ.239))THEN
        PU239NET=PKD(I,II,NET)
        PU239NETE=PKD(I,II,NETE)
    ENDIF
    TR.MC(TR.NOE)=PKD(I,II,NET)
    TR.ERR(TR.NOE)=PKD(I,II,NETE)
C
C SEE IF THERE ARE ANY OTHER LINES WITH THIS ISOTOPIC TAG. IF SO SET
C ITS HALF LIFE LINE VALUE NEGATIVE, SO A DOUBLE ENTRY WILL NOT BE MADE FOR
C THIS ISOTOPE.

```

```

                IF(I+1.GT.TR.NPS)GOTO 1500
                DO III=I+1,TR.NPS
                    ITA=PKD(III,2,NOT)
                    IF(ITA.EQ.0)GOTO 1500
                    DO J=1,ITA
                        IAT=PKD(III,J,IAA)/10
                        IZT=PKD(III,J,IZA)
                        IF(IAT.NE.TR.A(TR.NOE).OR.
1                      IZT.NE.TR.Z(TR.NOE))GOTO 1400
                        IF(PKD(III,J,ISA).NE.
1                      PKD(I,II,ISA))GOTO 1400
                        PKD(III,J,HLA)=-PKD(III,J,HLA)
                        GOTO 1500
                    ENDDO
                ENDDO
1400
1500
1700
            ENDIF
        ENDDO
C SORT NUCLIDES BY ASSENDING Z
    IF(TR.NOE .LE. 1)GOTO 1750
1725 DO I=1,TR.NOE-1
        IF(TR.Z(i) .GT. TR.Z(I+1))THEN
            IZT=TR.Z(i)
            IAT=TR.A(i)
            SID=TR.ID(i)
            TMC=TR.MC(i)
            TER=TR.ERR(i)

            TR.Z(i)=TR.Z(I+1)
            TR.A(i)=TR.A(I+1)
            TR.ID(i)=TR.ID(I+1)
            TR.MC(i)=TR.MC(I+1)
            TR.ERR(i)=TR.ERR(I+1)

            TR.Z(I+1)=IZT
            TR.A(I+1)=IAT
            TR.ID(I+1)=SID
            TR.MC(I+1)=TMC
            TR.ERR(I+1)=TER

            GOTO 1725
        ENDIF
    ENDDO
1750 IF(TR.NOE .LE. 0)THEN
        WRITE(6,8101)
        GOTO 1900

```

```

ENDIF

C
C GET DATA FROM HEADER BLOCK
C
  READ(1'1)HEADER          !READ HEADER BLOCK
C  DO I = 61, 160          !CHECK FOR END BYTE OF TITLE
C    IF (ICHAR(HEADER(I:I)) .EQ. 0) GOTO 50
C  ENDDO
C  I = 160
C
C BEGIN PRINTING OF SUMMARY SHEETS
C
  WRITE(6,8000)'ACTIVITIES ',curdat

  WRITE(6,8010)HEADER(61:160)

  WRITE(6,8020)HEADER(1:40)

  WRITE(6,8030)SDATE,STM

  WRITE(6,8040)CDATE,CTM

  IXU=INDEX(P.VOLUME_UNITS,' ')-1
  WRITE(6,8050)VHEAD(VOL),P.VOLUME_UNITS(1:IXU),p.sys_volume_units

  WRITE(6,8100)'uCi',P.VOLUME_UNITS(1:IXU)

  DO I=1,TR.NOE
    WRITE(6,8110)TR.Z(i),TR.ID(i),TR.A(i),TR.MC(i),TR.ERR(i)
  ENDDO

C PRINT THE PU-239 MASS = PU-239 NET ACTIVITY CORRECTED FOR ATTENUATION DIVIDED
C BY THE SPECIFIC ACTIVITY FOR PU-239
  IF(PU239NET .EQ. 0.0)THEN
    WRITE(6,9314)
  ELSE
    r_en_3=413.713 !Base attenuation corrections on the icode 1 line.
    call find_line(94,239,'PU',r_en_3)
    pu239m=pu239net*p.volume/ !Convert microCuries/unit to microCuries.
    (log(2.0)*6.023E23/(f1.h1*239.0*3.7e04)) !Divide by the specific activity in uCi/g (log = ln).
    pu239e=pu239m*(pu239nete/100.0)
    cPu239=sigfig(pu239m,pu239e)
    WRITE(6,9316)cPu239
  ENDF

C PRINT THE PU-239 MASS ASSUMING DISTRIBUTED SOURCE

```

```

C      CALL FIND_LINE(94,239,'PU',413.713)
C      IF(FL.ERR .NE. 0.0)THEN
C          WR=P.VOLUME/5400.0
C          IF(P.SYS_VOLUME_UNITS .EQ. 'S')THEN
C              RMU=0.074
C          ELSE
C              RMU=0.0903
C          ENDIF
C          PU239NET=(FL.CPS/FL.EFF)*WR*RMU*1.494
C          TEXP=(1.0-EXP(-WR*RMU))*(FL.BR/100.0)*3.7E+10*0.06204
C          PU239NET=PU239NET/TEXP
C          WRITE(6,9318)PU239NET
C      ELSE
C          WRITE(6,9319)
C      ENDIF

7000  CONTINUE
C
C DPY (XTP) the desired regions.
C
C      do i=1,numxtp
C          call dpy_region(estart(i),eend(i))
C      enddo
C
C DETERMINE IF WE ARE TO SAVE THE RESULTS FROM THIS ANALYSIS. IF WE ARE DO IT.
C      IF(.NOT. AP.SAVE_RESULTS)CALL EXIT
C
C OPEN THE RESULTS FILES. THE NAME OF THE RESULTS FILES WILL BE THE SAME AS
C THE SPECTRUM FILES EXCEPT FOR THE EXTENTION WHICH WILL BE .RES
C      CLOSE(UNIT=1)
C      SP_FILE=' '
C      SP_FILE='BATCH$RESULTS: '//SP_NAME//'.RES'
C      OPEN(UNIT=1,NAME=SP_FILE,TYPE='NEW',FORM='UNFORMATTED')
C      WRITE(1)HEADER      !SAVE THE SPECTRUM HEADER
C      WRITE(1)P           !SAVE SPECTRUM PARAMETER BLOCK
C WRITE INFO TO FILE CLOSE IT AND RETURN
C      WRITE(1)TR
C      write(1)ir          !Isotopic ratios.
C      CALL EXIT
C      END

```

C.2 CBAT_PRGIDXC Subroutine

```

C
C SUBROUTINE TO PRINT THE RESULTS OF THE ANALYSIS ON A LINE BY LINE BASIS

```



```

        STIME=VHEAD(TSD)                !SAMPLE TIME
        WRITE(UNIT=HDR(1:4),FMT='(I4)')STIME
        DO 120 I=1,3
            IF(HDR(I:I).EQ.' ')HDR(I:I)='0'
120     CONTINUE
        WRITE(6,9530)SDATE,HDR(1:4),VHEAD(STD)
9530    FORMAT(' *',128(' '), '**',/, ' * ',
        1 ' SAMPLE COLLECTION (DATE/TIME):',1(' '),
        1 I6, '/',A4,17(' '), 'SOURCE-TO-DETECTOR DISTANCE(CM):',
        2 2(' '),F10.1,22(' '), '**')
        CDATE=VHEAD(DSC)                !COLLECTION DATE
        CTIME=VHEAD(TSC)                !COLLECTION TIME
        WRITE(UNIT=HDR(1:4),FMT='(I4)')CTIME
        DO 130 I=1,3
            IF(HDR(I:I).EQ.' ')HDR(I:I)='0'
130     CONTINUE
        WRITE(6,9540)CDATE,HDR(1:4),VHEAD(ECF)
9540    FORMAT(' *',128(' '), '**',/, ' * ',
        1 ' SAMPLE COUNTED (DATE/TIME):',4(' '),
        1 I6, '/',A4,17(' '), 'EFF. MULT. FACTOR:',
        2 12(' '),F8.4,28(' '), '**')
C DECAY IS SAMPLE DECAY TIME IN HOURS
        DECAY=VHEAD(DCY)/3600.0
        PDEAD=((VHEAD(RLT)-VHEAD(LVT))/VHEAD(RLT))*100.0
        WRITE(6,9550)DECAY,PDEAD
9550    FORMAT(' *',128(' '), '**',/, ' * ',
        1 ' DECAY TIME (HRS):',11(' '),F10.2,
        1 21(' '), 'DET. DEAD TIME(%)',F18.2,30(' '), '**')
        IRLT=VHEAD(RLT)
        ILVT=VHEAD(LVT)
        HDR(1:13)=' '
        WRITE(UNIT=HDR(30:35),FMT='(I6)')IRLT
        WRITE(UNIT=HDR(40:45),FMT='(I6)')ILVT
        K=1
        DO 200 I=30,35
            IF(HDR(I:I).NE.' ')THEN
                HDR(K:K)=HDR(I:I)
                K=K+1
            ENDIF
200     CONTINUE
        HDR(K:K)='/'
        K=K+1
        DO 220 I=40,45
            IF(HDR(I:I).NE.' ')THEN
                HDR(K:K)=HDR(I:I)
                K=K+1
            ENDIF

```

```

220  CONTINUE
    WRITE(6,9560)HDR(1:13)
9560  FORMAT(' *',128(' '), '**',/, ' * ',
1 ' COUNTING TIMES (REAL/LIVE SEC.): ',A13,
1 79X,'**')
C
C COMPUTE TOTAL COUNTS IN SPECTRUM
    TOTAL=0.0D0
    INC=VHEAD(NC)
    DO 240 I=64,INC-1
240  TOTAL=TOTAL+FGET(I)
    IF(VHEAD(LVT).NE.0.0)THEN
        TOTAL=TOTAL/VHEAD(LVT)
    ENDIF
    HDR(1:60)=' '
    WRITE(UNIT=HDR(30:36),FMT='(F7.3)')VHEAD(RSC)
    WRITE(UNIT=HDR(40:49),FMT='(G10.4)')TOTAL
    K=1
    DO 260 I=30,36
        IF(HDR(I:I).NE.' ')THEN
            HDR(K:K)=HDR(I:I)
            K=K+1
        ENDIF
260  CONTINUE
    HDR(K:K)='/'
    K=K+1
    DO 280 I=40,49
        IF(HDR(I:I).NE.' ')THEN
            HDR(K:K)=HDR(I:I)
            K=K+1
        ENDIF
280  CONTINUE
    WRITE(6,9570)VHEAD(VOL),HDR(1:25)
9570  FORMAT(' *',128(' '), '**',/, ' * ',
1 ' SAMPLE QUANTITY:',10(' '),1PE13.3,21(' '),
1 ' RANDOM SUM CORRECTION/COUNT RATE: ',A25,7X,'**')
    I=VHEAD(ETB)
    J=VHEAD(LTB)
    HDR(1:9)=' '
    WRITE(UNIT=HDR(30:33),FMT='(I4)')I
    WRITE(UNIT=HDR(40:43),FMT='(I4)')J
    K=1
    DO 300 I=30,33
        IF(HDR(I:I).NE.' ')THEN
            HDR(K:K)=HDR(I:I)
            K=K+1
        ENDIF

```

```

300    CONTINUE
      HDR(K:K)='/'
      K=K+1
      DO 320 I=40,43
        IF(HDR(I:I).NE.' ')THEN
          HDR(K:K)=HDR(I:I)
          K=K+1
        ENDIF
320    CONTINUE
      WRITE(6,9580)P.MCA_SYSTEM,HDR(1:9)
9580   FORMAT(' *',128(' '), '**',/, ' * ',
1      ' DETECTOR SYSTEM:',17(' '),A4,
1      22(' '), 'EFFICIENCY/LINEARITY TABLES:',3(' '),A9,26(' '), '**')
C
C PACK NAMES OF LIBRARYS
      K=INDEX(ILIB,' ')-1
      J=INDEX(LLIB,' ')-1
      I=INDEX(ITLIB,' ')-1
      IF(K.LE.0)K=10
      IF(J.LE.0)J=10
      IF(I.LE.0)I=10
      HDR(1:33)=ILIB(1:K)///'///LLIB(1:J)///'///ITLIB(1:I)
C
      WRITE(6,9590)HDR(1:33)
9590   FORMAT(' *',128(' '), '**',/, ' * ',
1      ' LIBRARIES USED: ',4(' '),A33,72X,'**')
C
C PRINT THE PULSER CALIBRATION INFO
      CALL PR_PULSER_CAL
C
C PRINT THE BACKGROUNDS WHICH WERE AVERAGED
      HDR(1:130)=' '
      HDR(1:1)='*'
      HDR(130:130)='*'
      WRITE(6,9595)HDR(1:130)
      HDR(53:76)='--BACKGROUNDS AVERAGED--'
      WRITE(6,9595)HDR(1:130)
      K=0
      HDR(53:76)=' '
      IF(.NOT. BKGC)GOTO 1000
      DO I=1,BK.NO_BKG
        J=INDEX(BK.SP_ID(I),':')
        L=(K*15)+2
        HDR(L:L+10)=BK.SP_ID(I)(J+1:J+11)
        K=K+1
        IF(K.EQ.6)THEN
          WRITE(6,9595)HDR(1:130)

```

```

        K=0
        HDR(2:129)=' '
    ENDIF
    ENDDO
    WRITE(6,9595)HDR(1:130)
9595  FORMAT(1X,A130)
C
1000  WRITE(6,9600)
9600  FORMAT(' *',128(' '), '**',/, ' *',128(' '), '**',/, ' *',128(' '), '**',
1 /, ' ',130(' '))
C
    IF(PRES)CALL CBATGAUSS_OUT
    RETURN
    END

```

C.3 DPY_CBAT_GAUSS Subroutine

```

C ROUTINE MODIFIED TO USE THE FUNCTION "F_WIDTH" FOR PEAK WIDTH DETERMINATION
C IN THE BACKGROUND SPECTRA
C      EWK 1/15/91
C
C ROUTINE TO FIT PEAK FOUND BY THE SEARCH ROUTINE. THIS SUBROUTINE IS
C A NON-LINEAR FIT ROUTINE TO BE USED BY THE CBAT ROUTINES
C      EWK 4/1/88
C
C MODIFIED TO LOOK AT RESULTS OF MULTIPLY FITS IF RESULTING CENTRIDS
C ARE SEPERATED BY LESS THAN 1 KEV ELIMINATE THE PEAK IN THE FIT THAT
C MOVED THE MOST AND REFIT.
C      EWK 11/21/88
C
C Exponential overflow check commented out.  D. A. Femec, February 1, 1994.
C
    SUBROUTINE DPY_CBAT_GAUSS
C
    INCLUDE 'GAP$SRC:BACKGROUND_RESULTS.INC'
    INCLUDE 'GAP$SRC:VHEAD.INC'
    INCLUDE 'GAP$SRC:PKD.INC'
    INCLUDE 'GAP$SRC:BACK_FIT.INC'
    INCLUDE 'GAP$SRC:PLOT.INC'
C
    DOUBLE PRECISION AMATR,PVEC,DFT,BVEC,WTDEV,TOT
    REAL*4      LEAST_VAL,LL,E(5)
    DIMENSION MULST(5),
1  WDT(5),YVAL(5),XVAL(5),
2  AMATR(15,15),BVEC(15),PVEC(15),

```

```

3  XERR(5),AERR(5),WERR(5),
4  AREA(5)
   DIMENSION WDATA(300),WEIT(300)
   DATA NU,NSP,NE,NCC,NL/3,0,2,1,4/
   DATA FF1/.75/,FF2/.75/,FF3/5.0/,FF4/5.0/,FF5/0.85/,
1  FF6/1.25/,FF7/2.5/,NPS/300/,NMS/5/,IDTS/1/,ICBS/33/,ISRS/0/

C
   NCS=VHEAD(NC)           !NUMBER OF CHANNELS
   NPF=VHEAD(NP)           !NUMBER OF PEAKS TO FIT
   BP.LINEAR_FIT=.FALSE.   !INFORM BACKGROUND FIT PROCESS THAT THESE
                           !RESULTS ARE FROM A NONLINEAR FIT
C COMPUTE THE WIDTHS OF ALL PEAKS TO WHICH A FIT IS TO BE MADE
   DO I=1,NPF
       PKD(I,1,WV)=F_WIDTH(PKD(I,1,XV))
   ENDDO
   IR=0

C
C START FIT LOOP BY CHECKING FOR GOOD SEARCH PARAMETERS
C
   II=0
100 II=II+1
110 IF (II.GT.NPS) RETURN   ! CAN HANDLE AT MOST 'NPS' PEAKS
   IF(II.GT.NPF)RETURN      ! DONE ALL PEAKS IN LIST
   IF(PKD(II,1,XV).LT.40) GO TO 100   ! IGNORE PEAKS IN 1ST 40 CHANNELS

C
C FIND PEAK(S) FOR FIT REGION
C FILL ARRAY 'MULST' WITH THE INDEX NUMBERS OF THE PEAK TO BE FIT IN
C THIS SETUP
   DO IGAUS=1,NMS
       MULST(IGAUS)=II
       IF(II+1.GT.NPF)GOTO 103 !NO OTHER PEAKS LEFT TO CHECK FOR
                           !MULTIPEL FIT
       IRANGE=PKD(II+1,1,XV)-(((PKD(II,1,WV))*FF7-0.5)+PKD(II,1,XV))
       IF(IRANGE.GT.0)GO TO 103   ! FAR ENOUGH APART
       IF(PKD(MULST(1),1,XV)+40-PKD(II+1,1,XV).LE.0)GO TO 103
       II=II+1
   ENDDO
   II=II-1
   IGAUS=IGAUS-1

C
C FIND FIRST CHANNEL OF FIT REGION
C
C*****
103 IF (VHEAD(DPA).EQ.1.0)WRITE(6,1001)(PKD(MULST(KK),1,XV),KK=1,IGAUS)
1001 FORMAT(///,1X,25(' '),//,' FIT CHANNELS(S) =',5F6.1)
C*****
   IT=PKD(MULST(1),1,WV)*FF1+0.5

```

```

      IF(IT.LE.0) IT=1
      IFC=PKD(MULST(1),1,XV)-IT
      IF(IFC.LT.3) IFC=3
C FIND LAST CHANNEL OF FIT REGION
      IT=PKD(MULST(IGAUS),1,WV)*FF2+1.7
      IF(IT.LE.0) IT=1
      LC=PKD(MULST(IGAUS),1,XV)+IT
      IF(LC.GT.VHEAD(NC)-2.0) THEN
        IF(IGAUS.GT.1) THEN
          IGAUS=IGAUS-1
          II=II-1
          GOTO 103
        ELSE
          NPP=VHEAD(NP)
          VHEAD(NP)=VHEAD(NP)-(NPP-MULST(IGAUS)+1)
          RETURN
        ENDIF
      ENDIF
C FIND LOWER BACKGROUND CHANNEL LIMIT
      J=MULST(1) !INDEX OF FIRST PEAK IN THE FIT REGION
      FIVE_WIDTH=5.0*PKD(J,1,WV)
      IF(J.EQ.1) THEN
        IBEG=PKD(J,1,XV)-FIVE_WIDTH
        IF(IBEG.LT.2) IBEG=2
        IF(IBEG.GE.IFC) IBEG=IFC-1
      ELSE
        ITW=PKD(J-1,1,XV)+PKD(J-1,1,WV)
        IM5=PKD(J,1,XV)-FIVE_WIDTH
        IF(ITW.GT.IM5) THEN
          IBEG=ITW-1
        ELSE
          IBEG=IM5
        ENDIF
        IF(IBEG.GT.IFC) IBEG=IFC-1
      ENDIF
C FIND UPPER BACKGROUND CHANNEL LIMIT
      J=MULST(IGAUS) !THE INEX FOR THE LAST PEAK IN THE FIT REGION
      FIVE_WIDTH=5.0*PKD(J,1,WV)
      IF(J.EQ.NPF) THEN
        IEND=PKD(J,1,XV)+FIVE_WIDTH
        IF(IEND.GE.NCS-1) THEN
          IEND=NCS-1
          IF(IEND.LT.LC) THEN
            IEND=LC+1
          ENDIF
        ENDIF
      ELSE

```

```

        ITW=PKD(J+1,1,XV)-PKD(J+1,1,WV)
        IM5=PKD(J,1,XV)+5.0*PKD(J,1,WV)
        IF(IM5.GT.ITW)THEN
            IEND=ITW
        ELSE
            IEND=IM5
        ENDIF
        IF(IEND.LT.LC) IEND=LC+1
    ENDIF
C LOAD FIT DATA INTO ARRAY WDATA
    IF((IEND-IBEG).GT.300)THEN
        TYPE *, ' GAUSS-ERROR FIT REGION FOR PEAK AT ',
        1 PKD(MULST(1),1,XV), ' IS TO LARGE. GREATER THAN 300 CHANNELS'
        CALL EXIT
    ENDIF
    K=1
    DO I=IBEG, IEND
        WDATA(K)=FGET(I)
        K=K+1
    ENDDO

C
C LOAD WEIGHT VALUES FOR DATA IN REGION OF FIT INTO ARRAY WEIT
    K=1
    DO I=IFC, LC
        WEIT(K)=WEIGHT(I, WDATA(K))
        K=K+1
    ENDDO

C
    IE=LC-IFC+1
    LW08=IFC-IBEG

C
C *****TEMP*****
    IF (VHEAD(DPA).EQ.1.0) WRITE (6,1002) IBEG, IFC, LC, IEND
1002 FORMAT (/, ' IBEG, IFC, LC, IEND = ', 4I6, /, 1X)
    KA=IEND-IBEG+1
C *****
C
C CALCULTE THE COORDINATES OF BACKGROUND LINE
    CALL REGION_BKG_LINE
C
C
C CALCULATE BACKGROUND FUNCTION
40000     BKSLOP=(YRMIN-YLMIN)/(XRMIN-XLMIN)
        YMAX=0.0
        YMIN=999999999.0
C SET INITIAL APPROXIMATION FOR CENTROID, CENTROID HEIGHT,
C AND GAUSS WIDTH FOR EACH PEAK IN THE FIT REGION

```

```

DO 117 I=1, IGAUS
ITEMP=PKD(MULST(I), 1, XV)+0.5
LPEK=ITEMP+1-IBEG
AY=WDATA(LPEK-1)
BY=WDATA(LPEK)
CY=WDATA(LPEK+1)
IF (AY+BY+CY.EQ.0.0) THEN
    NAFLG=NE
    GOTO 301
ENDIF
FT=(CY-AY)/(AY+BY+CY)
XVAL(I)=PKD(MULST(I), 1, XV)+FT
WDTH(I)=PKD(MULST(I), 1, WV)
C FORMULA DERIVED FROM LAGRANGIAN POLYNOMIAL WITH EQUIDISTANT
C ABSISSAS
Y=(( -2.0*BY+AY+CY)*FT+(CY-AY))/2.0*FT+BY
YY=Y-((XVAL(I)-XLMIN)*BKSLOP+YLMIN)
IF (YY.GE.1.0) Y=YY
YVAL(I)=Y
IF (Y.LT.YMIN) YMIN=Y
117 IF (Y.GT.YMAX) YMAX=Y
C SUBTRACT BACKGROUND FROM FIT REGION VALUES
FC=IFC
DO 115 I=1, IE
WDATA(LW08+I)=WDATA(LW08+I)-((FC-XLMIN)*BKSLOP+YLMIN)
115 FC=FC+1.0
C
C PRINT BACKGROUND INFO IF REQUESTED
C
IF (VHEAD(DPA).EQ.1.0) THEN
    WRITE(6, 9000) XLMIN, YLMIN, XRMIN, YRMIN
9000 FORMAT(' LEFT BKG CH= ', F7.0, ' LEFT BKG VAL= ', F12.0
1 ' RIGHT BKG CH= ', F7.0, ' RIGHT BKG VAL= ', F12.0)
    WRITE(6, 9100) (WDATA(LW08+J), J=1, IE)
9100 FORMAT(' CHANNEL VALUES AFTER BACKGROUND SUBTRACTION
1 (STARTING AT LEFT FIT CHANNEL', (/ , 10F12.0))
ENDIF
XMIN=IFC
XMAX=LC
YMIN=YMIN/8.0
YMAX=YMAX*8.0
WMIN=WDTH(IGAUS)/10.0
WMAX=WMIN*100.0
MDIM=2*IGAUS+1
NOIT=MDIM*3+1
C *****TEMP*****
IF (VHEAD(DPA).EQ.0.0) GO TO 1016

```



```

      DO 1015 I=1, IGAUS
1015 WRITE(6,1003)XVAL(I),YVAL(I),WDTH(I)
1003 FORMAT (' XVAL=',F10.2,5X,'YVAL=',F15.2,5X,'WDTH=',F10.2)
1016 CONTINUE
C *****
C END OF GAUSS SETUP
C START OF GAUSS FITTING. DO FOR MAX ITERATIONS (NOIT) OR
C TIL CONVERGENCE IS MADE (WHICHEVER OCCURS FIRST).
      IPCFL=0
      NMS3=(2*NMS)+1
      JJ=0
200 JJ=JJ+1
      IF (JJ.GT.NOIT) GO TO 250
      DO 201 I=1,NMS3
      BVEC(I)=0.0
      DO 201 J=1,NMS3
201 AMATR(I,J)=0.0
      FC=IFC
C DO LEAST SQUARES FIT OVER REGION FOR EACH PEAK
      DO 230 JI=1,IE
      GSUM=0.0
C FOR MULTIPLETS THE WIDTHS ARE FIXED
C JUST ONE ENTRY IN PVEC FOR WIDTH
      I3=1
      PVEC(3)=0.0
      DO 221 I=1, IGAUS
      FT=(FC-XVAL(I))/WDTH(I)
      ET=FT*FT*(-2.77258873)
      DFDH=EXP(ET)
      IF(DFDH.EQ. 0)GOTO 290 !EXPONENT OVERFLOW
      PVEC(I3)=DFDH
      CNVC=YVAL(I)*DFDH
      GSUM=GSUM+CNVC
      DFDC=(CNVC*5.54517746*FT)/WDTH(I)
      PVEC(I3+1)=DFDC
C DFDW=DFDC*FT
      PVEC(3)=PVEC(3)+DFDC*FT
      I3=I3+2
      IF(I3.EQ.3) I3=I3+1
221 CONTINUE
      DFT=WEIT(JI)
      WTDEV=(WDATA(LW08+JI)-GSUM)*DFT
      DO 225 I=1,MDIM
      BVEC(I)=PVEC(I)*WTDEV+BVEC(I)
      DO 225 J=1,MDIM
225 AMATR(I,J)=AMATR(I,J)+PVEC(I)*PVEC(J)*DFT
230 FC=FC+1.0

```

```

        IF(IPCFL.EQ.1)GO TO 227
        CALL INVTB(MDIM,AMATR,BVEC,PVEC,15)
        GO TO 228
227 CALL INVT(MDIM,AMATR,BVEC,PVEC,15)
C COMPUTE NEW VALUES AND TEST FOR CONVERGENCE
228 IC=0
        I3=1
        DO 245 I=1, IGAUS
        DFT=DABS(PVEC(I3))
        IF(DFT.LT.1.0)GO TO 232
        FT=PVEC(I3)/ABS(YVAL(I))- .005
        IF(FT.LT.0.0)GO TO 232
        IC=IC+1
        FT=ABS(YVAL(I))/2.0
        IF(DFT.GT.FT)PVEC(I3)=FT
232 FT=PVEC(I3)+YVAL(I)
        IF(FT.GE.YMAX)GO TO 290
        IF(FT.LT.YMIN)GO TO 290
        YVAL(I)=FT
        DFT=DABS(PVEC(I3+1))
        IF(DFT.GE.0.005)IC=IC+1
        IF(DFT.GT.2.0)PVEC(I3+1)=2.0
        FT=PVEC(I3+1)+XVAL(I)
        IF(FT.GT.XMAX)GO TO 290
        IF(FT.LT.XMIN)GO TO 290
        XVAL(I)=FT
        DFT=DABS(PVEC(3))
        IF(DFT.GE.0.005)IC=IC+1
        FT=ABS(WDTH(I))/2.0
        IF(DFT.GT.FT)PVEC(3)=FT
        FT=PVEC(3)+WDTH(I)
        IF(FT.GT.WMAX)GO TO 290
        IF(FT.LT.WMIN)GO TO 290
        I3=I3+2
        IF(I3.EQ.3)I3=I3+1
245 WDTH(I)=FT
C *****TEMP*****
        IF(VHEAD(DPA).EQ.0.0)GO TO 1017
        DO 1005 I=1, IGAUS
1005 WRITE(6,1003)XVAL(I),YVAL(I),WDTH(I)
1017 CONTINUE
C *****
        IF(IC.NE.0)GO TO 200
        IF(IPCFL.NE.0)GO TO 300
        IPCFL=1
        JJ=NOIT-1
        GO TO 200

```

```

250 IF(IPCFL.NE.0)GO TO 300
    NAFLG=NCC
    GO TO 301
300 NAFLG=NSP
C COMPUTE ESTIMATED ERRORS
301 FC=IFC
C *****TEMP*****
    IF(VHEAD(DPA).EQ.1.0)WRITE(6,1006)NAFLG
1006 FORMAT(/,' NAFLG = ',I2)
C *****
C TEST FOR NORMAL FIT.
    IF (NAFLG.NE.0) GO TO 500
    TOTDEV=0.0
    DO 310 JI=1,IE
        GSUM=0.0
        DO 305 I=1,IGAUS
            FT=(FC-XVAL(I))/WDTH(I)
            ET=FT*FT*(-2.77258873)
305 GSUM=GSUM+YVAL(I)*EXP(ET)
            FT=GSUM-WDATA(LW08+JI)
            TOTDEV=TOTDEV+FT*FT*WEIT(JI)
310 FC=FC+1.0
            FT=IE-MDIM
            SCHISQ=SQRT(TOTDEV/FT)
            IF(SCHISQ.LT.1.0)SCHISQ=1.0
            I3=1
            WERR(1)=SQRT(AMATR(3,3))*SCHISQ
            DO 315 IA=1,IGAUS
                XERR(IA)=SQRT(AMATR(I3+1,I3+1))*SCHISQ
                WERR(IA)=WERR(1)
                FA=WDTH(IA)*YVAL(IA)
                AREA(IA)=FA*1.06446
                FT2=WDTH(IA)**2*AMATR(I3,I3)
                FT3=YVAL(IA)**2*AMATR(3,3)
                FT=FA*AMATR(I3,3)*2.0+FT2+FT3
                AERR(IA)=((SQRT(FT)*SCHISQ)/FA)*100.0
                I3=I3+2
                IF(I3.EQ.3)I3=4
315 CONTINUE
            GO TO 350
C
290 NAFLG=NU
    IF (IGAUS.EQ.1) NAFLG=NL
    GO TO 301
C BAD PEAK(S) - VALUES TO PKD ARRAY.
500 DO 510 I=1,IGAUS
    J=PKD(MULST(I),1,XV) !FOR PEAKS UNABLE TO FIT COMPUTE CENTRIOD

```

```

                                !BASED ON CONTENTS OF THREE CHANNELS AROUND
                                !INITIAL PEAK POSITION
      AY=FGET(J-1)
      BY=FGET(J)
      CY=FGET(J+1)
      IF (AY+BY+CY.EQ.0.0) THEN
        CHAN=PKD(MULST(I),1,XV)
      ELSE
        CHAN=(CY-AY)/(AY+BY+CY)+PKD(MULST(I),1,XV)
      ENDIF
      PKD(MULST(I),1,WV)=0.0
      PKD(MULST(I),1,AV)=0.0
      PKD(MULST(I),2,DW)=0.0
      PKD(MULST(I),2,DA)=0.0
      PKD(MULST(I),2,NOT)=0.0
      PKD(MULST(I),3,FLG)=NAFLG
      FCHNL=CHAN+VHEAD(CZ)
      RCHNL=FCHNL
      ENERGY=(RCHNL*VHEAD(CE)+VHEAD(BE))*RCHNL+VHEAD(AE)
      PKD(MULST(I),3,ERA)=ENERGY
      PKD(MULST(I),1,DER)=0.0
      PKD(MULST(I),1,XV)=CHAN
510  CONTINUE
      GO TO 100
C
C GOOD FIT SAVE RESULTS
C
350  IF (IGAUS.GT.1) THEN
      DO I=1, IGAUS
        X=XVAL(I)+VHEAD(CZ)
        E(I)=(X*VHEAD(CE)+VHEAD(BE))*X+VHEAD(AE)
      ENDDO
C
C CHECK FOR PEAK SEPERATION
      DO I=1, IGAUS-1
        IF (ABS(E(I)-E(I+1)).LT.1.0) THEN
          IF ((ABS(PKD(MULST(I),1,XV)-XVAL(I))).GT.
            1      (ABS(PKD(MULST(I+1),1,XV)-XVAL(I+1)))) THEN
C
C ELEMINATE PEAK AT POSITION POINTED TO MULST(I)
          DO J=MULST(I),NPF
            PKD(J,1,XV)=PKD(J+1,1,XV)
          ENDDO
          NPF=NPF-1
          VHEAD(NP)=VHEAD(NP)-1.0
          II=MULST(1)
          GOTO 110

```

```

ELSE
C
C ELEMINATE PEAK AT POSITION POINTED TO BY MULST(I+1)
      DO J=MULST(I+1),NPF
        PKD(J,1,XV)=PKD(J+1,1,XV)
      ENDDO
      NPF=NPF-1
      VHEAD(NP)=VHEAD(NP)-1.0
      II=MULST(1)
      GOTO 110
    ENDIF
  ENDIF
ENDDO
NAFLG=2
ENDIF
DO I=1,IGAUS
  FCHNL=XVAL(I)+VHEAD(CZ)
  ENERGY=(FCHNL*VHEAD(CE)+VHEAD(BE))*FCHNL+VHEAD(AE)
C SAVE RESULTS IN PEAK DATA ARRAY
  PKD(MULST(I),1,XV)=XVAL(I)
  PKD(MULST(I),1,WV)=WDTH(I)
  PKD(MULST(I),1,AV)=AREA(I)
  PKD(MULST(I),1,DER)=XERR(I)
  PKD(MULST(I),2,DW)=WERR(I)
  PKD(MULST(I),2,DA)=AERR(I)
  PKD(MULST(I),2,NOT)=0.0
  PKD(MULST(I),3,FLG)=NAFLG
  PKD(MULST(I),3,ERA)=ENERGY
  PKD(MULST(I),3,CHI)=SCHISQ
  BK.CHANNEL(MULST(I))=XVAL(I) !SAVE THIS PEAKS CENTROID
C WAS FIT A SINGLET? IF IT IS HANDLE THE BACKGROUND FIT A LITTLE DIFFERENTLY
  IF(IGAUS.EQ.1)THEN
C NO FIT EACH BACKGROUND SPECTRUM TO THE REGIONS DETERMINED BY THE
C THE BACKGROUND CHANNEL NUMBERS
    X=XLMIN+VHEAD(CZ)
    ENERGY_FC=(X*VHEAD(CE)+VHEAD(BE))*X+VHEAD(AE)
    X=XRMIN+VHEAD(CZ)
    ENERGY_LC=(X*VHEAD(CE)+VHEAD(BE))*X+VHEAD(AE)
    DO J=1,BK.NO_BKG
      BF.E(J).NO_PEAKS=IGAUS !NUMBER OF PEAKS IN BACKGROUND FIT
      BF.E(J).EXC(1)=ENERGY
      BF.E(J).EFC=ENERGY_FC
      BF.E(J).ELC=ENERGY_LC
      BF.E(J).AREA(1)=WDTH(1) !USE CALCULATE PEAK WIDTH AS
        !THE WIDTH OF THE BACKGROUND PEAK
      CALL FIT_BKG(J)
    
```

C

```

C SAVE RESULTS OF FIT FOR THIS PEAK. FIRST CHANGE AREA TO CPS AND RANDOM
C SUM CORRECT IT
C PULSERED SPECTRUM?
      IF(.NOT.BF.E(J).PULSER)THEN
        BK.B_CPS(MULST(1),J)=BF.E(J).AREA(1)/(BF.E(J).LIVE)
      ELSE
C
C PULSERED SPECTRUM, BUT DO WE HAVE VALID PARAMETERS IF NOT BF.E(J).SCA=1.0
C AND BF.E(J).SCB=0.0
      IF(BF.E(J).SCA.EQ.1.0.AND.BF.E(J).SCB.EQ.0.0)THEN
        BK.B_CPS(MULST(1),J)=BF.E(J).AREA(1)/BF.E(J).LIVE
      ELSE
        IF(BK.CHANNEL(MULST(1)).LE.3600.0)THEN
          BK.B_CPS(MULST(1),J)=(BF.E(J).AREA(1)/
1          ((BF.E(J).SCB-BF.E(J).SCA)/3600.0)*
2          BK.CHANNEL(MULST(1))+
2          BF.E(J).SCA))*BF.E(J).RATE
        ELSE
          BK.B_CPS(MULST(1),J)=(BF.E(J).AREA(1)/BF.E(J).SCB)*
1          BF.E(J).RATE
        ENDIF
      ENDIF
      ENDIF
      BK.B_ERROR(MULST(1),J)=
1      ABS((BF.E(J).A_ERR(1)/BF.E(J).AREA(1)))*100.0
      ENDDO
    ENDIF
  ENDDO
C
C IF FIT WAS TO 2 OR MORE PEAKS THEN THE BACKGROUND MUST BE FIT IN THE SAME
C MANNER
  IF(IGAUS.GT.1)THEN
    X=XLMIN+VHEAD(CZ)
    ENERGY_FC=(X*VHEAD(CE)+VHEAD(BE))*X+VHEAD(AE)
    X=XRMIN+VHEAD(CZ)
    ENERGY_LC=(X*VHEAD(CE)+VHEAD(BE))*X+VHEAD(AE)
    DO J=1,BK.NO_BKG
      BF.E(J).NO_PEAKS=IGAUS !NUMBER OF PEAKS IN BACKGROUND FIT
      BF.E(J).EFC=ENERGY_FC
      BF.E(J).ELC=ENERGY_LC
      DO I=1,IGAUS
        BF.E(J).EXC(I)=PKD(MULST(I),3,ERA)
        BF.E(J).AREA(I)=PKD(MULST(I),1,WV)
      ENDDO
    ENDDO
  C FIT THE PEAKS IN THE BACKGROUND
  CALL FIT_BKG(J)
  DO I=1,IGAUS

```

```

C SAVE RESULTS OF FIT FOR THIS PEAK. FIRST CHANGE AREA TO CPS AND RANDOM
C SUM CORRECT IT
C PULSERED SPECTRUM?
      IF(.NOT.BF.E(J).PULSER)THEN
        BK.B_CPS(MULST(I),J)=BF.E(J).AREA(I)/(BF.E(J).LIVE)
      ELSE
C
C PULSERED SPECTRUM, BUT DO WE HAVE VALID PARAMETERS IF NOT BF.E(J).SCA=1.0
C AND BF.E(J).SCB=0.0
        IF(BF.E(J).SCA.EQ.1.0.AND.BF.E(J).SCB.EQ.0.0)THEN
          BK.B_CPS(MULST(I),J)=BF.E(J).AREA(I)/BF.E(J).LIVE
        ELSE
          IF(BK.CHANNEL(MULST(I)).LE.3600.0)THEN
            BK.B_CPS(MULST(I),J)=(BF.E(J).AREA(I)/
1            ((BF.E(J).SCB-BF.E(J).SCA)/3600.0)*
2            BK.CHANNEL(MULST(I))+
2            BF.E(J).SCA))*BF.E(J).RATE
          ELSE
            BK.B_CPS(MULST(I),J)=(BF.E(J).AREA(I)/BF.E(J).SCB)*
1            BF.E(J).RATE
          ENDIF
        ENDIF
      ENDIF
      BK.B_ERROR(MULST(I),J)=
1      ABS((BF.E(J).A_ERR(I)/BF.E(J).AREA(I)))*100.0
      ENDDO
    ENDIF
  C
  C SAVE THE RESULTS OF THE FIT FOR LATER DISPLAY
  DO I=1,IGAUS
    PT(MULST(I)).EVSC=(IFC-IBEG)/2+IBEG
    PT(MULST(I)).ENDC=(IEND-LC)/2+LC
900    PT(MULST(I)).EVNC=(PT(MULST(I)).ENDC-PT(MULST(I)).EVSC)+1
    IF((PT(MULST(I)).EVNC*4).GT. 256)THEN
      PT(MULST(I)).EVSC=PT(MULST(I)).EVSC+1
      PT(MULST(I)).ENDC=PT(MULST(I)).ENDC-1
      GOTO 900
    ENDIF
    PT(MULST(I)).ENNF=2          !NUMBER OF FUNCTIONAL VALUES=
                                !GAUSSIAN + BACKGROUND
  C
  C FILL FIRST FUNCTION ARRAY WITH BACKGROUND VALUES
  JEV=1
  DO IEV=PT(MULST(I)).EVSC,PT(MULST(I)).ENDC
    PT(MULST(I)).F(1,JEV)=(IEV-XLMIN)*BKSLOP+YLMIN
    JEV=JEV+1
  
```

```

        ENDDO
        PT(MULST(I)).NPF(1)=PT(MULST(I)).EVNC      !NUMBER OF BACKGROUND FUNCTIONAL DATA POINTS
        !IS JUST THE NUMBER OF CHANNELS
C
C DO GAUSSIAN COMPONENTS
        X=PT(MULST(I)).EVSC
        KEV=1
        DO IEV=PT(MULST(I)).EVSC, PT(MULST(I)).ENDC-1
            DO IIEV=1, 4
                BX=(X-XLMIN)*BKSLOP+YLMIN
                PT(MULST(I)).F(2,KEV)=BX+YVAL(I)*EXP(-2.77258873*
1 ((X-XVAL(I))/WDTH(I))**2)
                KEV=KEV+1
                IF(KEV.EQ. 257)GOTO 1000
                X=X+0.25
            ENDDO
        ENDDO
1000    PT(MULST(I)).NPF(2)=KEV-1      !NUMBER OF POINTS IN EACH FUNCTION ARRAY
        ENDDO
        GO TO 100
    END

```

C.4 DPY_CBAT_LINFIT Subroutine

```

C ROUTINE MODIFIED TO USE THE FUNCTION "B_WIDTH" FOR PEAK WIDTH DETERMINATION
C IN THE BACKGROUND SPECTRA
C     EWK 1/15/91
C
C THE FUNCTION OF THIS ROUTINE IS TO FIT THE PEAKS WHICH WERE NOT FIT
C BY GAUSS PLUS ADD THE PEAKS IN THE LIMIT LIBRARY.
C THE FUNCTION LIB_CHECK (BELOW) IS CALLED TO SEE IF TWO LINES WILL BE TAGGED
C AS THE SAME ISOTOPE.
C     THIS ROUTINE DOESNT COMPUTE "UPPER LIMIT" VALUES AREAS ARE LEFT AS IS
C     EWK 7/7/87
C MODIFIED TO DO NOTHING DIFFERENT FOR THOSE FIT REGIONS WHICH CONTAIN ZEROS
C IF THE WHOLE REGION CONTAINS ALL ZEROS THE AREA=1 AND DA=1
C     EWK 1/27/88
C
C ROUTINE MODIFIED TO PERFORM THE CONCURRENT BACKGROUND ANALYSIS FUNCTION
C ROUTINE IS BASED ON THE NORMAL LINEAR FIT ROUTINE D_FIT
C     EWK 5/9/88
C
C SUBROUTINE DPY_CBAT_LINFIT
C
C INCLUDE 'GAP$SRC:VHEAD.INC'

```



```

INCLUDE 'GAP$SRC:PKD.INC'
INCLUDE 'GAP$SRC:BACKGROUND_RESULTS.INC'
INCLUDE 'GAP$SRC:BACK_FIT.INC'
INCLUDE 'GAP$SRC:PLOT.INC'
RECORD /PLOT/PT1,PT2

C
REAL*4 LMCP,LMWD
REAL*8      T1,T2,T3,T4
CHARACTER*1 ANS
CHARACTER*40 CTMP
LOGICAL SAME
DOUBLE PRECISION AMATR,PVEC,DFT,BVEC,WTDEV,TOT
DIMENSION IFIT(24),XLIM(128),CHM(2),WDM(2),
2  AMATR(3,3),BVEC(3),PVEC(3)
DIMENSION WDATA(300),WEIT(300)
DATA FF1/.75/,FF2/.75/,FF3/5.0/,FF4/5.0/,FF5/.85/,
1  FF6/1.25/,FF7/2.5/,NMS/1/,IDTS/1/,ICBS/33/,ISRS/0/

C
C PROGRAM START
C
      LINM=VHEAD(LTB)      !GET LINEARITY TABLE NO
      TL=VHEAD(LVT)       !GET LIVE TIME
      IRDC=0
C BEGIN THE LIMIT PROCESS BY INSERTING ALL THE LINES IN THE LIMIT LIBRARY
C INTO THE PEAK ANALYSIS RESULTS ARRAY 'PKD'. INITIALLY WE WILL CONVERT THE
C ENERGY VALUES IN THE LIMIT LIBRARY TO CHANNEL POSITIONS AND INSERTING
C THESE CHANNEL POSITIONS INTO THE ARRAY AFTER THOSE VALUES WHICH HAVE
C BEEN PLACED INTO THE ARRAY BY "GAUSS"
      IR=0 !IR=BLOCK NUMBER OF LIMIT LIBRARY
      NPF=VHEAD(NP)      !NPF IS THE NUMBER OF PEAKS IN ARRAY
100  IR=IR+1
      READ(3'IR)XLIM      !READ BLOCK OF LIMIT LIBRARY

C
C CALCULATE MAXIMUM WIDTH EXPECTED IN SPECTRUM
      CMAX=VHEAD(NC)-5.0*(F_WIDTH(VHEAD(NC)))
      DO 1000 I=1,128
        IF(NPF.EQ.300)GOTO 1100 !ARRAY FULL DONT ADD ANY MORE
        IF(XLIM(I).EQ.0.0)GOTO 1100 !END OF LIMIT LINES?
C CONVERT ENERGY VALUE TO CHANNEL NUMBER
C
        IF(VHEAD(CE).NE.0.0)THEN !TEST QUAD TERM OF ENERGY EQ.
          T1=VHEAD(AE)
          T2=VHEAD(BE)
          T3=VHEAD(CE)
          T4=T1-XLIM(I)
          T1=DSQRT(T2*T2-4.0*T3*T4)
          XL=(-T2+T1)/(2.0*T3)

```

```

        ELSE !ENERGY EQUATION IS LINEAR
            XL=(XLIM(I)-VHEAD(AE))/VHEAD(BE)
        ENDIF
        CVAL=XL-VHEAD(CZ)
C DONT ADD LINE TO ARRAY IF CENTROID IS GREATER THAN LENGTH OF SPECTRUM
C MINUS 5 WIDTHS
        IF(CVAL.GE.CMAX)GOTO 1000
        NPF=NPF+1
        PKD(NPF,1,XV)=CVAL
        PKD(NPF,1,WV)=F_WIDTH(CVAL)
        PKD(NPF,3,ERA)=XLIM(I)
        PKD(NPF,1,AV)=0.0
        PKD(NPF,1,DER)=0.0
        PKD(NPF,2,DW)=0.0
        PKD(NPF,2,DA)=0.0
        PKD(NPF,2,NOT)=0.0
        PKD(NPF,3,FLG)=1.222      !1.0 WAS OLD - USE 1.222 TO ID AS LIMIT
        PT(NPF).EVSC=0
C
                                !LIBRARY PEAK
1000  CONTINUE
        GOTO 100      !DO NEXT LIMIT BLOCK
1100  VHEAD(NP)=NPF
C NOW THAT ALL THE LIMIT LINES HAVE BEEN ADDED TO THE ARRAY
C WILL SORT THE ARRAY IN ACCENDING CHANNEL ORDER.
1110  DO I=1,NPF-1
        IF(PKD(I,1,XV).GT.PKD(I+1,1,XV))THEN
            DO J=1,3
                DO K=1,13
                    TMP=PKD(I,J,K)
                    PKD(I,J,K)=PKD(I+1,J,K)
                    PKD(I+1,J,K)=TMP
                ENDDO
            ENDDO
        ENDDO
C
C SWITCH THE BACKGROUND RESULTS AROUND ALSO
        TMP=BK.CHANNEL(I)
        BK.CHANNEL(I)=BK.CHANNEL(I+1)
        BK.CHANNEL(I+1)=TMP
        DO J=1,BK.NO_BKG
            TMP=BK.B_CPS(I,J)
            TMP2=BK.B_ERROR(I,J)
            BK.B_CPS(I,J)=BK.B_CPS(I+1,J)
            BK.B_ERROR(I,J)=BK.B_ERROR(I+1,J)
            BK.B_CPS(I+1,J)=TMP
            BK.B_ERROR(I+1,J)=TMP2
        ENDDO
C ALSO SWITCH FIT FUNCTION RESULTS IN PT

```

```

        PT1=PT(I)
        PT2=PT(I+1)
        PT(I)=PT2
        PT(I+1)=PT1
        GOTO 1110
    ENDIF

    ENDDO
C ARRAY HAS NOW BE SORTED INTO ACCENDING CHANNEL NO. NOW ELIMINATE
C ENTRIES WHOSE SEPERATION IS LESS THAN 1.0 KEV. IF A CHOISE EXISTS
C BETWEEN A LIMIT LINE AND FIT PEAK WITH VALID RESULTS ELIMINATE THE
C LIMIT LINE
    I=0
1500    I=I+1
        IF (I.GE.NPF) GOTO 1800          !ONLY DO UP TO NPF-1
        IF (ABS(PKD(I,3,ERA)-PKD(I+1,3,ERA)).GE.1.0) GOTO 1700
        IF ((PKD(I,3,FLG).GT.0).AND.(PKD(I+1,3,FLG).GT.0)) THEN
C CHECK IF 2 PEAKS WITH ENERGIES THAT DIFFER BY LESS THAN 1 KEV AND WERE
C NOT FIT BY SEARCH (FLG=0) HAVE DIFFERENT ISOTOPIC ASSIGNMENTS.
C IF SO, FIT BOTH OF THESE PEAKS. ELSE, DELETE THE PEAK THAT WAS NOT
C A LIMIT LIBRARY PEAK (TAG FOR LIMIT LIBRARY PEAKS IS 1.222).
            SAME=LIB_CHECK(PKD(I,3,ERA),PKD(I+1,3,ERA))
            IF (SAME) THEN
                IF (PKD(I+1,3,FLG).NE.1.222) THEN
                    IF (PKD(I,3,FLG).NE.1.222) GOTO 1520
                    IC=I !DELETE PEAK PKD(I+1)
                    NPF=NPF-1
                    GOTO 1522
                ELSE
                    IC=I+1 !DELETE PEAK PKD(I)
                    NPF=NPF-1
                    GOTO 1522
                END IF
            ELSE
                !NOT SAME - SHOULD FIT BOTH PEAKS...
                GOTO 1700
            END IF
        END IF
    END IF

1520    NPF=NPF-1
        IF (NPF.LE.2) THEN
            NPF=1
            GOTO 1800
        ENDIF
        IF (PKD(I,1,AV).EQ.0.0) THEN
            IC=I
        ELSE
            IC=I+1
            IF (IC.GT.NPF) IC=I
        END IF

```

```

1522      ENDIF
      DO II=IC,NPF
        DO J=1,3
          DO K=1,13
            PKD(II,J,K)=PKD(II+1,J,K)
          ENDDO
        ENDDO
      C
      C ELIMINATE FIT RESULTS
        PT2=PT(II+1)
        PT(II)=PT2
      C ELIMINATE BACKGROUND RESULTS ENTRY ALSO
      C
        BK.CHANNEL(II)=BK.CHANNEL(II+1)
        DO J=1,BK.NO_BKG
          BK.B_CPS(II,J)=BK.B_CPS(II+1,J)
          BK.B_ERROR(II,J)=BK.B_ERROR(II+1,J)
        ENDDO
      C
        ENDDO
        I=I-2
        IF (I.LT.0) I=0          !SET INDEX BACK TWO TO CHECK AGAINST
                                !PREVIOUS PEAK AND NEXT PEAK
      C
      1700 GOTO 1500
      C DONE WITH ELIMINATION
      C CHANGE PEAKS WITH TAG OF 1.222 BACK TO 1
      1800 DO II=1,NPF
        IF (PKD(II,3,FLG).EQ.1.222) PKD(II,3,FLG)=-1.0
      END DO
      VHEAD(NP)=NPF
      C
      C ANALYSIS ARRAY IS NOW SET UP WE CAN PROCEED TO PERFORM THE LIMIT CALCULATION
      C FOR THOSE ENTRIES IN THE ARRAY WHICH HAVE FIT VALUES ASSIGNED. WE WILL SEARCH
      C THE ARRAY FOR THOSE LINE WHICH HAVE A ZERO AREA ASSIGNED. ONCE WE FIND
      C THE LIMIT LINE WE MUST ASSIGN VALUES FOR 6 VARIABLES IN SETS OF TWO.
      C I.E. DETERMINE THE CHANNEL,WIDTH OF THE PEAK BEFORE THE LIMIT LINE, THE
      C LIMIT LINE ITSELF AND THE PEAK AFTER THE LIMIT LINE.
      C THESE VARIABLES ARE:
      C   BECP CHANNEL OF PEAK BEFORE LIMIT LINE
      C   BEWD WIDTH OF PEAK BEFORE LIMIT LINE
      C
      C   LMCP CHANNEL OF LIMIT LINE
      C   LMWD WIDTH OF LIMIT LINE
      C
      C   PSCP CHANNEL OF PEAK AFTER LIMIT LINE
      C   PSWD WIDTH OF PEAK AFTER LIMIT LINE
      C

```

```

DO 10000 IR=1,NPF
  IF(PKD(IR,1,AV).NE.0.0)GOTO 10000 !LIMIT LINE?
  IF(IR.EQ.1)THEN !LIMIT LINE FIRST ENTRY IN ARRAY
    BECP=0.0
    BEWD=0.0
    LMCP=PKD(1,1,XV)
    LMWD=F_WIDTH(LMCP)
    IF (NPF.EQ.1) GOTO 165 !NO PEAKS AFTER OR BEFORE...
    THEREFORE FIT AS NORMAL LIMIT
    PSCP=PKD(2,1,XV)
    IF(PKD(2,1,AV).EQ.0.0)THEN
      PSWD=F_WIDTH(PSCP)
    ELSE
      PSWD=PKD(2,1,WV)
    ENDIF
  ELSE
    !LIMIT LINE IS NOT FIRST ARRAY
    C IS PRECEEDING LINE A LIMIT VALUE LINE (I.E. LESS THAN) IF IT IS WE WANT
    C TO BACK UP TO THE LINE PRECEEDING IT UNTIL WE RUN OUT OF PEAK OR A
    C NONLIMIT LINE IS FOUND.
    J=IR-1
1825 IF((PKD(J,2,DA).LT.0.0).AND.(PKD(J,3,FLG).LT.0.0))THEN !PRECEEDING LINE IS LIMIT
      J=J-1
      IF(J.GT.1)GOTO 1825
    ENDIF
    BECP=PKD(J,1,XV)
    BEWD=PKD(J,1,WV)
    LMCP=PKD(IR,1,XV)
    LMWD=F_WIDTH(LMCP)
    IF(IR.EQ.NPF)THEN !LIMIT LINE LAST IN ARRAY?
      PSCP=CMAX
      PSWD=F_WIDTH(CMAX)
    C IF THERE IS NOT ENOUGH ROOM AT THE END OF THE SPECTRUM TO FIT THIS PEAK
    C TO 5 WIDTHS . TRY 2 IF THAT IS NOT ENOUGH IGNORE PEAK
      IF(PSCP.LT.LMCP)THEN
        IF(LMCP+(2.0*FWDTH).LT.(VHEAD(NC)-1))GOTO 165
        VHEAD(NP)=VHEAD(NP)-1.0
        RETURN
      ENDIF
    ELSE
      C IS NEXT LINE A LIMIT LINE IF IT IS CHECK NEXT PEAK IN LIST. WE WANT TO
      C MATCH LIMIT LINE WITH A NON LIMIT LINE IF POSSABLE.
      J=IR+1
1850 IF((PKD(J,1,AV).EQ.0.0).AND.(PKD(J,3,FLG).LT.0.0))THEN
      J=J+1
      IF(J.LT.NPF)THEN
        GOTO 1850
      ELSE

```

```

                                J=IR+1
                                ENDIF
                                ENDIF
                                PSCP=PKD(J,1,XV)
                                IF (PKD(J,1,AV).EQ.0.0) THEN
                                    PSWD=F_WIDTH(PSCP)
                                ELSE
                                    PSWD=PKD(J,1,WV)
                                ENDIF
                                ENDIF
                                ENDIF
                                ENDIF
C BEGIN CHECKS TO DETERMINE WHICH KINDS
C OF LIMIT FIT WE WILL DO. CRITERIA IS BASED
C ON THE RELATIONSHIP OF BECP,LMCP,PSCP
C
110   XL=BECP+(5.0*BEWD)
      XH=PSCP-(5.0*PSWD)
      IF (LMCP.LT.XL.OR.LMCP.GT.XH) GOTO 180
C PEAK WILL BE FIT AS A SINGLET IN THE RANGE
C OF 2 WIDTHS FROM LMCP
165   FWDTH=LMWD
170   IFC=LMCP-(2.0*FWDTH)
      LC=LMCP+(2.0*FWDTH)
      FXVAL=LMCP
      GOTO 5000
180   XH=PSCP-(3.0*PSWD)
      IF (LMCP.LT.XL.OR.LMCP.GT.XH) GOTO 200
C PEAK WILL BE FIT AS A SINGLET IN THE RANGE
C 3 WIDTHS ON LOW SIDE & 1 WIDTH ON HIGH SIDE
190   FWDTH=LMWD
      IFC=LMCP-(3.0*FWDTH)
      LC=LMCP+FWDTH
      FXVAL=LMCP
      GOTO 5000
200   XL=BECP+(2.5*BEWD)
      XH=PSCP-(5.0*PSWD)
      IF (LMCP.LT.XL.OR.LMCP.GT.XH) GOTO 220
C PEAK WILL BE FIT AS A SINGLET IN THE RANGE
C 1 WIDTH OF LOW SIDE AND 3 WIDTHS ON HIGH SIDE
210   FWDTH=LMWD
      IFC=LMCP-FWDTH
      LC=LMCP+(3.0*FWDTH)
      FXVAL=LMCP
      GOTO 5000
C CHECK FOR SEPERATION. IS THE TWO INTERFERANCE
C PEAKS SEPERATED BY LESS THAN 3 WIDTHS?
C IF SO DO NORMAL 2 WIDTH FIT

```

```

220   FWDTH=LMWD
      IF (ABS(BECP-PSCP).LT.3*FWDTH) GOTO 170
C CHECK FOR EQUAL SEPERATION OF ALL PEAKS
C IF EQUAL DO NORMAL 2 WIDTH FIT
      XL=ABS(BECP-LMCP)
      XH=ABS(PSCP-LMCP)
      IF (XL.EQ.XH) GOTO 170
C CHECK TO FIND WHICH PEAK IS CLOSER TO LMCP
      IF (XL.LT.XH) GOTO 230
      IF (XH.GT.(5.0*LMWD)) GOTO 165
      IB=LMCP-(PSWD*FF1+0.5)
      XL=LMWD
      IE=XL*FF2+1.7+PSCP
C DETERMINE IF REGION TO BE FIT IS JUST ZEROS
      DO I=IB,IE
        IF (FGET(I).GT.0.0) GOTO 226
      ENDDO
      IFC=IB
      LC=IE
      GOTO 5055 !REGION IS ZEROS SET VALUES
C DO DOUBLET FIT
226   CALL DPY_CBAT_LINFIT_DP(PSCP,LMCP,PSWD,XL,IB,IE,IR)
      GOTO 10000
230   IB=BECP-(BEWD*FF1+0.5)
      IF (XL.GT.5.0*LMWD) GOTO 165
      XL=LMWD
      IE=XL*FF2+1.7+LMCP
C DETERMINE IF REGION TO BE FIT IS JUST ZEROS
      DO I=IB,IE
        IF (FGET(I).GT.0.0) GOTO 236
      ENDDO
      IFC=IB
      LC=IE
      GOTO 5055 !REGION IS ZEROS SET VALUES
236   CALL DPY_CBAT_LINFIT_DP(BECP,LMCP,BEWD,XL,IB,IE,IR)
      GOTO 10000
CC BEGIN NORMAL LIMIT CALCULATION
C POSITION,WIDTH,AND BACKGROUND FIXED
C HEIGHT IS CALCULATED AS A LEAST SQUARES
5000  IXT=0
      DO 5025 I=1,3
        BVEC(I)=0.0
      DO 5025 J=1,3
5025  AMATR(I,J)=0.0
      FXERR=0.0
      FWDTH=F_WIDTH(FXVAL)
      FWERR=0.0

```

```

5026 IF(IFC-LC.GT.64)THEN      !RANGE TO WIDE SQUEEZE IT UP
      IFC=IFC+1
      LC=LC-1
      GOTO 5026
ENDIF
IBEG=IFC
IEND=LC
J=1
DO I=IBEG,IEND
      WDATA(J)=FGET(I)
      J=J+1
END DO
C
C LOAD WEIGHT VALUES FOR DATA IN REGION OF FIT INTO ARRAY WEIT
      K=1
      DO I=IFC,LC
            WEIT(K)=WEIGHT(I)
            K=K+1
      ENDDO
      LW08=0
      IE=LC-IFC+1
5050 FC=IFC
C
C CHECK TO SEE IF WHOLE REGION IS EQUAL TO ZERO
C
      DO I=1,IE
            IF (WDATA(I).NE.0.0) GO TO 5100
      ENDDO
5055 PKD(IR,1,AV)=1.0
      PKD(IR,2,DA)=100.0
      PKD(IR,1,WV)=FWDTH
      PKD(IR,1,DER)=0.0
      PKD(IR,2,DW)=0.0
      PKD(IR,2,NOT)=0.0
      PKD(IR,3,FLG)=1.0
      PKD(IR,3,CHI)=0.0
C SET COMPUTED VALUES TO ZERO
      PT(IR).EVSC=IBEG
      PT(IR).ENDC=IEND
      PT(IR).EVNC=(PT(IR).ENDC-PT(IR).EVSC)+1
5056 IF(PT(IR).EVNC.GT.64)THEN
      PT(IR).ENDC=PT(IR).ENDC-1
      PT(IR).EVSC=PT(IR).EVSC+1
      PT(IR).EVNC=PT(IR).EVNC-2
      GOTO 5056
ENDIF
PT(IR).ENNF=1

```



```

X=PT(IR).EVSC
K=1
DO I=PT(IR).EVSC,PT(IR).ENDC-1
    DO II=1,4
        PT(IR).F(1,K)=0.0
        K=K+1
        X=X+0.25
    ENDDO
ENDDO
PT(IR).NPF(1)=K-1
GOTO 5525 !FIT BACKGROUND REGIONS
5100 FC=IFC
DO 5300 JI=1,IE
    FT=FC-FXVAL
    ET=((FT/FWDTH)**2)*(-2.77258)
    FT3=EXP(ET)
    FT2=FT3*WEIT(JI)
    AMATR(1,3)=AMATR(1,3)+FT2
    AMATR(2,3)=AMATR(2,3)+FT2*FC
    AMATR(3,3)=AMATR(3,3)+FT2*FT3
    BVEC(3)=BVEC(3)+FT2*WDATA(LW08+JI)
    FT3=FC*WEIT(JI)
    AMATR(1,2)=AMATR(1,2)+FT3
    AMATR(2,2)=AMATR(2,2)+FT3*FC
    BVEC(2)=BVEC(2)+FT3*WDATA(LW08+JI)
    AMATR(1,1)=AMATR(1,1)+WEIT(JI)
    BVEC(1)=BVEC(1)+WEIT(JI)*WDATA(LW08+JI)
5300 FC=FC+1.0
    CALL INVT(3,AMATR,BVEC,PVEC,3)
C WERE ANY OF THE DATA POINTS .LT. 10 IF THEY WERE
C REPEAT THE PROCESS WHEN THESE CHANNELS ARE REPLACED
C WITH FUNCTIONAL VALUES
C FROM FIRST PASS
5400 TOTDEV=0.0
    FC=IFC
C COMPUTE ERRORS
DO 5450 JI=1,IE
    FT=FC-FXVAL
    ET=((FT/FWDTH)**2)*(-2.77258)
    FT3=EXP(ET)*PVEC(3)
    FT3=WDATA(JI)-PVEC(1)-(FC*PVEC(2))-FT3
    TOTDEV=TOTDEV+FT3*FT3*WEIT(JI)
5450 FC=FC+1.0
    FT=IE-3
    IF(TOTDEV.GE.0.0)GOTO 5455
    SCHISQ=100.0
    GOTO 5460

```

```

5455 SCHISQ=SQRT(TOTDEV/FT)
      IF(SCHISQ.LT.1.0)SCHISQ=1.0
5460 IF(AMATR(3,3).LE.0.0D0)GOTO 5555
      FAERR=DSQRT(AMATR(3,3))*SCHISQ*1.06445701*FWDTH
      FAREA=PVEC(3)*FWDTH*1.06445701
5470 FAERR=ABS((FAERR/FAREA)*100.0)
5510 PKD(IR,1,WV)=FWDTH
      IF(FAREA.EQ.0.0)THEN
          PKD(IR,1,AV)=1
          PKD(IR,2,DA)=1
      ELSE
          PKD(IR,1,AV)=FAREA
          PKD(IR,2,DA)=MIN(99999.0,FAERR)
      ENDIF
      PKD(IR,1,DER)=0.0
      PKD(IR,2,DW)=0.0
      PKD(IR,2,NOT)=0.0
      PKD(IR,3,FLG)=1.0
      PKD(IR,3,CHI)=SCHISQ
C
C LOAD FUNCTIONAL VALUES
C
      PT(IR).EVSC=IBEG
      PT(IR).ENDC=IEND
      PT(IR).EVNC=(PT(IR).ENDC-PT(IR).EVSC)+1
11000 IF(PT(IR).EVNC.GT.64)THEN
          PT(IR).ENDC=PT(IR).ENDC-1
          PT(IR).EVSC=PT(IR).EVSC+1
          PT(IR).EVNC=PT(IR).EVNC-2
          GOTO 11000
      ENDIF
      PT(IR).ENNF=1          !NUMBER OF FUNCTIONAL VALUES=
C
      X=PT(IR).EVSC
      K=1
      DO I=PT(IR).EVSC,PT(IR).ENDC-1
          DO II=1,4
              ET=EXP((((X-FXVAL)/FWDTH)**2)*(-2.77258))
              PT(IR).F(1,K)=ET*PVEC(3)+PVEC(1)
              +PVEC(2)*X
              K=K+1
              X=X+0.25
          ENDDO
      ENDDO
      PT(IR).NPF(1)=K-1      !NUMBER OF POINTS IN EACH FUNCTION ARRAY
C NOW FIT EACH BACKGROUND SPECTRUM TO THE REGIONS DETERMINED BY THE

```

```

C THE BACKGROUND CHANNEL NUMBERS
5525 BP.LINEAR_FIT=.TRUE.
      BK.CHANNEL(IR)=PKD(IR,1,XV) !CENTRIOD OF THIS PEAK
      DO J=1,BK.NO_BKG
        BF.E(J).NO_PEAKS=1
        BF.E(J).EXC(1)=PKD(IR,3,ERA)
        BF.E(J).EFC=C_ENERGY(FLOAT(IFC))
        BF.E(J).ELC=C_ENERGY(FLOAT(LC))
        CALL FIT_BKG(J)
      C
      C CALCULATE THE ERROR
        BK.B_ERROR(IR,J)=
          1 MIN(ABS((BF.E(J).A_ERR(1)/BF.E(J).AREA(1))*100.0),99999.0)
      C
      C
      C SAVE RESULTS OF FIT FOR THIS PEAK. FIRST CHANGE AREA TO CPS AND RANDOM
      C SUM CORRECT IT
      C PULSERED SPECTRUM?
        IF(.NOT.BF.E(J).PULSER)THEN
          BK.B_CPS(IR,J)=BF.E(J).AREA(1)/(BF.E(J).LIVE)
        ELSE
      C
      C PULSERED SPECTRUM, BUT DO WE HAVE VALID PARAMETERS IF NOT BF.E(J).SCA=1.0
      C AND BF.E(J).SCB=0.0
        IF(BF.E(J).SCA.EQ.1.0.AND.BF.E(J).SCB.EQ.0.0)THEN
          BK.B_CPS(IR,J)=BF.E(J).AREA(1)/BF.E(J).LIVE
        ELSE
          IF(BK.CHANNEL(1).LE.3600.0)THEN
            BK.B_CPS(IR,J)=(BF.E(J).AREA(1)/
              1 (((BF.E(J).SCB-BF.E(J).SCA)/3600.0)*
              2 BK.CHANNEL(1)+
              2 BF.E(J).SCA))*BF.E(J).RATE
          ELSE
            BK.B_CPS(IR,J)=(BF.E(J).AREA(1)/BF.E(J).SCB)*
              1 BF.E(J).RATE
            ENDIF
          ENDIF
        ENDIF
      ENDDO
      GOTO 10000
5555 WRITE(6,6009)
6009 FORMAT(' MATRIX ELEMENT NEGATIVE; IGNORING PEAK')
C END OR LIMIT FIT LOOP
10000 CONTINUE
      RETURN
      END

```

C.5 DPY_CBAT_LINFIT_DP Subroutine

```

C ROUTINE MODIFIED TO USE THE FUNCTION "F_WIDTH" FOR PEAK WIDTH DETERMINATION
C      EWK 1/15/91
C
C SUBROUTINE TO PERFORM A GAUSS FIT TO A PEAK TOGETHER WITH ITS INTERFERENCE
C LINE. ROUTINE CALL FROM CBAT_LINFIT. TO BE USED IN THE CBAT ENVIORNMENT
C      EWK 5/10/88
C      THIS ROUTINE DOESNT COMPUTE "UPPER LIMIT" VALUES
C      EWK 7/7/87
C
C      SUBROUTINE DPY_CBAT_LINFIT_DP(X1,X2,W1,W2,IB,IE,LIX)
C SUBROUTINE TO PERFORM GAUSS FIT ON A DOUBLET UNDER LIMFIT CRITERIA
C      X1=INTERFERENCE PEAD
C      X2=LIMIT PEAK
C      W1=WIDTH OF INTERFERACE PEAK
C      W2=WIDTH OF LIMIT PEAK
C      IB=1ST FIT CHANNEL
C      IE=LAST FIT CHANNEL
C      LIX=PEAK INDEX NUMBER FOR LIMIT PEAK
C
C      INCLUDE 'GAP$SRC:VHEAD.INC'
C      INCLUDE 'GAP$SRC:PKD.INC'
C      INCLUDE 'GAP$SRC:BACK_FIT.INC'
C      INCLUDE 'GAP$SRC:BACKGROUND_RESULTS.INC'
C      INCLUDE 'GAP$SRC:PLOT.INC'
C
C      DOUBLE PRECISION AMATR(4,4),PVEC(4),DFT,BVEC(4),WTDEV,TOT
C      DIMENSION
C      1  WDTN(2),YVAL(2),XVAL(2),
C      3  XERR(2),AERR(2),WERR(2),
C      4  AREA(2)
C      REAL LEAST_VAL,LL
C      DIMENSION WDATA(300),WEIT(300)
C      COMMON /FITIO/ IFTNR,IFTNW,IEKNW
C      DATA FF3/5.0/,FF4/5.0/,NMS/2/,IDTS/1/,ISRS/0/
C      FZR=VHEAD(CZ)
C      ATERM=VHEAD(AE)
C      BTERM=VHEAD(BE)
C      CTERM=VHEAD(CE)
C      XLINM=VHEAD(LTB)
C      IR=0
C      IGAUS=2
C      WDTN(1)=W1
C      WDTN(2)=W2
C      XVAL(1)=X1

```

```

        XVAL(2)=X2
        IFC=IB
        LC=IE
C DETERMINE WHICH PEAK IS LARGER
        ILX=1
        IMX=2
        IF(XVAL(1).LT.XVAL(2))GOTO 100
        ILX=2
        IMX=1
C FIND LOWER BACKGROUND CHANNEL LIMIT
100  TI=(F_WIDTH(XVAL(ILX)))*FF3+0.5
        IF(TI.LE.0.0)TI=1.0
        IBEG=XVAL(ILX)-TI
        IF (IBEG.LT.0) IBEG=2
        IF (IBEG.GE.IFC) IBEG=IFC-1
C FIND UPPER BACKGROUND CHANNEL LIMIT
        IEND=XVAL(IMX)+((F_WIDTH(XVAL(IMX))))*FF4
        NCS=VHEAD(NC)
        IF (IEND.GE.NCS-2) THEN
                IEND=NCS-1
                IF (LC.GE.IEND) LC=IEND-1
        ENDIF
C
C GET DATA FOR FIT
        IF ((IEND-IBEG).GT.300) THEN
                DO WHILE ((IEND-IBEG).GT.300)
                        IEND=IEND-1
                        IBEG=IBEG+1
                ENDDO
        ENDIF
        J=1
        DO I=IBEG, IEND
                WDATA(J)=FGET(I)
                J=J+1
        END DO
C
C LOAD WEIGHT VALUES FOR DATA IN REGION OF FIT INTO ARRAY WEIT
        K=1
        DO I=IFC, LC
                WEIT(K)=WEIGHT(I)
                K=K+1
        ENDDO
        IE=LC-IFC+1
        LW08=IFC-IBEG
C
C CALCULATE THE COORDINATES OF BACKGROUND LINE
        CALL REGION_BKG_LINE

```

```

40000 BKSLOP=(YRMIN-YLMIN)/(XRMIN-XLMIN)
      YMAX=0.0
      YMIN=999999999.0
C SET INITIAL APPROXIMATION FOR CENTROID, CENTROID HEIGHT,
C AND GAUSS WIDTH FOR EACH PEAK IN THE FIT REGION
      DO 117 I=1, IGAUS
        LPEK=XVAL(I)+1.0-IBEG
        AY=WDATA(LPEK-1)
        BY=WDATA(LPEK)
        CY=WDATA(LPEK+1)
        SABC=AY+BY+CY
        IF(SABC.EQ.0.0) THEN
          FT=0.0
        ELSE
          FT=(CY-AY)/SABC
        ENDIF
C FOR THIS TYPE OF FIT XVAL(2)&WDTH(2) ARE FIXED
        IF(I.EQ.2) GOTO 115
        XVAL(I)=XVAL(I)+FT
        WDTH(I)=F_WIDTH(XVAL(I))
C FORMULA DERIVED FROM LAGRANGIAN POLYNOMIAL WITH EQUIDISTANT
C ABSISSAS
115      Y=(((-2.0*BY+AY+CY)*FT+(CY-AY))/2.0*FT+BY
        YY=Y-((XVAL(I)-XLMIN)*BKSLOP+YLMIN)
        IF(YY.GE.1.0) Y=YY
        YVAL(I)=Y
        IF(Y.LT.YMIN) YMIN=Y
117 IF(Y.GT.YMAX) YMAX=Y
C SUBTRACT BACKGROUND FROM FIT REGION VALUES
      FC=IFC
      DO 130 I=1, IE
        WDATA(LW08+I)=WDATA(LW08+I)-((FC-XLMIN)*BKSLOP+YLMIN)
130 FC=FC+1.0
      XMIN=IFC
      XMAX=LC
      YMIN=YMIN/8.0
      YMAX=YMAX*8.0
      WMIN=WDTH(IGAUS)/10.0
      WMAX=WMIN*100.0
      MDIM=4
      NOIT=MDIM*2+1
C *****
C END OF GAUSS SETUP
C START OF GAUSS FITTING. DO FOR MAX ITERATIONS (NOIT) OR
C TIL CONVERGENCE IS MADE (WHICHEVER OCCURS FIRST).
      IPCFL=0
      NMS3=4

```

```

      JJ=0
200  JJ=JJ+1
      IF (JJ.GT.NOIT) GO TO 700
      DO 201 I=1,NMS3
      BVEC(I)=0.0
      DO 201 J=1,NMS3
201  AMATR(I,J)=0.0
      FC=IFC
C DO LEAST SQUARES FIT OVER REGION FOR EACH PEAK
      DO 230 JI=1,IE
      GSUM=0.0
C FOR MULTIPLETS THE WIDTHS ARE FIXED
C JUST ONE ENTRY IN PVEC FOR WIDTH
      DO 221 I=1,IGAUS
      I3=3*(I-1)+1
      FT=(FC-XVAL(I))/WDTH(I)
      ET=FT*FT*(-2.77258873)
      DFDH=EXP(ET)
      IF(DFDH.EQ. 0)GOTO 700 !EXPONENT OVERFLOW
      PVEC(I3)=DFDH
      CNVC=YVAL(I)*DFDH
      GSUM=GSUM+CNVC
      DFDC=(CNVC*5.54517746*FT)/WDTH(I)
      IF(I.EQ.2)GOTO 221
      PVEC(I3+1)=DFDC
C DFDW=DFDC*FT
      PVEC(I3+2)=DFDC*FT
221  CONTINUE
      DFT=WEIT(JI)
      WTDEV=(WDATA(LW08+JI)-GSUM)*DFT
      DO 225 I=1,MDIM
      BVEC(I)=PVEC(I)*WTDEV+BVEC(I)
      DO 225 J=1,MDIM
225  AMATR(I,J)=AMATR(I,J)+PVEC(I)*PVEC(J)*DFT
230  FC=FC+1.0
      IF(IPCFL.EQ.1)GO TO 227
      CALL INVTB(MDIM,AMATR,BVEC,PVEC,4)
      GO TO 228
227  CALL INVT(MDIM,AMATR,BVEC,PVEC,4)
C COMPUTE NEW VALUES AND TEST FOR CONVERGENCE
228  IC=0
      I3=1
      DO 245 I=1,IGAUS
      DFT=DABS(PVEC(I3))
      IF(DFT.LT.1.0)GO TO 232
      IF(YVAL(I).EQ.0.0)GOTO 700
      FT=PVEC(I3)/ABS(YVAL(I))-0.005

```

```

        IF(FT.LT.0.0)GO TO 232
        IC=IC+1
        FT=ABS(YVAL(I))/2.0
        IF(DFT.GT.FT)PVEC(I3)=FT
232  FT=PVEC(I3)+YVAL(I)
        IF(FT.GE.YMAX)GOTO 700
        IF(FT.LT.YMIN)GOTO 700
        YVAL(I)=FT
        IF(I.EQ.2)GOTO 240
        DFT=DABS(PVEC(I3+1))
        IF(DFT.GT.XMAX)GOTO 700
        IF(DFT.GE.0.005)IC=IC+1
        IF(DFT.GT.2.0)PVEC(I3+1)=2.0
        FT=PVEC(I3+1)+XVAL(I)
        IF(FT.GT.XMAX)GOTO 700
        IF(FT.LT.XMIN)GOTO 700
        XVAL(I)=FT
        DFT=DABS(PVEC(3))
        IF(DFT.GE.0.005)IC=IC+1
        FT=ABS(WDTH(I))/2.0
        IF(DFT.GT.FT)PVEC(3)=FT
        FT=PVEC(3)+WDTH(I)
        IF(FT.GT.WMAX)GOTO 700
        IF(FT.LT.WMIN)GOTO 700
        I3=4
245  WDTH(I)=FT
240  IF(IC.NE.0)GO TO 200
        IF(IPCFL.NE.0)GO TO 300
        IPCFL=1
        JJ=NOIT-1
        GO TO 200
C  COMPUTE ESTIMATED ERRORS
300  FC=IFC
        TOTDEV=0.0
        DO 310 JI=1, IE
        GSUM=0.0
        DO 305 I=1, IGAUS
        FT=(FC-XVAL(I))/WDTH(I)
        ET=FT*FT*(-2.77258873)
305  GSUM=GSUM+YVAL(I)*EXP(ET)
        FT=GSUM-WDATA(LW08+JI)
        TOTDEV=TOTDEV+FT*FT*WEIT(JI)
310  FC=FC+1.0
        FT=IE-MDIM
        SCHISQ=SQRT(TOTDEV/FT)
        IF(SCHISQ.LT.1.0)SCHISQ=1.0
C  ERROR FOR FIXED PARAMETERS ARE 0.0

```



```

WERR(2)=0.0
XERR(2)=0.0
WERR(1)=DSQRT(AMATR(3,3))*SCHISQ
DO 315 IA=1, IGAUS
  I3=3*(IA-1)+1
  IF(IA.EQ.2)GOTO 312
  XERR(IA)=DSQRT(AMATR(I3+1,I3+1))*SCHISQ
312  FA=WDTH(IA)*YVAL(IA)
  AREA(IA)=FA*1.06446
  FT2=WDTH(IA)**2*AMATR(I3,I3)
  FT3=YVAL(IA)**2*AMATR(3,3)
  IF(IA.EQ.2)FT3=0.0
  FT=FA*AMATR(I3,3)*2.0
  IF(IA.EQ.2)FT=0.0
  FT=FT+FT2+FT3
  AERR(IA)=SQRT(FT)*SCHISQ*1.06446
315  CONTINUE
316  NAFLG=3
330  FLTV=AREA(2)
350  FXVAL=XVAL(2)
  FXERR=XERR(2)
  FWDTH=WDTH(2)
  FWERR=WERR(2)
  FAREA=AREA(2)
  SAERR=AERR(2)
  FAERR=ABS(SAERR/FAREA)*100.0
C
C LOAD RESULTS INTO ARRAY
  PKD(LIX,1,WV)=FWDTH
  IF(FAREA.EQ.0.0)THEN
    PKD(LIX,1,AV)=1.0E-5
  ELSE
    PKD(LIX,1,AV)=FAREA
  ENDIF
  PKD(LIX,1,DER)=FXERR
  PKD(LIX,2,DW)=0.0
  PKD(LIX,2,DA)=MIN(99999.0,FAERR)
  PKD(LIX,2,NOT)=0.0
  PKD(LIX,1,XV)=FXVAL
  PKD(LIX,3,FLG)=NAFLG
  PKD(LIX,2,CPS)=FAREA/VHEAD(LVT)
  PKD(LIX,3,CHI)=SCHISQ
C
C SAVE THE RESULTS OF THE FIT
10000 PT(LIX).EVSC=(IFC-IBEG)/2+IBEG
  PT(LIX).ENDC=(IEND-LC)/2+LC
  PT(LIX).EVNC=(PT(LIX).ENDC-PT(LIX).EVSC)+1

```

```

11000 IF (PT(LIX).EVNC.GT.64) THEN
    PT(LIX).ENDC=PT(LIX).ENDC-1
    PT(LIX).EVSC=PT(LIX).EVSC+1
    PT(LIX).EVNC=PT(LIX).EVNC-2
    GOTO 11000
ENDIF
C LINEAR DOUBLET FIT

    PT(LIX).ENNF=2
C
C FILL FIRST FUNCTION ARRAY WITH BACKGROUND VALUES
    J=1
    DO I=PT(LIX).EVSC,PT(LIX).ENDC
        PT(LIX).F(1,J)=(I-XLMIN)*BKSLOP+YLMIN
        J=J+1
    ENDDO
    PT(LIX).NPF(1)=PT(LIX).EVNC !NUMBER OF BACKGROUND FUNCTIONAL DATA POINTS
                                !IS JUST THE NUMBER OF CHANNELS
C
C DO GAUSSIAN COMPONENT
    X=PT(LIX).EVSC
    K=1
    DO I=PT(LIX).EVSC,PT(LIX).ENDC-1
        DO II=1,4
            BX=(X-XLMIN)*BKSLOP+YLMIN
            PT(LIX).F(2,K)=BX+YVAL(2)*EXP(-2.77258873*
1 ((X-XVAL(2))/WDTH(2))**2)
            K=K+1
            X=X+0.25
        ENDDO
    ENDDO
    PT(LIX).NPF(2)=K-1 !NUMBER OF POINTS IN EACH FUNCTION ARRAY
C
C
C NOW FIT EACH OF THE BACKGROUND SPECTRA AS A DOUBLET LINEAR FIT. THE RESULTS
C FROM THE REQUIRED LINE ONLY WILL BE SAVED AS A BACKGROUND VALUE
    BP.LINEAR_FIT=.TRUE.
    BK.CHANNEL(LIX)=XVAL(2) !CENTRIOD OF REQUIRED PEAK
    DO I=1,BK.NO_BKG
        BF.E(I).NO_PEAKS=2
        BF.E(I).EXC(1)=C_ENERGY(XVAL(1)) !INTERFERENCE PEAK CENTRIOD ENG
        BF.E(I).EXC(2)=C_ENERGY(XVAL(2)) !REQUIRED PEAK CENTRIOD ENERGY
        BF.E(I).EFC=C_ENERGY(FLOAT(IFC)) !FIRST CHANNEL OF THIS FIT
        BF.E(I).ELC=C_ENERGY(FLOAT(LC)) !LAST CHANNEL OF THIS FIT
        CALL FIT_BKG(I)
    ENDDO
C
C CALCULATE THE ERROR

```

```

        BK.B_ERROR(LIX,I)=
1      MIN(ABS((BF.E(I).A_ERR(2)/BF.E(I).AREA(2))*100.0),99999.0)
C
C SAVE RESULTS OF FIT FOR THIS PEAK. FIRST CHANGE AREA TO CPS AND RANDOM
C SUM CORRECT IT
C PULSERED SPECTRUM?
        IF(.NOT.BF.E(I).PULSER) THEN
            BK.B_CPS(LIX,I)=BF.E(I).AREA(2)/(BF.E(I).LIVE)
        ELSE
C
C PULSERED SPECTRUM, BUT DO WE HAVE VALID PARAMETERS IF NOT BF.E(I).SCA=1.0
C AND BF.E(I).SCB=0.0
            IF(BF.E(I).SCA.EQ.1.0.AND.BF.E(I).SCB.EQ.0.0) THEN
                BK.B_CPS(LIX,I)=BF.E(I).AREA(2)/BF.E(I).LIVE
            ELSE
                IF(BK.CHANNEL(2).LE.3600.0) THEN
                    BK.B_CPS(LIX,I)=(BF.E(I).AREA(2)/
1                (((BF.E(I).SCB-BF.E(I).SCA)/3600.0)*
2                BK.CHANNEL(2)+
2                BF.E(I).SCA))*BF.E(I).RATE
                ELSE
                    BK.B_CPS(LIX,I)=(BF.E(I).AREA(2)/BF.E(I).SCB)*
1                BF.E(I).RATE
                ENDIF
            ENDIF
        ENDIF
    ENDDO
    RETURN
C IF FITTING PROCESS BLEW UP WE COME HERE WHERE ALL
C VARIABLES EXCEPT THE WHIGHTS ARE FIXED
700  IF((IEND-IBEG).GT.300) THEN
        DO WHILE ((IEND-IBEG).GT.300)
            IEND=IEND-1
            IBEG=IBEG+1
        ENDDO
    ENDIF
    J=1
    DO 711 I=IBEG,IEND
        WDATA(J)=FGET(I)
        J=J+1
711  CONTINUE
    DO 720 I=1,2
        BVEC(I)=0.0
    DO 720 J=1,2
720  AMATR(I,J)=0.0
        XVAL(1)=X1
        XVAL(2)=X2

```

```

WDTH(1)=W1
WDTH(2)=W2
FC=IFC
DO 780 JI=1, IE
DO 730 I=1, IGAUS
FT=(FC-XVAL(I))/WDTH(I)
ET=FT*FT*(-2.77258873)
FT=EXP(ET)
IF(FT.LT.1.0E-30)FT=0.0
PVEC(I)=FT
IF(WDATA(LW08+JI).LT.1)WDATA(LW08+JI)=1.0
BVEC(I)=BVEC(I)+PVEC(I)-(PVEC(I)*(YLMIN+BKSLOP*(FC-XLMIN))
1 /WDATA(LW08+JI))
730 AMATR(I,I)=AMATR(I,I)+PVEC(I)*PVEC(I)/WDATA(LW08+JI)
AMATR(1,2)=AMATR(1,2)+(PVEC(1)*PVEC(2))/WDATA(LW08+JI)
780 FC=FC+1.0
CALL INVT(2,AMATR,BVEC,PVEC,4)
YVAL(1)=PVEC(1)
YVAL(2)=PVEC(2)
FC=IFC
DO 790 I=1, IE
WDATA(LW08+I)=WDATA(LW08+I)-((FC-XLMIN)*BKSLOP+YLMIN)
790 FC=FC+1.0
C CALCULATE THE ERRORS
DO 800 I=1,2
WERR(I)=0.0
800 XERR(I)=0.0
FC=IFC
TOTDEV=0.0
DO 810 JI=1, IE
GSUM=0.0
DO 805 I=1, IGAUS
FT=(FC-XVAL(I))/WDTH(I)
ET=FT*FT*(-2.77258873)
805 GSUM=GSUM+YVAL(I)*EXP(ET)
FT=GSUM-WDATA(LW08+JI)
TOTDEV=TOTDEV+FT*FT*WEIT(JI)
810 FC=FC+1.0
FT=IE-2
IF(TOTDEV.GE.0.0)GOTO 812
SCHISQ=1000.0
GOTO 814
812 SCHISQ=SQRT(TOTDEV/FT)
IF(SCHISQ.LT.1.0)SCHISQ=1.0
814 DO 815 IA=1, IGAUS
FA=WDTH(IA)*YVAL(IA)
AREA(IA)=FA*1.06446

```

```

      FT2=WDTH(IA)**2*AMATR(IA,IA)
      IF(FT2.LT.0.0)GOTO 820
815   AERR(IA)=SQRT(FT2)*SCHISQ*1.06446
      FLTV=AREA(2)
      GOTO 316
820   IF(VHEAD(DPA).NE.1.0)RETURN
      WRITE(6,7005)
7005  FORMAT(//,' MATRIX ELEMENT THIS FIT IS NEGATIVE IGNORING PEAK')
      RETURN
      END

```

C.6 FIND_LINE Subroutine

```

C ROUTINE TO LOOK UP A LINE VALUE IN THE PKD ARRAY
C      EWK 4/6/92
C
C Modified to also return the half-life (see also FIND_LINE_STRUC.INC).
C D. A. Femec, July 22, 1993.
C
C Modified to also return the decay correction factor and to return the
C explicitly calculated efficiency at the library energy in FL.EFF. (see also
C FIND_LINE_STRUC.INC). D. A. Femec, January 25, 1994.
C

```

```

      SUBROUTINE FIND_LINE(Z,A,ID,ENG)
C
C WHERE:
C      Z=ATOMIC NUMBER
C      A=ATOMIC WEIGHT
C      ID=NUCLIDE ID
C      ENG=ENERGY OF LINE
C VALUES ARE RETURNED IN STRUCTURE FIND_LINE_STRUC
C
      INCLUDE 'GAP$SRC:PKD.INC'
      INCLUDE 'GAP$SRC:VHEAD.INC'
      INCLUDE 'FIND_LINE_STRUC.INC'

      INTEGER*4  Z,A
      CHARACTER*4 CID
      CHARACTER*(*)  ID
      REAL*4      ENG,RID
      EQUIVALENCE (RID,CID)

      NPF=VHEAD(NP)
      DO I=1,NPF

```

```

      IF (ABS (PKD (I, 3, ERA) - ENG) .GT. 1.0) GOTO 1000
      NT = PKD (I, 2, NOT)
      IF (NT .EQ. 0) GOTO 1000
      DO J = 1, NT
        IF (PKD (I, J, IZA) .NE. Z) GOTO 500
        IF (PKD (I, J, IAA) / 10.0 .NE. A) GOTO 500
        RID = PKD (I, J, ISA)
        IF (CID (1:1) .EQ. ' ') CID (1:4) = CID (2:4)
        K = LEN (ID)
        IF (CID (1:K) .NE. ID (1:K)) GOTO 500
        FL.LMC = PKD (I, J, MCI)
        FL.BR = PKD (I, J, BRA)
        FL.DCA = PKD (I, J, DCA)
        FL.ERR = PKD (I, J, EMC) * FL.BR / 100.0 * FL.LMC
        FL.FLAG = PKD (I, 3, FLG)
        FL.CPS = PKD (I, 2, CPS)
        FL.EFF = FEFF (PKD (I, J, L$ENG))
        FL.HL = ABS (PKD (I, J, HLA)) !Just in case . . . .
      RETURN
500      ENDDO
1000     ENDDO
      FL.ERR = 0.0
      RETURN
      END

```

C.7 MU_LOOK Subroutine

```

C SUBROUTINE TO DO A TABLE LOOK UP IN THE ATTENUATION TABLE AND RETURN
C AN ATTENUATION FACTURE (MU). TWO DIFFERENT TABLES ARE AVAILABLE
C A SILICON AND AN IRON TABLE THE TABLE TO USE IS FOUND IN THE PARAMETER BLOCK
C P.SYS_VOLUME_UNITS.
C      EWK 4/8/92
C
C      SUBROUTINE MU_LOOK (ENG, MU)
C
C WHERE:
C      ENG = ENERGY OF LINE
C      MU = LOOKED UP VALUE
C
C      INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
C      INCLUDE 'AT_TABLE.INC'
C
C      CHARACTER*2 TBLE

```

```

      REAL*4      ENERGY
      REAL*4      MU
      REAL*8      MX(3,3)
      REAL*8      V(3)
      REAL*8      PP(3)

C LOAD THE RESIDENT ATTENUATION STRUCTURE WITH THE INFORMATION FROM THE
C PROPER ATTENUATION TABLE
      IF(TBLE .NE. P.SYS_VOLUME_UNITS(1:2))CALL READ_atable
      TBLE=P.SYS_VOLUME_UNITS(1:2)
C
C IS ENERGY VALUES INSIDE THE RANGE OF THE TABLE
      IF((ENG .LT. AT.ENERGY(1)) .OR. (ENG .GT. AT.ENERGY(AT.NP)))THEN
          MU=1.0
          RETURN
      ENDIF

C FIND TABLE ENERGY VALUE CLOSEST TO THE INPUT ENERGY VALUE
C
      DELTA=1.0E+30
      DO I=2,AT.NP
          DE=ABS(AT.ENERGY(I)-ENG)
          IF(DE .LT. DELTA)THEN
              DELTA=DE
          ELSE
              GOTO 200
          ENDIF
      ENDDO

C SET UP MATRIX VALUES AND INVERT

200  NIT=I-2
      MX(1,1)=3.0
      MX(1,2)=0.0
      MX(1,3)=0.0
      MX(2,3)=0.0
      MX(3,3)=0.0

      DO I=1,3
          J=NIT+I-1
          MX(1,2)=MX(1,2) + DBLE(AT.ENERGY(J))
          MX(1,3)=MX(1,3) + (DBLE(AT.ENERGY(J))**2)
          MX(2,3)=MX(2,3) + (DBLE(AT.ENERGY(J))**3)
          MX(3,3)=MX(3,3) + (DBLE(AT.ENERGY(J))**4)
      ENDDO

      MX(2,1)=MX(1,2)

```

```

MX(2,2)=MX(1,3)
MX(3,1)=MX(2,2)
MX(3,2)=MX(2,3)

V(1)=0.0
V(2)=0.0
V(3)=0.0

DO I=1,3
    J=NIT+I-1
    V(1)=V(1)+DBLE(AT.ATF(J))
    V(2)=V(2)+(DBLE(AT.ATF(J))*(DBLE(AT.ENERGY(J))))
    V(3)=V(3)+(DBLE(AT.ATF(J))*(DBLE(AT.ENERGY(J))**2))
ENDDO

CALL INVT(3,MX,V,PP,3)

MU=SNGL(PP(1)+(PP(2)*DBLE(ENG))+(PP(3)*(DBLE(ENG)**2)))
RETURN
END

```

C.8 READ_ATABLE Subroutine

```

C ROUTINE TO READ THE VALUES FROM AN ATTENUATION TABLE AND PUT RESULTS IN
C COMMON ATTENUATION STRUCTURE 'AT'. THE TABLES WILL BE LOCATED IN AT$TABLE
C AND HAVE A NAMEING CONVENTION OF <DD>.TBL. WITH THE VALUE FOR DD DETERMINED
C FROM THE FIRST TWO CHARACTERS IN PARAM_BLOCK ENTRY P.SYS_VOLUME_UNITS.
C THE ENTRIES IN THE TABLE ARE ASCII WITH A C IN COLUMN 1 FOR COMMENT.
C FORMAT OF VALID LINE ENERGY,FACTOR
C
C      EWK 3/9/93
C
C Modified to include the attenuation table file name in all messages
C preceding CALL EXITS. D. A. Femec, June 23, 1993.
C
C Modified to accept up to 100 energy-attenuation factor pairs (this change was
C missed when AT_TABLE.INC was modified). D. A. Femec, March 1, 1994.
C

```

SUBROUTINE READ_ATABLE

```

INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
INCLUDE 'AT_TABLE.INC'

```

```

CHARACTER*80      LINE
CHARACTER*15      FNAME /'AT$TABLE:FE.TBL'/

```



```

LOGICAL EX

CALL LIB$GET_LUN(LUN)
IF(INDEX(P.SYS_VOLUME_UNITS(1:2),' ') .EQ. 1) THEN
    FNAME(10:10)=' '
    FNAME(11:11)=P.SYS_VOLUME_UNITS(2:2)
ELSE
    FNAME(10:11)=P.SYS_VOLUME_UNITS(1:2)
ENDIF

C DOES FILE EXIST?
INQUIRE(FILE=FNAME, EXIST=EX)
IF(.NOT. EX) THEN
    TYPE *, ' ATTENUATION TABLE FILE (' ,fname,') DOES NOT EXIST'
    CALL EXIT
ENDIF

C FILE DOES EXIST OPEN IT AND START READING

AT.NP=0
OPEN(UNIT=LUN, NAME=FNAME, TYPE='OLD', READONLY, SHARED)

100  READ(LUN, 9000, END=2000) NOC, LINE
9000  FORMAT(Q, A80)

IF(NOC .EQ. 0) GOTO 100

IF(LINE(1:1) .EQ. 'C') GOTO 100      !COMMENT IGNORE

I=AT.NP+1
READ(LINE, FMT=*, IOSTAT=IOS) AT.ENERGY(I), AT.ATF(I)

IF(IOS .NE. 0) THEN

200  TYPE *, ' ATTENUATION TABLE (' ,fname,') FORMAT ERROR'
    TYPE *, ' ', LINE(1:NOC)
    CALL EXIT
ENDIF

C SEE IF VALUES ARE SOME WHAT REASONABLE
IF((AT.ENERGY(I) .LT. 1.0) .OR. (AT.ENERGY(I) .GT. 3000.0)) GOTO 200

IF((AT.ATF(I) .LT. 0.0) .OR. (AT.ATF(I) .GT. 100.0)) GOTO 200

AT.NP=AT.NP +1
IF(AT.NP .EQ. 100) THEN
    READ(LUN, 9000, END=2000) NOC, LINE

```

```

1      TYPE *, ' MAXIMUM NUMBER OF LINES FOR ATTENUATION FILE (' ,
      fname, ') EXCEEDED'
      TYPE *, ' MAXIMUM NUMBER OF ENTRIES IS 100'
      CALL EXIT
ENDIF

GOTO 100

C END OF FILE ON TABLE FILE
2000 CLOSE(UNIT=LUN)
      CALL LIB$FREE_LUN(LUN)
      RETURN
      END

```

C.9 MU_SELF_LOOK Subroutine

```

C SUBROUTINE TO DO A TABLE LOOK UP IN THE ATTENUATION TABLE AND RETURN
C AN ATTENUATION FACTURE (MU). TWO DIFFERENT TABLES ARE AVAILABLE
C A SILICON AND AN IRON TABLE THE TABLE TO USE IS FOUND IN THE PARAMETER BLOCK
C P.SYS_VOLUME_UNITS.
C      EWK 4/8/92
C
C Derivative version of MU_LOOK to obtain the self-attenuation factor from the
C PU table. The parameter TBLE will be used to save the original value of
C P.SYS_VOLUME_UNITS(1:2), P.SYS_VOLUME_UNITS(1:2) will temporarily be reset to
C 'PU', and a read of the appropriate attenuation table will be forced. Just
C before returning to the calling routine, P.SYS_VOLUME_UNITS(1:2) will be
C restored to its original value. D. A. Femec, June 23, 1993.
C
C Expanded to access various self-attenuation tables based on the identification
C code ID passed to it. D. A. Femec, July 27, 1993.
C
C Forced a call to READ_ATALE at the end of the routine to ensure that the
C matrix attenuation table is properly reloaded. D. A. Femec, August 2, 1993.
C
      SUBROUTINE MU_self_LOOK(id,ENG,MU)
C
C WHERE:
C      id=identification code for the attenuation table (PU, U, or EU)
C      ENG=ENERGY OF LINE
C      MU=LOOKED UP VALUE
C
      INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'

```

```

C
  INCLUDE 'AT_TABLE.INC'

  character*4 id    !Identification code for the attenuation table.
  CHARACTER*2 TBLE  !Temporary storage of P.SYS_VOLUME_UNITS(1:2).


  REAL*4      ENERGY
  REAL*4      MU
  REAL*8      MX(3,3)
  REAL*8      V(3)
  REAL*8      PP(3)

C LOAD THE RESIDENT ATTENUATION STRUCTURE WITH THE INFORMATION FROM THE
C PROPER ATTENUATION TABLE
  tble=p.sys_volume_units(1:2)
  p.sys_volume_units(1:2)=id(1:2)
  CALL READ_ATABLE
C
C IS ENERGY VALUES INSIDE THE RANGE OF THE TABLE
  IF((ENG .LT. AT.ENERGY(1)) .OR. (ENG .GT. AT.ENERGY(AT.NP)))THEN
    MU=1.0
C
C Restore the first two characters of P.SYS_VOLUME_UNITS to their original
C value before returning.
C
    p.sys_volume_units(1:2)=tble
    RETURN
  ENDIF

C FIND TABLE ENERGY VALUE CLOSEST TO THE INPUT ENERGY VALUE
C
  DELTA=1.0E+30
  DO I=2,AT.NP
    DE=ABS(AT.ENERGY(I)-ENG)
    IF(DE .LT. DELTA)THEN
      DELTA=DE
    ELSE
      GOTO 200
    ENDIF
  ENDDO

C SET UP MATRIX VALUES AND INVERT
200  NIT=I-2
     MX(1,1)=3.0
     MX(1,2)=0.0

```

```

MX(1,3)=0.0
MX(2,3)=0.0
MX(3,3)=0.0

```

```

DO I=1,3
  J=NIT+I-1
  MX(1,2)=MX(1,2) + DBLE(AT.ENERGY(J))
  MX(1,3)=MX(1,3) + (DBLE(AT.ENERGY(J))**2)
  MX(2,3)=MX(2,3) + (DBLE(AT.ENERGY(J))**3)
  MX(3,3)=MX(3,3) + (DBLE(AT.ENERGY(J))**4)
ENDDO

```

```

MX(2,1)=MX(1,2)
MX(2,2)=MX(1,3)
MX(3,1)=MX(2,2)
MX(3,2)=MX(2,3)

```

```

V(1)=0.0
V(2)=0.0
V(3)=0.0

```

```

DO I=1,3
  J=NIT+I-1
  V(1)=V(1)+DBLE(AT.ATF(J))
  V(2)=V(2)+(DBLE(AT.ATF(J))*(DBLE(AT.ENERGY(J))))
  V(3)=V(3)+(DBLE(AT.ATF(J))*(DBLE(AT.ENERGY(J))**2))
ENDDO

```

```

CALL INVT(3,MX,V,PP,3)

```

```

MU=SNGL(PP(1)+(PP(2)*DBLE(ENG))+(PP(3)*(DBLE(ENG)**2)))

```

```

C
C Restore the first two characters of P.SYS_VOLUME_UNITS to their original
C value and read the appropriate attenuation table before returning.
C

```

```

  p.sys_volume_units(1:2)=tble
  call read_atable
  RETURN
END

```

C.10 LSTSQR Subroutine

```

C
C Subroutine for performing a weighted least-squares fit to user-supplied
C linear data.

```

```

C      SUBROUTINE LSTSQR (X, Y, SIGMAY, NSETS, NTERMS, A, stdevY, stdevA)
C
C PURPOSE
C      MAKE A WEIGHTED LEAST-SQUARES FIT TO USER-SUPPLIED LINEAR DATA OF THE
C      FORM  $Y(I) = A(1)X(1,I) + A(2)X(2,I) + \dots + A(NTERMS)X(NTERMS,I)$ 
C      WHERE THERE ARE NSETS SETS OF DATA (I.E., I RANGES FROM 1 TO NSETS)
C
C PARAMETERS
C      X      ARRAY (NTERMS BY NSETS) OF DATA POINTS FOR INDEPENDENT VARIABLE
C      Y      ARRAY OF DATA POINTS FOR DEPENDENT VARIABLE
C      SIGMAY  ARRAY OF STANDARD DEVIATIONS FOR DEPENDENT VARIABLE
C      NPTS   NUMBER OF DATA POINTS
C      NSETS  NUMBER OF SETS OF DATA POINTS
C      NTERMS NUMBER OF COEFFICIENTS TO RETURN
C      A      ARRAY OF COEFFICIENTS OF POLYNOMIAL RETURNED
C      CHISQR REDUCED CHI SQUARE OF FIT RETURNED
C
C This generalized version, using user-supplied linear data, is based on
C POLYFIT, with additions and annotations as per NBS Handbook 91, "Experimental
C Statistics" by M. G. Natrella. It has been expanded to calculate and return
C the uncertainties in the coefficients (SIGMAA). The prior goodness-of-fit
C calculation based on a reduced chi-squared value (CHISQR) has been replaced
C with a standard deviation of the predicted Y from the observed Y. Weights
C are applied by multiplying all matrix terms by the weight. D. A. Femec,
C December 8, 1993.
C
C      IMPLICIT NONE
C      INTEGER*4 NPTS, NTERMS, nsets
C      REAL*8 X(NTERMS,*), Y(*)
C      REAL*8 SIGMAY(*)
C      REAL*8 A(*)
C      REAL*8 SUMX(19), !Squares and cross products that make up X'X.
C      1 SUMY(10), !Sums that make up X'Y (also called Q).
C      2 ARRAY(10,10) !Plays various roles.
C      REAL*8 XI, YI, XTERM, YTERM
C      REAL*8 predcY, stdevY, stdevA(*)
C      REAL*8 DETERM, DELTA
C      INTEGER*4 NMAX
C      INTEGER*4 I, J, K, L, N, jj, kk
C
C ACCUMULATE WEIGHTED SUMS
C
C      do i=1,nterms !First do the SUMX terms.
C      do j=1,nterms
C          n=(i-1)*nterms+j !This general indexer method is used throughout.
C          sumx(n)=0.0

```

```

do k=1,nsets
  sumx(n)=sumx(n)+x(i,k)*x(j,k)/(sigmay(k)**4)    !1/sig^2 weighting for both terms.
enddo
enddo
enddo
do i=1,nterms    !Then do the SUMY terms.
  sumy(i)=0.0
  do j=1,nsets
    sumy(i)=sumy(i)+x(i,j)*y(j)/(sigmay(j)**4)    !1/sig^2 weighting for both terms.
  enddo
enddo

```

```

C
C CONSTRUCT MATRICES AND CALCULATE COEFFICIENTS
C

```

```

DO J = 1, NTERMS
  DO K = 1, NTERMS
    N=(j-1)*nterms+k
    ARRAY(J,K) = SUMX(N)    !Here ARRAY is X'X.
  ENDDO
ENDDO
DELTA = DETERM (ARRAY, NTERMS)
IF (DELTA .EQ. 0.0) THEN
  DO J = 1, NTERMS
    A(J) = 0.0
  ENDDO
  GO TO 80
ELSE
  DO L = 1, NTERMS
    DO J = 1, NTERMS
      DO K = 1, NTERMS
        N=(j-1)*nterms+k
        ARRAY (J, K) = SUMX(N)    !Here ARRAY is X'X with the
                                !appropriate column replaced
      ENDDO
      ARRAY(J,L) = SUMY(J)        !with the Y values.
    ENDDO
    A(L) = DETERM (ARRAY, NTERMS) /DELTA
  ENDDO
ENDIF

```

```

C
C Now calculate the standard deviations.
C

```

```

stdevY=0.0    !Between the predicted Ys and the observed Ys (s)
do i=1,nsets
  predcY=0.0
  do j=1,nterms
    predcY=predcY+a(j)*x(j,i)
  enddo

```

```

        stdevY=stdevY+(y(i)-predcY/sigmaY(i))**2
    enddo

C
C The standard method for calculating s does not treat the case where NSETS
C equals NTERMS. Since dividing by NSETS-NTERMS serves to reduce the size
C of s, dividing by one instead should yield an overestimate of the proper
C value for s; and this is surely preferable to underestimating s.
C
    if(nsets.eq.terms)then
        stdevY=sqrt(stdevY)
    else
        stdevY=sqrt(stdevY/(nsets-terms))
    endif
    do i=1,terms !For the coefficients (s*sqrt(c(ii))).
    DO J = 1, NTERMS !Repack ARRAY with only those elements needed
        DO K = 1, NTERMS !for calculating the appropriate cofactor c(ii).
            if((j.ne.i).and.(k.ne.i))then !Exclude the ith row and column.
                N =(j-1)*terms+k
                if(j.gt.i)then !Shift the rows as needed.
                    jj=j-1
                else
                    jj=j
                endif
                if(k.gt.i)then !Shift the columns as needed.
                    kk=k-1
                else
                    kk=k
                endif
                ARRAY(JJ,KK) = SUMX(N) !Here ARRAY is the appropriate minor of X'X.
            endif
        ENDDO
    ENDDO
    stdevA(i)=stdevY*sqrt(determ(array,(terms-1))/delta)
    enddo
80  CONTINUE
    RETURN
    END

C
C Real*8 function for calculating the determinant of a square matrix by taking
C the product of the diagonal terms of the diagonalized form of the matrix.
C
    REAL*8 FUNCTION DETERM (ARRAY, NORDER)
C
C PURPOSE TO CALCULATE THE DETERMINANT OF A SQUARE MATRIX
C
C PARAMETERS
C   ARRAY      MATRIX ELEMENTS

```

```

C      NORDER      ORDER OF DETERMINANT
C
      IMPLICIT NONE
      REAL*8 ARRAY(10,10)
      INTEGER*4 NORDER
      INTEGER*4 I, J, K
      INTEGER*4 K1
      REAL*8 SAVE
C
      DETERM = 1.0
      DO K = 1, NORDER
C
C      INTERCHANGE COLUMNS IF DIAGONAL ELEMENT IS ZERO
C
          IF (ARRAY(K,K) .EQ. 0.0) THEN
              DO J = K, NORDER
                  IF (ARRAY(K,J) .NE. 0.0) GO TO 31
              ENDDO
              DETERM = 0.0
              GO TO 60
31          DO I = K, NORDER
              SAVE = ARRAY(I,J)
              ARRAY(I,J) = ARRAY(I,K)
              ARRAY(I,K) = SAVE
              ENDDO
              DETERM = -DETERM
C
C      SUBTRACT ROW K FROM LOWER ROWS TO GET DIAGONAL MATRIX
C
          ELSE
              DETERM = DETERM * ARRAY(K,K)
              IF (K .NE. NORDER) THEN
                  K1 = K + 1
                  DO I = K1, NORDER
                      DO J = K1, NORDER
                          ARRAY(I,J) = ARRAY(I,J) -
2                          ARRAY(I,K)*ARRAY(K,J)/ARRAY(K,K)
                      ENDDO
                  ENDDO
              ENDIF
          ENDIF
      ENDDO
60      CONTINUE
C
      RETURN
      END

```


C.11 SIGFIG Function

character*(*) function sigfig(value,error)

C
C Function to return a character-string representation in scientific
C notation of a value and its associated uncertainty (error) that
C contains the proper number of significant figures (limited to a total
C of ndigits digits plus a decimal point). The scientific notation form
C for character-string representation is (<real number> +/- <real
C number>)E<integer>; the overstruck plus-or-minus sign is not a
C FORTRAN character and so cannot be used.
C
C The following rules are used to arrive at the representation:
C (1) all digits to be dropped on rounding are considered in
C determining the direction in which to round, and so the whether
C the preceding digit is odd or even is not taken into account.
C (2) uncertainties with one as the sole significant digit will be
C expanded to include the digit following the one.
C (3) if the uncertainty is at least an order of magnitude larger
C than the input value, the value in the representation will be
C zero.
C
C Coded by D. A. Femec, May 26, 1992.
C
C Corrected so that value has its original sign on return. D. A. Femec,
C August 4, 1992.
C
C Added code to decrease LEXPON by one when error is slightly larger
C than value; in this instance, LEXPON had always been one too large.
C Also added a test of the converted error to help check for incorrect
C conversions. D. A. Femec, August 7, 1992.
C
C Modified to handle errors of zero by resetting ERROR to VALUE *
C 10.0E-08 so that all 0s will be displayed in the error portion of
C the returned representation. D. A. Femec, August 27, 1993.
C
C Modified analogously to the above to handle values of zero by resetting VALUE
C to ERROR * 10.0E-08 so that 0.00... will be displayed in the value portion of
C the returned representation. Also modified so that if both ERROR and VALUE
C are zero, SIGFIG will fail with an appropriate error message since there is
C no meaningful information to be displayed. D. A. Femec, January 20, 1994.
C
C Declaration statements.
C
C implicit real*4 (a-h,o-z)
C character*80 blanks

```

        character*1 spaces(80)/80*' '/
        equivalence (spaces(1),blanks(1:1))
C
C Beginning of the function.
C
C Determine ndigits from the length of sigfig.
C
        lcharp=len(sigfig)
        ndigits=(lcharp-12)/2
C
C Save the sign of the value, and make sure the numbers passed on are
C positive.
C
        svalue=sign(1.0,value)
        value=abs(value)
        error=abs(error)
        if((value.eq.0.0).and.(error.eq.0.0))then
            print *,' '
            print *,' *** SIGFIG FAILURE - BOTH VALUE AND ERROR ARE ',
            'ZERO ***'
            print *,' *** PLEASE DELETE ANY FILES CREATED DURING ',
            'EXECUTION ***'
            print *,' *** EXECUTION OF PROGRAM WILL NOW BE ABORTED ***'
            print *,' '
            call exit
        endif
        if(error.eq.0.0)error=value*10.0e-08    !So all 0s will be displayed.
        if(value.eq.0.0)value=error*10.0e-08    !So all 0s will be displayed.
C
C Calculate the logarithms of the value and its uncertainty, then obtain
C the integer value that correctly reflects the exponential term in a
C scientific notation (negative values need to be increased in magnitude
C by one).
C
        lvalue=int(log10(value))
        if(log10(value).lt.0.0)lvalue=lvalue-1
        lerror=int(log10(error))
        if(log10(error).lt.0.0)lerror=lerror-1
C
C Divide the value and its uncertainty by the most negative of the
C two above integer values for the exponents, then by an additional
C ten, to obtain the numbers to be rounded and then used for the
C character representation.
C
        minexp=min(lvalue,lerror)
        evalue=value/10.0**(minexp-1)
        eerror=error/10.0**(minexp-1)

```

```

C
C Round the numbers (first, if the fractional portion is greater than or
C equal to 0.5, increase the magnitude of the respective number by one;
C then, truncate the number).
C
    if(mod(evalue,1.0).ge.0.5)evalue=evalue+1.0
    rvalue=aint(evalue)/10.0**(lvalue-minexp+1)
    if(mod(eerror,1.0).ge.0.5)eerror=eerror+1.0
    rerror=aint(eerror)/10.0**(lerror-minexp+1)
    if((error/10.0**lerror).ge.1.95)then
        if(mod(rerror,1.0).ge.0.5)rerror=rerror+1.0
        rerror=aint(rerror)
        nadcml=0
    else !For one as leading error digit.
        nadcml=1
    endif
    rerror=rerror/10.0**(lvalue-lerror)
C
C Construct the character representation.
C
C Reset the character representation.
C
    sigfig='( '//blanks(1:ndigits)//' +/- '//blanks(1:ndigits)//')E+ '
C
C If the uncertainty is at least an order of magnitude larger than
C the value use a value number of zero and set the exponent for
C the character representation to the integer exponent of the
C uncertainty. Otherwise, use the integer exponent for the value.
C
    if((error/value).gt.10.0)then !Force the value to be zero.
        rvalue=0.0
        lexpon=lerror
        rerror=rerror/10.0**(lerror-lvalue)
    else
        if(value.ge.error)then
            lexpon=lvalue
            nadcml=nadcml+abs(lvalue-lerror)
        else
            if(lvalue.ne.lerror)then
                lexpon=lerror-1
            else
                lexpon=lerror
            endif
            if(rerror.gt.1.95)then
                nadcml=0
            else
                nadcml=1 !Include the tenths place.
            endif
        endif
    endif

```

```

endif
endif
endif
C
C Now check the significant figures error against the error passed to
C the function; flag as fault the conversion if the converted error
C deviates from the error passed by more than fifty percent.
C
if(abs(((error*10.0**lexpon)-error)/error).gt.0.5)then
  print *,' '
  print *,' *** SIGFIG FAILURE - EXPONENT INCORRECTLY ',
  'CONVERTED ***'
  print *,'      Original value =',(svalue*value),
  ' ; original error =',error
  print *,'      ERROR =',error,' ; LEXPON =',lexpon
  print *,' *** PLEASE DELETE ANY FILES CREATED DURING ',
  'EXECUTION ***'
  print *,' *** EXECUTION OF PROGRAM WILL NOW BE ABORTED ***'
  print *,' '
  call exit
endif
C
C Write the exponent, value, and uncertainty into the character
C representation.
C
if(lexpon.lt.0)sigfig((lcharp-2):(lcharp-2))='- ' !Minus sign if needed.
write(sigfig((lcharp-1):(lcharp)),fmt='(i2.2)')abs(lexpon) !Leading 0.
if(nadcml.gt.(ndigts-2))nadcml=ndigts-2 !Maximum after the decimal point.
nbdcm1=1 !Start with one digit before the decimal point.
100 write(sigfig(3:(2+ndigts)),fmt='(f<ndigts>.<nadcml-nbdcm1+1>)',
. err=110)rvalue
if(svalue.lt.0.0)then !Place minus sign if needed.
  isign=2
105 if(sigfig((isign+1):(isign+1)).eq.' ')then
  isign=isign+1
  goto 105
endif
sigfig(isign:isign)='- '
endif
write(sigfig((8+ndigts):(7+2*ndigts)),
. fmt='(f<ndigts>.<nadcml-nbdcm1+1>)',err=110)error
goto 120 !Successful internal writes of value and uncertainty.
110 nbdcm1=nbdcm1+1 !Try another digit before the decimal point.
if((nadcml-nbdcm1+1).lt.0)then
  print *,' '
  print *,' *** SIGFIG FAILURE - CANNOT FIT NUMBERS IN FIELDS ',
  'PROVIDED ***'

```

```

        print *, '      Original value =', (svalue*value),
        ' ; original error =', error
        print *, '      NDIGTS =', ndigits, ' ; NADCML =', nadcml,
        ' ; last NBDCML =', nbdcm1
        print *, ' *** EXECUTION OF PROGRAM WILL NOW BE ABORTED ***'
        print *, ' *** PLEASE DELETE ANY FILES CREATED DURING ',
        'EXECUTION ***'
        print *, ' '
        call exit
    endif
    goto 100
120 value=svalue*value    !Return value with its original sign.
    return
C
C End of the function.
C
    end

```

C.12 BR_UC Function

```

C FUNCTION TO EXTRACT THE BRANCHING RATIO UNCERTAINTY FOR A NUCLIDE
C ROUTINE ASSUMES THE VALUES IN THE STRUCTURE 'BRANCH.INC' HAS BEEN INITIALIZED
C
C      REAL*4      FUNCTION BR_UC (A, Z, ID, ENERGY)
C
C WHERE: A= NUCLIDE AW
C       Z= NUCLIDE AN
C       ID= NUCLIDE CHEMICAL SYMBOL
C       ENERGY=LINE ENERGY OF NUCLIDE
C
C      INCLUDE 'BRANCH.INC'
C      COMMON /BRUC/NOB, BRU
C
C      INTEGER*4   A, Z
C      CHARACTER*4 ID
C      REAL*4      ENERGY
C
C      DO I=1,NOB
C         IF((BRU.E(I).A .EQ. A) .AND.
C1          (BRU.E(I).Z .EQ. Z) .AND.
C2          (BRU.E(I).ID .EQ. ID)) THEN
C            DO II=1, BRU.E(I).NR
C              IF (ABS(ENERGY-BRU.E(I).ENERGY(II)) .LT. 0.3) THEN
C                BR_UC=BRU.E(I).BR_UC(II)
C              RETURN
C            END DO
C          END IF
C        END DO

```

```

                ENDIF
            ENDDO
            BR_UC=-1.0
        ENDIF
    ENDDO
    BR_UC=-1.0
    RETURN
END

```

C.13 WAVG_O Subroutine

```

subroutine wavg_o(values,errors,ndatum,ifrchi,mean,stdev,wmean,
. wstdev,rchisq,ifvald)

```

```

C
C Subroutine to calculate unweighted and weighted means and standard deviations
C of the means of the values passed, and then, if requested, perform a reduced
C chi-squared outlier test on the weighted values to determine if there are any
C outliers to be rejected, as per the procedure described in the VAXGAP manual
C (E. W. Killian and J. K. Hartwell, "VAXGAP: A Code for the Routine Analysis
C of Gamma-Ray Pulse-Height Spectra on a VAX Computer," EGG-2533, May, 1988)
C prior to returning the acceptable mean and standard deviation of the mean
C (and the final reduced chi-squared value if outlier testing was requested -
C zero will be returned otherwise - along with a vector indicator of which data
C points were used).
C
C The final means and standard deviations are calculated as follows, where xi
C represents individual values, wi the weight associated with xi, and N the
C number of individual values, and the corresponding mean is used in each
C standard deviation calculation:
C
C   unweighted mean = sum(xi) / N (i.e., all wi = 1),
C   unweighted standard deviation = sqrt( sum((xi-mean)^2) / (N-1)), or
C                                   = sqrt( (N*sum(xi^2) - sum(xi)^2) /
C                                       (N*(N-1)));
C
C   weighted mean = sum(wi*xi) / sum(wi) and
C   weighted standard deviation = the greater of
C                               sqrt( sum(wi*(xi-mean)^2) /
C                                   (sum(wi)*(N-1)) ),
C                               = sqrt( (sum(wi*xi^2) -
C                                   2*mean*sum(wi*xi) +
C                                   mean^2*sum(wi)) /
C                                   (sum(wi)*(N-1)) ), or
C                               sqrt( 1/sum(wi) ).
C
C The weighted standard deviation for purposes of outlier testing is calculated

```

```

C as follows:
C
C   weighted standard deviation = sqrt( sum(wi^2*(xi-mean)^2) /
C   (note that the "lost"          sum(wi^2)), or
C   degree of freedom is    = sqrt( (sum(wi^2*xi^2) -
C   neglected - what it      2*mean*sum(wi^2*xi) +
C   represents is not clear)  mean^2*sum(wi^2)) /
C                               sum(wi^2)).
C
C Written by D. A. Femec, January 11, 1994.
C
C Modified to take the absolute value of standard deviation-related square root
C arguments before taking the square root.  D. A. Femec, February 8, 1994.
C
C Modified to perform a final calculation of the standard deviation of the
C weighted mean that conforms to ANSI standard N42.14-1991; specifically, the
C standard deviation of the weighted mean is the greater of sqrt(1/sum wi) or
C sqrt[(sum wi(xi-mean)^2)/((N-1)*sum wi)]; the method used in outlier testing
C has been retained for testing only.  Increased the precision of the sum-
C containing variables from REAL*4 to REAL*8.  D. A. Femec, February 24, 1994.
C
C Declaration statements.
C
C   implicit none      !For safety's sake.
C   integer*4 i,j,k,    !Common "DO" loop variables.
C   . nvalus,ndatum     !The number of data points passed in VALUES and ERRORS.
C   real*4 values(ndatum),errors(ndatum),    !Passed values and errors.
C   . mean,stdev,wmean,wstdev,    !Unweighted and weighted results.
C   . weight,           !The weight of a single data point.
C   . rchisq            !The reduced chi-squared value.
C   real*8 sumv,sumvv,wsumv,wsumvv,    !Value-related sums.
C   . sumw,wsumw,wsumww,wsumvw,wsumwv,  !Weight-related sums.
C   . rejst            !For testing whether a previously valid data point should be rejected.
C   integer*2 irejct    !Pointer to the previously valid data point to be rejected.
C   logical ifrchi,     !Should the reduced chi-squared outlier test be applied?
C   . ifvald(ndatum)    !Is the corresponding data point valid (i.e., not an outlier)?
C
C Beginning of the routine.
C
C Assume, initially, that no outlier testing is being performed, and that all
C data points are valid.
C
C   rchisq=0.0
C   do i=1,ndatum
C       ifvald(i)=.true.
C   enddo
C

```

C If only one VALUES and ERRORS were passed, use them as the results.

```
C
  if(ndatum.eq.1)then
    mean=values(1)
    stdev=errors(1)
    wmean=values(1)
    wstdev=errors(1)
    return
  endif
```

C
C Obtain the needed sums after initializing all of the stored sums.

```
C
  nvalus=ndatum  !NVALUS may change with outlier testing.
100 sumv=0.0      !Sum of values (unweighted)
  sumvv=0.0      !Sum of values squared (unweighted)
  wsumv=0.0      !Sum of weighted values (one-over-sigma-squared weighting).
  wsumvv=0.0     !Sum of weighted values squared.
  sumw=real(nvalus) !Sum of unweighted weights (number of data points).
  wsumw=0.0      !Sum of weights.
  wsumww=0.0     !Sum of weights squared.
  wsumvw=0.0     !Sum of doubly weighted values.
  wsumvv=0.0     !Sum of values squared times weights.
  do i=1,ndatum  !Calculate the requisite sums.
    if(ifvald(i))then !Use only valid data points.
      sumv=sumv+values(i)
      sumvv=sumvv+values(i)**2
      weight=1.0/errors(i)**2
      wsumv=wsumv+weight*values(i)
      wsumvv=wsumvv+(weight*values(i))**2
      wsumw=wsumw+weight
      wsumww=wsumww+weight**2
      wsumvw=wsumvw+(weight**2)*values(i)
      wsumvv=wsumvv+weight*(values(i)**2)
    endif
  enddo
```

C
C Calculate the means and standard deviations.

```
C
  mean=sumv/sumw
  stdev=sqrt(abs((sumw*sumvv-sumv*sumv)/(sumw*(sumw-1.0))))
  wmean=wsumv/wsumw
  wstdev=sqrt(abs((wsumvv-2*wmean*wsumvw+wsumww*wmean**2)/wsumww))
```

C
C Perform the outlier test, if requested, as per the VAXGAP manual, by first
C calculating the reduced chi-squared value. Otherwise, return to the calling
C routine.

C


```

if(ifrchi)then
  rchisq=0.0
  nvalus=0
  do i=1,ndatum  !Calculate the reduced chi-squared.
    if(ifvald(i))then  !Use only valid data points.
      rchisq=rchisq+((values(i)-wmean)/errors(i))**2
      nvalus=nvalus+1
    endif
  enddo
  if(nvalus.gt.3)then
    rchisq=rchisq/real(nvalus-1)
  else
    rchisq=rchisq/real(nvalus)
  endif
endif

```

C
 C If the reduced chi-squared value is less than four, the set of currently
 C valid data points is considered to be a consistent set and there are no
 C outliers. The final form of the standard deviation of the weighted mean
 C is calculated as per the ANSI standard.

```

    if(rchisq.le.4.0)then
      wstdev=max(sqrt(abs((wsumvv-2*wmean*wsumv+wsumw*wmean**2)/
        (wsumw*(sumw-1.0)))),abs(sqrt(1.0/wsumw)))
      return
    endif

```

C
 C Find the worst outlier and then mark it as an invalid data point.

```

    rejst=0.0
    do i=1,ndatum
      if(ifvald(i))then  !Test only previously valid data points.
        if(abs((values(i)-wmean)/wstdev).gt.rejst)then  !Worst so far.
          rejst=abs((values(i)-wmean)/wstdev)
          irejct=i
        endif
      endif
    enddo
    ifvald(irejct)=.false.
    nvalus=nvalus-1

```

C
 C If only one valid data point remains after the outlier test, use its value
 C and error as the results.

```

    if(nvalus.eq.1)then
      do i=1,ndatum
        if(ifvald(i))then  !Find the lone valid data point.
          mean=values(i)

```

```

        stdev=errors(i)
        wmean=values(i)
        wstdev=errors(i)
        return
    endif
enddo
endif
C
C   Otherwise, obtain the sums for the new set of valid data points before
C   performing another cycle of outlier testing.
C
C       goto 100
C   endif
C
C End of the WAVG_O routine. Calculate the final form of the standard
C deviation of the weighted mean as per the ANSI standard.
C
C       wstdev=max(sqrt(abs((wsumvv-2*wmean*wsumv+wsumw*wmean**2)/
C   . (wsumw*(sumw-1.0))))),abs(sqrt(1.0/wsumw)))
C       return
C   end

```

C.14 DPY_REGION Subroutine

```

C ROUTINE MODIFIED TO INCLUDE THE PEAKS WITH EVEN FIT FLAGS FOR GAUSS FIT
C AND THEN INSERT THE ODD FIT FLAGS FOR THE NON LINEAR FITS. THIS FIX MAKES
C THE ROUTINE FIT THINGS IN THE SAME ORDER AS THE ORIGINAL ANALYSIS
C       EWK 11/4/93
C ROUTINE TO CREATE A HARD COPY PLOT OF THE RESULTS OF FITS MADE IN AN ENERGY
C REGION. ROUTINE EXTRACTS FROM THE PKD ARRAY THOSE PEAKS WHOSE ENERGIES FALL
C BETWEEN THE SPECIFIED REGION AND LOADS THOSE PEAK POSITIONS INTO A SMALLER
C PKD ARRAY AND CALLS ROUTINES XTP_FIT_GAUSS AND XTP_FIT_LINEAR WHICH REFIT
C THE PEAKS AND GENERATES THE FIT PARAMETERS.
C       EWK 4/1/93
C
C Modified to include a flag page with the output sent to XTP$QUE if this
C routine is called from batch mode (such as when it is executed as part of
C the routine BATCH_CBAT). D. A. Femec, April 20, 1993.
C
C Modified to immediately print the output rather than exiting whenever "REGION
C OF INTEREST CONTAINS ALL ZEROS" occurs (to prevent premature termination of
C batch analyses calling this routine and so as to provide some record of where
C this condition was encountered). D. A. Femec, September 10, 1993.
C
C Commented out (with CJPI) the code related to including a flag page with the

```

C output sent to XTP\$QUE if this routine is called from batch mode for purposes
 C of SWEPP. D. A. Femec, June 22, 1994.

C

SUBROUTINE DPY_REGION(BENG,EENG)

C WHERE: BENG= BEGINING ENERGY

C EENG= ENDING ENERGY

C

INCLUDE '(\$SYSSRVNAM)'

INCLUDE '(\$SSDEF)'

INCLUDE '(\$LNMDEF)'

INCLUDE '(\$SJCDEF)'

INCLUDE '(\$JBCMSGDEF)'

CJPI include '(\$jpi) !JPI system service definitions.

INCLUDE 'SYS\$LIBRARY:GKSDEFS.FOR'

INCLUDE 'GAP\$SRC:SPEC_PLOT.INC'

INCLUDE 'GAP\$SRC:PLOT.INC'

INCLUDE 'GAP\$SRC:PARAM_BLOCK.INC'

C

INCLUDE 'GAP\$SRC:VHEAD.INC'

INCLUDE 'GAP\$SRC:PKD.INC'

STRUCTURE /ITEMLIST/

INTEGER*2 BUFLTH !BUFFER LENGTH

INTEGER*2 ITMCDE !ITEM CODE

INTEGER*4 BUFADR !BUFFER ADDRESS

INTEGER*4 RTNLTH !RETURN LENGTH ADDRESS

END STRUCTURE

C

RECORD /ITEMLIST/ CREITM(2)

RECORD /ITEMLIST/ SNDITM(7)

CJPI record /ITEMLIST/ jpiitm(2) !Item list for SYS\$GETJPI call to obtain execution mode.

CJPI integer*4 jpimod

C

C STRUCTURE USED TO TRANSLATE LOGICAL'S WITH A CALL TO SYS\$TRNLNM

STRUCTURE /ITEM_LIST/

INTEGER*2 BL

INTEGER*2 ITEM

INTEGER*4 ADRESS

INTEGER*4 RET

INTEGER*4 END /0/

END STRUCTURE

RECORD /ITEM_LIST/IL

INTEGER*4 LQUE_LN

C

```

CHARACTER*9 DEV_LOGICAL /'GKS$CONID'/
CHARACTER*15 DEV_NAME
CHARACTER*11 LNTAB /'LNM$PROCESS'/
CHARACTER*512 BHEAD
INTEGER*4 DTIME(2) /-20000000,-1/
INTEGER*4 I
REAL*4 BIG, SMALL, VALUE
CHARACTER*30 LQUEUE
CHARACTER*7 SETUP_NAME /'LASER80'/
INTEGER*4 PSTART, PEND
INTEGER*4 IOSB(2)

```

C

```

CHARACTER*80 SDAT_REC
INTEGER*4 WKID / 1 /
INTEGER*4 ERR_STATUS
INTEGER*4 DUMMY_INT
INTEGER*4 DEF_MODE, REG_MODE
INTEGER*4 IN_DEV / 1 /
INTEGER*4 IN_STATUS
INTEGER*4 WSTYPE
INTEGER*4 WSMASK /'FFFF'X/
INTEGER*4 OP_MODE, ECHO_FG, RTRN_LNG
INTEGER*4 SYS_STATUS
REAL*4 DMAX_X, DMAX_Y
REAL*4 DXP
CHARACTER*1 C_ONE /'1'/
CHARACTER*2 C_TEN /'10'/
CHARACTER*2 C_ZERO /'0'/
CHARACTER*1 C_DIGIT
CHARACTER*6 C_GRID
CHARACTER*75 SPID_HD

```

C

```

INTEGER*4 STDPAGE /1/
INTEGER*4 N,J,NPF
INTEGER*4 NINTS, MINTS
INTEGER*4 CINDEX

```

C

```

REAL*4 XRECT(5) /PLOWX, PHIGHX, PHIGHX, PLOWX, PLOWX/
REAL*4 YRECT(5) /PLOWY, PLOWY, PHIGHY, PHIGHY, PLOWY/
REAL*4 XGRID(4) /PGRIDHX, PGRIDLX, PGRIDLX, PGRIDHX/
REAL*4 YGRID(4) /PGRIDHY, PGRIDHY, PGRIDLY, PGRIDLY/
REAL*4 XTIC(2)
REAL*4 YTIC(2)
REAL*4 DELTA_Y, DELTA_X, DX
REAL*4 XX, YY
REAL*4 F
REAL*4 DTEXT, ETEXT

```

```

REAL*4 XORIGIN, XLENGTH, YORIGIN, YLENGTH
INTEGER*4 NORMDATA /2/
INTEGER*4 SCHAN, ECHAN, NCHAN
REAL*4 POINT_SIZE
REAL*4 X(2), Y(2)
REAL*4      XCHAN(0:8191)
REAL*4      CMX, CMY, EMX, EMY, AMX, AMY, PMX, PMY
CHARACTER*8 OUTPUT
CHARACTER*7 CSTG, ESTG
CHARACTER*8 ASTG
CHARACTER*4 PSTG

```

C

```

C NOW PLOT RESULTS
NPF=VHEAD(NP)

```

C

```

C DETERMINE THE START AND END INDEX OF THE PLOT
C ISX START INDEX , IEX END INDEX
C

```

```

ISX=1
IEX=NPF
NR=0
DO I=1,NPF
    IF(PKD(I,3,ERA) .GE. BENG) THEN
        ISX=I
        GOTO 175
    ENDIF
ENDDO

```

```

175 DO J=ISX+1,NPF
    IF(PKD(J,3,ERA) .LE. EENG) THEN
        NR=NR+1
    ELSE
        IEX=J-1
        GOTO 180
    ENDIF
ENDDO

```

C

```

C GET HEADER INFORMATION
C

```

C

```

180 READ (1'1) BHEAD
PLOTDREC.SPCID = P.MCA_SYSTEM(1:2) // BHEAD(12:20)
PLOTDREC.HEADER = BHEAD(61:135)

```

C

```

PLOTDREC.NSPEC=P.NO_CHANNELS-1
PLOTDREC.PTYPE=1 !POINT PLOT
PLOTDREC.XUNITS=1 !CHANNEL

```

```

PLOTDREC.START=PT(ISX).EVSC - 5
PLOTDREC.END=PT(IEY).ENDC + 5
PLOTDREC.CTSCALE=1      !LOG PLOT
PLOTDREC.NPASS=1

```

```

      DEV_NAME = 'SPECTRA.PLT'
      PLOTDREC.WTYPE = GKS$K_LN03_PLUS .OR. GKS$M_PAPERSIZE_A
2      .OR. GKS$M_LANDSCAPE

```

```

C
C CREATE LOGICAL NAME FOR PLOT DEVICE FOR GKS
C

```

```

      CREITM(1).BUFLTH = INDEX(DEV_NAME, ' ') - 1
      CREITM(1).ITMCDE = LNM$_STRING
      CREITM(1).BUFADR = %LOC(DEV_NAME)
      CREITM(1).RTNLTH = 0

```

```

C
      CREITM(2).BUFLTH = 0
      CREITM(2).ITMCDE = 0

```

```

C
      SYS_STATUS = SYS$CRELNM
2      (, %DESCR(LNTAB), %DESCR(DEV_LOGICAL),, CREITM)
      IF ((SYS_STATUS.NE. SS$_NORMAL) .AND.
2      (SYS_STATUS.NE. SS$_SUPERSEDE)) THEN
          CALL LIB$SIGNAL (%VAL(SYS_STATUS))
      ENDIF

```

```

C
C
C CHECK LIMITS
C

```

```

      PSTART = PLOTDREC.START
      PEND = PLOTDREC.END
      IF (PEND .GT. PLOTDREC.NSPEC) PEND = PLOTDREC.NSPEC

```

```

C
C DETERMINE MAJOR TICK PLACEMENT ON X-AXIS
C

```

```

      PLOTDREC.XRANG = (PEND - PSTART + PLOTDREC.NPASS)/PLOTDREC.NPASS
      IF (PLOTDREC.XRANG .GE. 5000) THEN
          PLOTDREC.IMAJT = 2000
      ELSE IF (PLOTDREC.XRANG .GE. 3000) THEN
          PLOTDREC.IMAJT = 1000
      ELSE IF (PLOTDREC.XRANG .GE. 1500) THEN
          PLOTDREC.IMAJT = 500
      ELSE IF (PLOTDREC.XRANG .GE. 750) THEN
          PLOTDREC.IMAJT = 250
      ELSE IF (PLOTDREC.XRANG .GE. 600) THEN
          PLOTDREC.IMAJT = 200

```

```

ELSE IF (PLOTDREC.XRANG .GE. 300) THEN
  PLOTDREC.IMAJT = 100
ELSE IF (PLOTDREC.XRANG .GE. 150) THEN
  PLOTDREC.IMAJT = 50
ELSE IF (PLOTDREC.XRANG .GE. 100) THEN
  PLOTDREC.IMAJT = 20
ELSE IF (PLOTDREC.XRANG .GE. 10) THEN
  PLOTDREC.IMAJT = 5
ELSE
  PLOTDREC.IMAJT = 2
ENDIF

C
C SET XRANG TO MULTIPLE OF IMAJT
C
  PLOTDREC.XRANG = ((PLOTDREC.XRANG/PLOTDREC.IMAJT)+1)
  2 * PLOTDREC.IMAJT
C
C SET START TO MULTIPLE OF IMAJT
C
  PSTART = (PSTART/PLOTDREC.IMAJT) * PLOTDREC.IMAJT
C
C SET END TO SIZE OF PLOT
C
  PEND = PSTART + (PLOTDREC.XRANG * PLOTDREC.NPASS) - 1
  IF (PEND .GT. PLOTDREC.NSPEC) PEND = PLOTDREC.NSPEC
C
C DETERMINE START AND END CHANNELS FOR EACH PASS
C
  DO I = 1, PLOTDREC.NPASS
    IF (I .EQ. 1) THEN
      PLOTDREC.XPASS (1,I) = PSTART
    ELSE
      PLOTDREC.XPASS (1,I) = PSTART + ((I-1) * PLOTDREC.XRANG)
    ENDIF
    IF (I .EQ. PLOTDREC.NPASS) THEN
      PLOTDREC.XPASS (2,I) = PEND
    ELSE
      PLOTDREC.XPASS (2,I) = PSTART + (I * PLOTDREC.XRANG) - 1
    ENDIF
    PLOTDREC.XPASS (3,I) = PLOTDREC.XRANG
  ENDDO
C
  BIG = 0.0
  SMALL = 1.0E+10
  IBEG = PLOTDREC.XPASS (1,1)
  IEND = PLOTDREC.XPASS (2,PLOTDREC.NPASS)
  DO I = IBEG, IEND

```

```

        VALUE = FGET(I)
        IF (VALUE .LT. SMALL) SMALL = VALUE
        IF (VALUE .GT. BIG) BIG = VALUE
    ENDDO
    IF (SMALL .EQ. 0.0) THEN
        PLOTDREC.IDEC = 0
    ELSE
        PLOTDREC.IDEC = IFIX(LOG10(SMALL))
    ENDIF
    IF (BIG .EQ. 0.0) THEN
        WRITE (*,8015)
        goto 1000 !Print the output.
    ENDIF
8015    FORMAT (' REGION OF INTEREST CONTAINS ALL ZEROS')
        PLOTDREC.NDEC = IFIX(LOG10(BIG)) - PLOTDREC.IDEC + 1
C
C ADJUST THE DATA FOR LOG PLOTS
C
    SCHAN = PLOTDREC.XPASS(1,1)
    ECHAN = PLOTDREC.XPASS(2,PLOTDREC.NPASS)
    IF (SCHAN .LT. 0) SCHAN = 0
    IF (ECHAN .GT. PLOTDREC.NSPEC) ECHAN = PLOTDREC.NSPEC

    DO I = SCHAN, ECHAN
        VALUE = FGET(I)
        IF (VALUE .LE. 0.0) THEN
            PLOTDREC.SDATA(I) = -1.0
        ELSE
            PLOTDREC.SDATA(I) = LOG10(VALUE)
        ENDIF
    ENDDO
C
C CREATE THE PLOT
C INITIALIZE THE GKS SYSTEM,
C OPEN AND ACTIVATE WORKSTATION "WKID"
C
    CALL GKS$OPEN_GKS (%DESCR('SYS$ERROR'),)
    CALL GKS$OPEN_WS (WKID, GKS$K_CONID_DEFAULT, PLOTDREC.WTYPE)
    CALL GKS$ACTIVATE_WS (WKID)
C
C CHECK OUTPUT DEFERENTIAL AND REGENERATION MODES
C
    CALL GKS$INQ_DEF_DEFER_STATE (PLOTDREC.WTYPE, ERR_STATUS,
2      DEF_MODE, REG_MODE)
    IF (ERR_STATUS .NE. 0) THEN
        DEF_MODE = GKS$K_ASTI
        REG_MODE = GKS$K_IRG_SUPPRESSED
    
```



```

        ENDIF
C
C OUTPUT IMMEDIATELY AND SUPPRESS REGENERATIONS
C
        IF ((DEF_MODE .NE. GKS$K_ASAP) .OR.
            2 (REG_MODE .NE. GKS$K_IRG_SUPPRESSED)) THEN
            CALL GKS$SET_DEFER_STATE (WKID, GKS$K_ASAP,
            2 GKS$K_IRG_SUPPRESSED)
        ENDIF
C
C CHECK COLOR FACILITIES
C
        CALL GKS$INQ_COLOR_FAC (PLOTREC.WTYPE, ERR_STATUS, DUMMY_INT,
            2 PLOTREC.COLOR, DUMMY_INT)
        IF (ERR_STATUS .NE. 0) THEN
            PLOTREC.COLOR = GKS$K_MONOCHROME
        ENDIF
C
C DETERMINE THE PHYSICAL WORKSTATION SIZE
C
        CALL GKS$INQ_MAX_DS_SIZE (PLOTREC.WTYPE, ERR_STATUS, DUMMY_INT,
            2 DMAX_X, DMAX_Y, DUMMY_INT, DUMMY_INT)
        IF (ERR_STATUS .NE. 0) THEN
            DMAX_X = 1.0
            DMAX_Y = 1.0
        ENDIF
        PLOTREC.ASPECT_X = 1.0
        PLOTREC.ASPECT_Y = DMAX_Y / DMAX_X
C
C SET WORKSTATION WINDOW AND VIEWPOINT TO THE MAXIMUM RECTANGLE
C
        CALL GKS$SET_WS_WINDOW (WKID, 0.0, PLOTREC.ASPECT_X,
            2 0.0, PLOTREC.ASPECT_Y)
        CALL GKS$SET_WS_VIEWPORT (WKID, 0.0, DMAX_X, 0.0, DMAX_Y)
C
C SET TEXT FONT AND PRECISION
C
        WSTYPE = PLOTREC.WTYPE .AND. WSMASK
        IF (WSTYPE .EQ. GKS$K_LN03_PLUS) THEN
            CALL GKS$SET_TEXT_FONTPREC (-12, GKS$K_TEXT_PRECISION_STROKE)
        ELSE
            CALL GKS$SET_TEXT_FONTPREC (-2, GKS$K_TEXT_PRECISION_STROKE)
        ENDIF
C
C CREATE THE PLOT GRID
C
C

```

```

C
C*****
C
C WORLD COORDINATES WILL BE FOR 8 1/2 X 11 INCH PAPER
C IN LANDSCAPE MODE (X AXIS ON LONG EDGE)
C SET NORMALIZATION WINDOW FOR THIS SIZE
C
    CALL GKS$SET_WINDOW (STDPAGE, PLOWX, PHIGHX, PLOWY, PHIGHY)
C
C NDC VIEWPORT FOR HARDCOPY WILL LEAVE MARGIN AROUND PAGE
C USE SAME ASPECT AS WORKSTATION
C
    WSTYPE = PLOTDREC.WTYPE .AND. WSMASK
    XORIGIN = ((PLOWX-PLOWX+PMARGIN)/(PHIGHX-PLOWX+(2*PMARGIN)))
    2 *PLOTDREC.ASPECT_X
    XLENGTH = ((PHIGHX-PLOWX+PMARGIN)/(PHIGHX-PLOWX+(2*PMARGIN)))
    2 *PLOTDREC.ASPECT_X
    YORIGIN = ((PLOWY-PLOWY+PMARGIN)/(PHIGHY-PLOWY+(2*PMARGIN)))
    2 *PLOTDREC.ASPECT_Y
    YLENGTH = ((PHIGHY-PLOWY+PMARGIN)/(PHIGHY-PLOWY+(2*PMARGIN)))
    2 *PLOTDREC.ASPECT_Y
C
    IF (XORIGIN .LT. 0.0) XORIGIN = 0.0
    IF (XLENGTH .GT. PLOTDREC.ASPECT_X) XLENGTH = PLOTDREC.ASPECT_X
    IF (YORIGIN .LT. 0.0) YORIGIN = 0.0
    IF (YLENGTH .GT. PLOTDREC.ASPECT_Y) YLENGTH = PLOTDREC.ASPECT_Y
C
    CALL GKS$SET_VIEWPORT (STDPAGE, XORIGIN, XLENGTH, YORIGIN, YLENGTH)
    CALL GKS$SET_CLIPPING (GKS$K_CLIP)
C
C ENABLE TRANSLATION FROM WORLD TO NDC
C
    CALL GKS$SELECT_XFORM (STDPAGE)
C
C*****
C
C SELECT COLOR 1 FOR BACKGROUND
C
    IF (PLOTDREC.COLOR .EQ. GKS$K_COLOR) THEN
        CINDEX = 1
        CALL GKS$SET_PLINE_COLOR_INDEX (CINDEX)
        CALL GKS$SET_TEXT_COLOR_INDEX (CINDEX)
    ENDIF
C
C DRAW A RECTANGLE AROUND THE EDGE
C FIRST REDUCE THE SIZE OF MAX X AND Y SO LINE DOES NOT GET CLIPPED
C

```

```

XRECT(2) = PHIGHX - 0.005
XRECT(3) = PHIGHX - 0.005
YRECT(3) = PHIGHY - 0.005
YRECT(4) = PHIGHY - 0.005
CALL GKS$POLYLINE (5, XRECT, YRECT)
C
C CENTER TITLES USING DOUBLE SIZE CHARACTERS
C
DTEXT = PTXSIZE * 2.0
CALL GKS$SET_TEXT_ALIGN (GKS$K_TEXT_HALIGN_CENTER,
2 GKS$K_TEXT_VALIGN_BOTTOM)
CALL GKS$SET_TEXT_HEIGHT (DTEXT)
C
SPID_HD=' '
SPID_HD(1:11)=PLOTDREC.SPCID
SPID_HD(13:75)=PLOTDREC.HEADER(1:63)

DO I = 75, 12, -1
    IF (SPID_HD(I:I) .GT. ' ') GOTO 100
ENDDO
100 CONTINUE
C
    CALL GKS$TEXT (PCENTX, PHEADY, %DESCR(SPID_HD(1:I)))
C
C
    CALL GKS$TEXT (PCENTX, PUNITY, %DESCR('CHANNEL'))
C
C DRAW X-Y AXIS LINES
C
    CALL GKS$POLYLINE (4, XGRID, YGRID)
C
C LABEL Y AXIS
C
    IF (PLOTDREC.CTSCALE .EQ. 1) THEN
C
C LOG SCALE TIC MARKS
C
        DELTA_Y = (PGRIDHY - PGRIDLY) / PLOTDREC.NDEC
        XTIC(1) = PGRIDLX
        DO I = 0, PLOTDREC.NDEC-1
            XTIC(2) = XTIC(1) + PTICL
            YY = PGRIDLY + (FLOAT(I) * DELTA_Y)
            YTIC(1) = YY
            YTIC(2) = YY
            CALL GKS$POLYLINE (2, XTIC, YTIC)
            XTIC(2) = XTIC(1) + PTICS
        DO N = 1, 9

```

```

        YTIC(1) = YY + (LOG10(FLOAT(N)) * DELTA_Y)
        YTIC(2) = YTIC(1)
        CALL GKS$POLYLINE (2, XTIC, YTIC)
    ENDDO
ENDDO
C
C LOG SCALE NUMBERS
C
    ETEXT = PTXSIZE * 0.75
    CALL GKS$SET_TEXT_ALIGN (GKS$K_TEXT_HALIGN_LEFT,
2    GKS$K_TEXT_VALIGN_BASE)
    DO I = 0, PLOTDREC.NDEC
        YY = PGRIDLY + (FLOAT(I) * DELTA_Y)
        N = PLOTDREC.IDEC + I
        CALL GKS$SET_TEXT_HEIGHT (PTXSIZE)
        IF (N .EQ. 0) THEN
            XX = PLOWX + PTXSIZE
            CALL GKS$TEXT (XX, YY, %DESCR(C_ONE))
        ELSE
            XX = PLOWX
            CALL GKS$TEXT (XX, YY, %DESCR(C_TEN))
            WRITE (C_DIGIT, FMT='(I1)') N
            CALL GKS$SET_TEXT_HEIGHT (ETEXT)
            YY = YY + PTXSIZE
            XX = PLOWX + (PTXSIZE * 1.3)
            CALL GKS$TEXT (XX, YY, %DESCR(C_DIGIT))
        ENDIF
    ENDDO
ENDIF
C
C DRAW TIC MARKS ON X AXIS
C
    IF (PLOTDREC.XUNITS .EQ. 1) THEN
        NINTS = PLOTDREC.XRANG / PLOTDREC.IMAJT
        MINTS = 10
        IF (MINTS .GT. PLOTDREC.IMAJT) MINTS = PLOTDREC.IMAJT
        DELTA_X = (PGRIDHX-PGRIDLX) / FLOAT(NINTS)
        DX = DELTA_X / FLOAT(MINTS)
    ELSE
        NINTS = PLOTDREC.ERANG / PLOTDREC.EMAJT
        MINTS = 10
        IF (MINTS .GT. PLOTDREC.EMAJT) MINTS = PLOTDREC.EMAJT
        DELTA_X = (PGRIDHX-PGRIDLX) / FLOAT(NINTS)
        DX = DELTA_X / FLOAT(MINTS)
    ENDIF
C
C TOP X AXIS

```

```

C
YTIC(1) = PGRIDHY
DO I = 0, NINTS
  YTIC(2) = YTIC(1) - PTICL
  XX = PGRIDLX + (FLOAT(I) * DELTA_X)
  XTIC(1) = XX
  XTIC(2) = XX
  CALL GKS$POLYLINE (2, XTIC, YTIC)
  YTIC(2) = YTIC(1) - PTICS
  IF (I .LT. NINTS) THEN
    DO N = 1, MINTS-1
      XTIC(1) = XX + (FLOAT(N) * DX)
      XTIC(2) = XTIC(1)
      CALL GKS$POLYLINE (2, XTIC, YTIC)
    ENDDO
  ENDIF
ENDDO

C
C BOTTOM X AXIS
C
YTIC(1) = PGRIDLY
DO I = 0, NINTS
  YTIC(2) = YTIC(1) + PTICL
  XX = PGRIDLX + (FLOAT(I) * DELTA_X)
  XTIC(1) = XX
  XTIC(2) = XX
  CALL GKS$POLYLINE (2, XTIC, YTIC)
  YTIC(2) = YTIC(1) + PTICS
  IF (I .LT. NINTS) THEN
    DO N = 1, MINTS-1
      XTIC(1) = XX + (FLOAT(N) * DX)
      XTIC(2) = XTIC(1)
      CALL GKS$POLYLINE (2, XTIC, YTIC)
    ENDDO
  ENDIF
ENDDO

C
C LABEL X AXIS
C
CALL GKS$SET_TEXT_ALIGN (GKS$K_TEXT_HALIGN_CENTER,
2   GKS$K_TEXT_VALIGN_TOP)
CALL GKS$SET_TEXT_HEIGHT (PTXSIZE)
DO N = 1, PLOTDREC.NPASS
  IF (PLOTDREC.COLOR .EQ. GKS$K_COLOR) THEN
    CINDEX = N + 1
    CALL GKS$SET_TEXT_COLOR_INDEX (CINDEX)
  ENDIF

```

```

DO I = 0, NINTS
  XX = PGRIDLX + (FLOAT(I) * DELTA_X)
  YY = PGRIDLY - (FLOAT(N-1) * PTXSIZE * 0.8)
  IF (PLOTDREC.XUNITS .EQ. 1) THEN
    F = FLOAT((PLOTDREC.XPASS(1,N)) +
2      (I * PLOTDREC.IMAJT))
  ELSE
    F = FLOAT(PLOTDREC.START_ENE + ((N-1) * PLOTDREC.ERANG)
2      + (I * PLOTDREC.EMAJT))
  ENDIF
  WRITE (C_GRID, FMT='(F6.0)') F
  CALL GKS$TEXT (XX, YY, %DESCR(C_GRID(1:5)))
ENDDO
ENDDO

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C PRINT THE PEAK INFORMATION, CENTROID, ENERGY, AREA AND %ERROR
C
C
C SET THE CORDINATE VALUES FOR THE INFORMATION
  XX=1.0          !CENTROID BEGINING X VALUE
  CMY=7.75        !CENTROID Y VALUE

  EMY=7.55        !ENERGY Y VALUE

  AMY=7.35        !AREA Y VALUE

  PMY=7.15        !ERROR Y VALUE

  CALL GKS$SET_TEXT_ALIGN(GKS$K_TEXT_HALIGN_LEFT,
1      GKS$K_TEXT_VALIGN_BASE)
  CALL GKS$SET_TEXT_HEIGHT(PTXSIZE)
C
C*****
C
C
C PLOT THE DATA
C
C*****
C
C NDC VIEWPORT IS AREA OUTLINED BY GRID
C
  WSTYPE = PLOTDREC.WTYPE .AND. WSMASK
  XORIGIN = ((PGRIDLX-PLOWX+PMARGIN) / (PHIGHX-PLOWX+ (2*PMARGIN)))
2    *PLOTDREC.ASPECT_X
  XLENGTH = ((PGRIDHX-PLOWX+PMARGIN) / (PHIGHX-PLOWX+ (2*PMARGIN)))
2    *PLOTDREC.ASPECT_X

```

```

        YORIGIN = ((PGRIDLY-LOWY+PMARGIN)/(PHIGHY-LOWY+(2*PMARGIN)))
2      *PLOTDREC.ASPECT_Y
        YLENGTH = ((PGRIDHY-LOWY+PMARGIN)/(PHIGHY-LOWY+(2*PMARGIN)))
2      *PLOTDREC.ASPECT_Y
C
        IF (XORIGIN .LT. 0.0) XORIGIN = 0.0
        IF (XLENGTH .GT. PLOTDREC.ASPECT_X) XLENGTH = PLOTDREC.ASPECT_X
        IF (YORIGIN .LT. 0.0) YORIGIN = 0.0
        IF (YLENGTH .GT. PLOTDREC.ASPECT_Y) YLENGTH = PLOTDREC.ASPECT_Y
C
        CALL GKS$SET_VIEWPORT (NORMDATA, XORIGIN, XLENGTH, YORIGIN, YLENGTH)
        CALL GKS$SET_CLIPPING (GKS$K_CLIP)
C
C  ENABLE TRANSLATION FROM WORLD TO NDC
C
        CALL GKS$SELECT_XFORM (NORMDATA)
C
C*****
C
C  LOOP THROUGH ALL PLOT PASSES
C
        DO I = 1, PLOTDREC.NPASS
C
C
C  SET COLOR INDEX FOR EACH PASS
C
        IF (PLOTDREC.COLOR .EQ. GKS$K_COLOR) THEN
            CINDEX = I + 1
            CALL GKS$SET_PLINE_COLOR_INDEX (CINDEX)
            CALL GKS$SET_PMARK_COLOR_INDEX (CINDEX)
        ENDIF
C
C  WORLD COORDINATES WILL BE CHANNELS X DECADES
C  OR CHANNELS X COUNTS
C  ENERGY PLOTS MAY HAVE SLIGHTLY DIFFERENT CHANNELS EACH PASS
C  CALCULATE ACTUAL NUMBERS AND SET NORMALIZATION WINDOW
C
        XORIGIN = 0.0
        XLENGTH = FLOAT(PLOTDREC.XPASS(3,I))
        IF (PLOTDREC.CTSCALE .EQ. 1) THEN
            YORIGIN = FLOAT(PLOTDREC.IDEC)
            YLENGTH = YORIGIN + FLOAT(PLOTDREC.NDEC)
        ELSE
            YORIGIN = 0.0
            YLENGTH = 10 ** PLOTDREC.NDEC
        ENDIF
        CALL GKS$SET_WINDOW (NORMDATA, XORIGIN, XLENGTH,

```

```

      2      YORIGIN, YLENGTH)
C
C FOR POINT PLOTS A VERTICAL LINE SEGMENT WILL BE USED
C SO THAT THE SIZE OF THE POINT CAN BE CONTROLLED
C THE LINE WIDTH WILL BE 2 TIMES MINIMUM WIDTH AND
C THE LINE HEIGHT WILL BE APROXIMATELY 0.01 INCHES
C
      IF (PLOTREC.PTYPE .EQ. 1) THEN
        CALL GKS$SET_PLINE_LINEWIDTH (2.0)
        POINT_SIZE = ((YLENGTH - XORIGIN) / (PHIGHY-LOWY)) * 0.01
      ENDIF
C
C SET POLYMARKER ATTRIBUTE FOR ASTERISK PLOTS
C
      IF (PLOTREC.PTYPE .EQ. 2) THEN
        CALL GKS$SET_PMARK_TYPE (GKS$K_MARKERTYPE_ASTERISK)
        CALL GKS$SET_PMARK_SIZE (0.5)
      ENDIF
C
C GET STARTING AND ENDING CHANNEL TO BE PLOTTED
C
      SCHAN = PLOTREC.XPASS(1,I)
      IF (SCHAN .LT. 0) SCHAN = 0
      ECHAN = PLOTREC.XPASS(2,I)
      IF (ECHAN .GT. PLOTREC.NSPEC) ECHAN = PLOTREC.NSPEC
      NCHAN = ECHAN - SCHAN + 1
C
C CALCULATE CHANNEL VALUE ARRAY
C
      DO N = SCHAN, ECHAN
        XCHAN(N) = FLOAT(N - SCHAN)
      ENDDO
C
C PLOT THE DATA
C
      IF (PLOTREC.PTYPE .EQ. 1) THEN
        DO N = SCHAN, ECHAN
          X(1) = XCHAN(N)
          X(2) = X(1)
          Y(1) = PLOTREC.SDATA(N)
          Y(2) = Y(1) + POINT_SIZE
          CALL GKS$POLYLINE (2, X, Y)
        ENDDO
C
      ELSE IF (PLOTREC.PTYPE .EQ. 2) THEN
        CALL GKS$POLYMARKER (NCHAN, XCHAN(SCHAN),
          2      PLOTREC.SDATA(SCHAN))

```



```

C
    ELSE
        CALL GKSSPOLYLINE (NCHAN, XCHAN(SCHAN),
2          PLOTREC.SDATA(SCHAN))
    ENDIF
ENDDO

C
C NOW PLOT ALL THE FUNCTIONAL VALUES WITH IN THIS CHANNEL REGION
NPF=VHEAD(NP)
DY=YLENGTH-YORIGIN+1
CALL GKSSSET_TEXT_HEIGHT(DY/60.0)
CALL GKSSSET_TEXT_EXPFAC((1.0/DY)*240.0)
DO I=ISX, IEX
    DO J=1, PT(I).ENNF
        XX=PT(I).EVNC-1
        YY=PT(I).NPF(J)
        DXP=XX/YY
        XX=FLOAT(PT(I).EVSC-SCHAN)

C DETERMINE X CORRDINATE VALUES

        DO N=1, PT(I).NPF(J)
            XCHAN(N)=XX
            XX=XX + DXP
        ENDDO

C DETERMINE Y COORDINATE VALUES

        DO N=1, PT(I).NPF(J)
            IF(PT(I).F(J,N) .GT. 0.0) THEN
                PLOTREC.SDATA(N)=LOG10(PT(I).F(J,N))
            ELSE
                PLOTREC.SDATA(N)=-1.0
            ENDIF
        ENDDO

C PLOT THESE FUNCTIONAL VALUES

        NCHAN=PT(I).NPF(J)
        CALL GKSSPOLYLINE(NCHAN, XCHAN(1), PLOTREC.SDATA(1))

    ENDDO

C TAG THE LINES WITH THEIR ENERGY VALUES

    IF(PKD(I,1,AV) .LT. 0.0) THEN
        XMIN=1.0E+10
        DO J=1, PT(I).ENNF

```

```

        DO N=1,PT(I).NPF(J)
            IF (PT(I).F(J,N) .GT. 0.0) THEN
                FFV=LOG10 (PT(I).F(J,N))
                IF (FFV .LT. XMIN) XMIN=FFV
            ENDIF
        ENDDO
    ENDDO

    XCHAN(1)=PKD(I,1,XV)-SCHAN
    XCHAN(2)=XCHAN(1)
    PLOTDREC.SDATA(1)=XMIN-(YLENGTH-YORIGIN)/100.0
    PLOTDREC.SDATA(2)=PLOTDREC.SDATA(1)-
1      ((YLENGTH-YORIGIN)/10.5)

    CALL GKS$SET_PLINE_LINEWIDTH (1.0)

    CALL GKS$POLYLINE(2,XCHAN(1),PLOTDREC.SDATA(1))

    CALL GKS$SET_TEXT_ALIGN(GKS$K_TEXT_HALIGN_CENTER,
1      GKS$K_TEXT_VALIGN_TOP)
    CALL GKS$SET_TEXT_PATH(GKS$K_TEXT_PATH_DOWN)
    WRITE(OUTPUT,FMT='(I4)')NINT(PKD(I,3,ERA))
    IF (OUTPUT(1:1) .EQ. ' ') OUTPUT(1:4)=OUTPUT(2:4)
    IF (OUTPUT(1:1) .EQ. ' ') OUTPUT(1:4)=OUTPUT(2:3)
    CALL GKS$TEXT(XCHAN(1),PLOTDREC.SDATA(2),
1      OUTPUT(1:4))
    CALL GKS$SET_PLINE_LINEWIDTH (2.0)

ELSE
    XMAX=-1.0E+10
    DO J=1,PT(I).ENNF
        DO N=1,PT(I).NPF(J)
            IF (PT(I).F(J,N) .GT. 0.0) THEN
                FFV=LOG10 (PT(I).F(J,N))
                IF (FFV .GT. XMAX) XMAX=FFV
            ENDIF
        ENDDO
    ENDDO

    XCHAN(1)=PKD(I,1,XV)-SCHAN
    XCHAN(2)=XCHAN(1)
    PLOTDREC.SDATA(1)=XMAX+(YLENGTH-YORIGIN)/100.0
    PLOTDREC.SDATA(2)=PLOTDREC.SDATA(1)+
1      ((YLENGTH-YORIGIN)/10.5)

    CALL GKS$SET_PLINE_LINEWIDTH (1.0)

```

```

        CALL GKS$POLYLINE(2,XCHAN(1),PLOTREC.SDATA(1))

        CALL GKS$SET_TEXT_ALIGN(GKS$K_TEXT_HALIGN_CENTER,
1          GKS$K_TEXT_VALIGN_BASE)
        CALL GKS$SET_TEXT_PATH(GKS$K_TEXT_PATH_DOWN)
        WRITE(OUTPUT,FMT='(I4)')NINT(PKD(I,3,ERA))
        CALL GKS$TEXT(XCHAN(1),PLOTREC.SDATA(2),
1          OUTPUT(1:4))
        CALL GKS$SET_PLINE_LINEWIDTH (2.0)

        ENDIF

    ENDDO

C
C CLOSE UP THE GKS SYSTEM
C
    CALL GKS$DEACTIVATE_WS (WKID)
    CALL GKS$CLOSE_WS (WKID)
    CALL GKS$CLOSE_GKS ()

C QUE THE OUTPUT

1000    SNDITM(1).BUFLTH = 11
        SNDITM(1).ITMCDE = SJC$_FILE_SPECIFICATION
        SNDITM(1).BUFADR = %LOC(DEV_NAME)
        SNDITM(1).RTNLTH = 0

C
        SNDITM(2).BUFLTH = 7
        SNDITM(2).ITMCDE = SJC$_FILE_SETUP_MODULES
        SNDITM(2).BUFADR = %LOC(SETUP_NAME)
        SNDITM(2).RTNLTH = 0

CJPIC
CJPIC Determine whether this routine is being called from a batch process,
CJPIC and if it is, include a flag page.
CJPIC
CJPIC Fill in the item list to determine whether the current process is a
CJPIC batch process by asking for its mode.
CJPIC
CJPI jpiitm(1).buflth = 8
CJPI jpiitm(1).itmcode = jpi$_mode
CJPI jpiitm(1).bufadr = %loc(jpimod)
CJPI jpiitm(1).rtnlth = 0
CJPI jpiitm(2).buflth = 0
CJPI jpiitm(2).itmcode = 0
CJPIC
CJPIC Ask for the mode of the current process.
CJPIC

```

```

CJPI      call sys$getjpi(,,,%ref(jpiitm),,,)
CJPI      if(jpimod.eq.jpi$k_batch)then      !Batch job, so include the flag page.
CJPI      SNDITM(3).ITMCDE = SJC$_FILE_FLAG
CJPI      else      !Otherwise, do not include it.
CJPI      SNDITM(3).ITMCDE = SJC$_NO_FILE_FLAG
CJPI      endif
          SNDITM(3).BUFLTH = 0
          SNDITM(3).BUFADR = 0
          SNDITM(3).RTNLTH = 0

C
          SNDITM(4).BUFLTH = 0
          SNDITM(4).ITMCDE = SJC$_DELETE_FILE
          SNDITM(4).BUFADR = 0
          SNDITM(4).RTNLTH = 0

C
C DETERMINE THE QUE ON WHICH TO DIRECT THE OUTPUT
C THE QUE NAME WILL BE DEFINED IN THE LOGICAL XTP$QUE
  IL.ITEM=LNM$_STRING
  IL.BL=30
  IL.ADRSS=%LOC(LQUEUE)
  IL.RET=%LOC(LQUE_LN)
  SYS_STATUS=SYS$STRNLNM('LNM$PROCESS_TABLE','XTP$QUE',,IL)
  IF(.NOT. SYS_STATUS)CALL LIB$STOP(%VAL(SYS_STATUS))

          SNDITM(5).BUFLTH = LQUE_LN
          SNDITM(5).ITMCDE = SJC$_QUEUE
          SNDITM(5).BUFADR = %LOC(LQUEUE)
          SNDITM(5).RTNLTH = 0

C
          SNDITM(6).BUFLTH = 8
          SNDITM(6).ITMCDE = SJC$_AFTER_TIME
          SNDITM(6).BUFADR = %LOC(DTIME)
          SNDITM(6).RTNLTH = 0

C
          SNDITM(7).BUFLTH = 0
          SNDITM(7).ITMCDE = 0

C
  SYS_STATUS = SYS$SNDJBCW
2    (,%VAL(SJC$_ENTER_FILE),%VAL(0),SNDITM,IOSB,,)
  IF (SYS_STATUS.NE. SS$_NORMAL) THEN
    CALL LIB$SIGNAL(%VAL(SYS_STATUS))
  ENDIF
  IF (IOSB(1).NE. JBC$_NORMAL) THEN
    CALL LIB$SIGNAL(%VAL(IOSB(1)))
  ENDIF

C
END

```

C.15 DCL Command File for Linking BATCH_GIDXC_ANALYSIS

```
$ LINK/EXE=INEL$EXE:BATCH_gidxc_ANALYSIS BATCH_gidxc_ANALYSIS,CBAT_PRGIDXC,-  
    FIND_LINE,MU_LOOK,NOR_SEARCH,FIND_BKG_FILES,READ_ATALE,-  
    MU_self_LOOK,lstsqr,sigfig,br_uc,wavg_o,-  
    DPY_CBAT_GAUSS,DPY_CBAT_LINFIT,DPY_CBAT_LINFIT_DP,-  
    DPY_REGION,-  
GAP$SRC:CBATGAUSS_OUT,FIT_BKG,-  
    BGET,BKG_CHANNEL,C_ENERGY,SUB_BKG,-  
    BATCH_CAL_PULSER,-  
    OPEN$SP,TRUE_POSITIVE,GAUSS,LIBRARY_TAG,-  
    FGET,-  
    WEIGHT,RANSUM,EFF,LIB_CHECK,LIBR_LOOKUP,INTCOR,-  
    CHECK_ICODE,RSOLVE,-  
    PR_NC_SUMMARY,PR_EXIT,-  
    MCURIES,OTHERTAG,SAVE_LOOKUP,-  
GAP$LIB/LIB,VT100$LIB/LIB  
$EXIT
```

C.16 BATCH_GIDXC_SUMMARY Main Routine

program batch_gidxc_summary

```
C  
C Routine to generate a summary report for a SWEPP barrel analysis (using up to  
C six detectors). Output based on that found in BATCH_GIDXC_ANALYSIS, but with  
C isotopic ratios preceding radionuclide activities and no Pu-239 mass assay  
C values. Averages and standard deviations, with outlier testing, are obtained  
C using the subroutine WAVG_O.  
C  
C Written by D. A. Femec, January 27, 1994.  
C  
C Modified to not use the linear-methods flag and to replace the below-  
C detection-limits flag with a not-true-positive flag. Added the program  
C version number (starting at 01.00) and the date the program was run to the  
C banner. Expanded the summary output to include a grand weighted average of  
C all mass ratios over all detectors without outlier testing. Also now  
C generate a file in the directory pointed to by the logical PAN$RATIOS with a  
C name of the form <bar code id>.GAM containing mass ratios and fractions as  
C per L. V. East's draft EDF of January 26, 1994. D. A. Femec, June 20, 1994.  
C  
C  
C This program, modified as per the above, constitutes version 01.00 of the
```

```

C SWEPP version of BATCH_GIDXC_SUMMARY. D. A. Femec, July 28, 1994.
C
C
C Declaration statements.
C
    implicit none      !For safety's sake.
    character*80 chline,chlin2,blanks
    character*24 sigfig  !Function for displaying significant figures.
    character*16 spcusd  !For displaying which detectors were used.
    character*9 curdat   !Date program was run.
    character*8 contrn/'UNKNWNID'/  !The container's id.
    character*1 spaces(80)/80*' '/
    real*4 values(6),errors(6),  !Data points passed to WAVG_O.
    . mean,stdev,wmean,wstdev,rchisq,  !Returned from WAVG_O.
    . rtimes(6),  !Real times from the spectra used.
    . pu239a,  !The Pu-239 activity for the current spectrum.
    . mrati(5),emrati(5),
    . rP238P,eP238P,rP240P,eP240P,rP241P,eP241P,rP242P,eP242P,
    . rA241P,eA241P,rU235P,eU235P,rA241U,eA241U,  !Ratios and uncertainties.
    . mfrac(5),emfrac(5),  !Mass fractions and uncertainties.
    . fPu239,ePu239,fPu238,ePu238,fPu240,ePu240,fPu241,ePu241,fPu242,
    . ePu242,
    . dmfrac(5)/0.00012,0.93977,0.05821,0.00166,0.00024/  !Default mass fractions.
    integer*4 i,j,k,l,  !Common "DO" loop variables.
    . nvalus,ndatum,  !Number of data points passed to WAVG_O.
    . jlast,indexs(6),jindexs(6),  !Indice records.
    . numchr,nfiles  !Character and file counters.
    logical ifvald(6),  !Flag whether a value was used by WAVG_O.
    . ifirst,  !Flag whether a heading should be printed.
    . doout1/.true./,  !Flag to perform outlier testing in WAVG_O.
    . noout1/.false./  !Flag to not perform outlier testing in WAVG_O.
    equivalence (blanks(1:1),spaces(1)),
    . (mfrac(1),fPu238),(emfrac(1),ePu238),
    . (mfrac(2),fPu239),(emfrac(2),ePu239),
    . (mfrac(3),fPu240),(emfrac(3),ePu240),
    . (mfrac(4),fPu241),(emfrac(4),ePu241),
    . (mfrac(5),fPu242),(emfrac(5),ePu242),
    . (mrati(1),rP238P),(emrati(1),eP238P),
    . (mrati(3),rP240P),(emrati(3),eP240P),
    . (mrati(4),rP241P),(emrati(4),eP241P),
    . (mrati(5),rP242P),(emrati(5),eP242P)
C
C INCLUDE ANALYSIS COMMONS
    INCLUDE '($LNMDEF)'
    INCLUDE 'GAP$SRC:PARAM_BLOCK.INC'
C
C STRUCTURE USE TO SAVE THE RESULTS OF A SPECTRUM ANALYSIS

```

```

C          EWK 5/16/90
STRUCTURE  /TWR_RESULTS/
          INTEGER*4  NPS    !NUMBER OF PEAKS ANALYZED

C
C RAW PEAK DATA
          REAL*4
                                PEAK(200,4) !PEAK ANALYSIS RESULTS
                                !PEAK(I,1)=ENERGY
                                !PEAK(I,2)=COUNTS/SEC IN PEAK
                                !PEAK(I,3)=% ERROR
                                !PEAK(I,4)=EFFICIENCY @ ENERGY

C NUCLIDE DATA
          INTEGER*4  NOE    !NUMBER OF NUCLIDES IDENTIFIED
          INTEGER*4  Z(50)  !NUCLIDE Z
          INTEGER*4  A(50)  !NUCLIDE A
          CHARACTER*4 ID(50) !NUCLIDE ALPHA NUMERIC ID
          REAL*4     MC(50)  !NET NUCLIDE MICRO-CURIES
          REAL*4     ERR(50) !ERROR IN MICRO-CURIES
END STRUCTURE
RECORD      /TWR_RESULTS/TR

C
C Structure for recording isotopic ratios from an individual detector.
C
structure /isorat/
  integer*4 nratio    !Number of ratios calculated (maximum = 6).
  integer*4 rat_al(100) !For the 1st isotope of the ratio: nuclide A
  character*4 rat_il(100) ! and ID.
  integer*4 rat_a2(100) !For the 2nd isotope of the ratio: nuclide A
  character*4 rat_i2(100) ! and ID.
  real*4 ratios(100)  !Ratios (as percentages)
  real*4 uncrat(100)  ! and uncertainties in the ratios (also as percentages).

C
C Ratio source and fitting warning flag: first character of S=standard,
C U=excessive uranium, L=lead liner present; second character=which line
C was fit using linear methods; third character=which line was below
C detection limits.
C
  character*4 ratflg(100)
end structure
record /isorat/ ir

C
C Structure for recording isotopic ratios and radionuclide activities from
C all detectors.
C
structure /ratact/
  integer*4 nratio(100) !Number of ratios calculated (maximum = 6).
  integer*4 rat_al(100,6) !For the 1st isotope of the ratio: nuclide A
  character*4 rat_il(100,6) ! and ID.

```

```

integer*4 rat_a2(100,6)  !For the 2nd isotope of the ratio:  nuclide A
character*4 rat_i2(100,6)  ! and ID.
real*4 ratios(100,6)  !Ratios (as percentages)
real*4 uncrat(100,6)  ! and uncertainties in the ratios (also as percentages).

C
C Ratio source and fitting warning flag:  first character of S=standard,
C U=excessive uranium, L=lead liner present; second character=which line
C was fit using linear methods; third character=which line was below
C detection limits; fourth character=which line is invalid.
C
character*4 ratflg(100,6)
C NUCLIDE DATA
      INTEGER*4  NOE(6)          !NUMBER OF NUCLIDES IDENTIFIED
      INTEGER*4  Z(50,6)        !NUCLIDE Z
      INTEGER*4  A(50,6)        !NUCLIDE A
      CHARACTER*4 ID(50,6)      !NUCLIDE ALPHA NUMERIC ID
      REAL*4     MC(50,6)        !NET NUCLIDE MICRO-CURIES
      REAL*4     ERR(50,6)      !ERROR IN MICRO-CURIES (convert to absolute).
      end structure
record /ratact/ ra

      INTEGER*4  SDATE,STIME,CDATE,CTIME
      CHARACTER*4 STM,CTM

      INTEGER*4  STATUS
      CHARACTER*512  HEADER

C
C STRUCTURE USED TO RUN SYSTEM CALL $TRNLNM
      STRUCTURE /ITEM_LIST/
      INTEGER*2  BL
      INTEGER*2  ITEM
      INTEGER*4  ADRESS
      INTEGER*4  RET
      INTEGER*4  END    /0/
      END STRUCTURE
      RECORD /ITEM_LIST/IL

C
      CHARACTER*30  SP_NAME
      INTEGER*4  SP_NAME_LN,SYS$TRNLNM
      CHARACTER*80  SP_FILE

C FOLLOWING IS A LIST OF THE NUCLIDES FOR WHICH THE DOUBLE MATRIX CORRECTION IS
C GOING TO BE PERFORMED.  The first radionuclide in each list is the one on
C which the double matrix correction is based.  This list also defines the TRU
C radionuclides for which activities should be printed.
C
      CHARACTER*4 dxc_ID(8,3) /'PU ','PU ','PU ','AM ',

```



```

1      'U      'U      'U      '
2      'EU     '      '      '
3      '      '      '      '
4      '      '      '      '
5      '      '      '      '
INTEGER*4  dxc_A(8,3)  /   239,   240,   241,   241,
1      235,   233,   238,   0,
2      152,   0,   0,   0,
3      0,   0,   0,   0,
4      0,   0,   0,   0,
5      0,   0,   0,   0/
INTEGER*4  dxc_Z(8,3)  /   94,   94,   94,   95,
1      92,   92,   92,   0,
2      63,   0,   0,   0,
3      0,   0,   0,   0,
4      0,   0,   0,   0,
5      0,   0,   0,   0/

```

C

C Format statements.

C

```

2350 format(//10X,'GRAND WEIGHTED AVERAGE ISOTOPIC MASS RATIOS ',
. ' (%) FOR ALL DETECTORS'//)
2355 format(10x,'NONE CAN BE CALCULATED - A LEAD LINER IS LIKELY ',
. 'PRESENT')
2360 format(2x,a2,'-',i3,' to ',a2,'-',i3,': ',a24)
2420 format('; Mass Fractions and Ratios for Container ',a<numchr>/
. '; Generated ',a9,' by BATCH_GIDXC_SUMMARY V. 01.00'/'/' !Version number.
. 'PU_MASS_FRACTIONS'/5(4x,i3,2(4x,1pe10.4)), 'END'/
. 'PU_MASS_RATIOS'/4(4x,i3,2(4x,1pe10.4)), 'END'/
. 'AM_PU_MASS_RATIOS'/4x,i3,2(4x,1pe10.4)/'END'/
. 'U_PU_MASS_RATIOS'/4x,i3,2(4x,1pe10.4)/'END'/
. 'AM_U_MASS_RATIOS'/4x,i3,2(4x,1pe10.4)/'END')
8000 FORMAT('1',///,' ',13('*')),
1 'SWEPP BARREL MONITOR GAMMA-RAY SPECTROMETERY SUMMARY REPORT ',
2 'V. 01.00 - FINAL REPORT - GENERATED ',a9,13('*'))
8010 FORMAT(///,' SAMPLE NAME: ',A100)
8020 FORMAT(//,' SAMPLE ID(S): ',a<numchr>,<76-numchr>(' '),
1 'OPERATOR: ',30('_'))
8030 FORMAT(//,' SAMPLE COLLECTION (DATE/TIME): ',1(' '),
1 I6,'/',A4,48(' '), 'APPROVED BY: ',27('_'))
8040 FORMAT(//,' SAMPLE COUNTED (DATE/TIME): ',4(' '),I6,
1 '/',A4,48(' '), 'COMMENTS: ',30('_'))
8050 FORMAT(//,' SAMPLE QUANTITY:',1PE12.5,' QUANTITY UNITS: ',
1 A<numchr>,<54-numchr>(' '),30('_'),//,' REAL COUNT TIME(S): ',
2 a<nfiles*12>,<80-nfiles*12>(' '),30('_'),//,' ATTENUATION ',
3 'TABLE USED: ',a<nfiles*12>,<76-nfiles*12>(' '),30('_'))
8100 FORMAT(//10x,'ACTIVITIES OBSERVED BY DETECTOR #',i2,

```

```

1 ' (AND POSSIBLY OTHERS): TRU RADIONUCLIDES AND OTHERS ',
1 '> 5% OF PU-239',/, ' RADIONUCLIDE',4(' '),
1 'SAMPLE ACTIVITY (' ,A3, '/',A<numchr>,' )',6(' '),
2 '% ERROR (1-SIGMA)',6x, 'OBSERVED BY DETECTOR(S)'/)
8105 FORMAT(/,
2 ' ***** NO PHOTOPEAKS TAGGED IN THE ANALYSIS FOR DETECTOR #',
3 i2, '*****')
8107 format(9x, '(A LEAD LINER IS LIKELY PRESENT)')
8110 FORMAT(I5,A4,I3,10X,1PE11.3,12X,0PF8.2,16x,a16)
9200 format(/,10X, 'ISOTOPIC MASS RATIOS (%) OBSERVED BY DETECTOR #',
. i2, ' (AND POSSIBLY OTHERS)'/
. 10x, 'NONE CAN BE CALCULATED - A LEAD LINER IS LIKELY PRESENT')
9202 format(/,10X, 'ISOTOPIC MASS RATIOS (%) OBSERVED BY DETECTOR #',
. i2, ' (AND POSSIBLY OTHERS)'/
. 10x, '(ONLY A RESTRICTED SET CAN BE CALCULATED - EXCESSIVE ',
. 'URANIUM IS LIKELY PRESENT)'/)
9204 format(/,10X, 'ISOTOPIC MASS RATIOS (%) OBSERVED BY DETECTOR #',
. i2, ' (AND POSSIBLY OTHERS)'/
. 10x, '(THE STANDARD SET CAN BE CALCULATED)'/)
9206 format(/,10X, 'ISOTOPIC MASS RATIOS (%) OBSERVED BY DETECTOR #',
. i2, ' (AND POSSIBLY OTHERS)'/
. 10x, '(ONLY A RESTRICTED SET CAN BE CALCULATED - EXCESSIVE ',
. 'AMERICIUM IS LIKELY PRESENT)'/)
9208 format(/,10X, 'ISOTOPIC MASS RATIOS (%) OBSERVED BY DETECTOR #',
. i2, ' (AND POSSIBLY OTHERS)'/
. 10x, '(THE IS A TEST SET)'/)
9210 format(2x,a2,'-',i3, ' to ',a2,'-',i3,': ',a24, ' (OBSERVED BY ',
. 'DETECTOR(S) ',a16,')')
9215 format(5x, '*** WARNING: THE ABOVE RATIO MAY BE IN ERROR ',
. 'BECAUSE ',a46)
9220 format(5x, 'BECAUSE ',a59)
C
C Beginning of the routine.
C
C Get the current date for use in generating output.
C
call date(curdat)
C
C Attempt to translate all six A<n>$$SPECTRUM$ID logicals and open the
C corresponding results files.
C
IL.ADRESS=%LOC(SP_NAME)
IL.ITEM=LNMS_STRING
IL.BL=30
IL.RET=%LOC(SP_NAME_LN)
chline=' ' !Reset CHLINE, CHLIN2, and NUMCHR.
chlin2=' '

```

```

numchr=0
do i=1,6
STATUS=SYS$STRNLNM(, 'LNM$PROCESS_TABLE', ('A'//char(i+48)//
1 '$SPECTRUM$ID'),, IL)
IF(.NOT. STATUS)goto 50      !Assume that all have been read.
C
C OPEN THE RESULTS FILES. THE NAME OF THE RESULTS FILES WILL BE THE SAME AS
C THE SPECTRUM FILES EXCEPT FOR THE EXTENTION WHICH WILL BE .RES
C
SP_FILE=' '
SP_FILE='BATCH$RESULTS:'//SP_NAME//'.RES'
OPEN(UNIT=1,NAME=SP_FILE,TYPE='OLD',FORM='UNFORMATTED')
READ(1)HEADER      !SAVE THE SPECTRUM HEADER
READ(1)P           !SAVE SPECTRUM PARAMETER BLOCK
C READ INFO FROM FILE CLOSE IT
READ(1)TR
read(1)ir          !Isotopic ratios.
CLOSE(UNIT=1)
chline((numchr+1):(numchr+11))=sp_name(1:11)
chlin2((numchr+1):(numchr+8))=p.sys_volume_units
numchr=numchr+12    !Add a space between each spectrum id.
C
C Transfer the results into the extended structure RA.
C
ra.nratio(i)=ir.nratio
do j=1,ir.nratio
ra.rat_a1(j,i)=ir.rat_a1(j)
ra.rat_i1(j,i)=ir.rat_i1(j)
ra.rat_a2(j,i)=ir.rat_a2(j)
ra.rat_i2(j,i)=ir.rat_i2(j)
ra.ratios(j,i)=ir.ratios(j)
ra.uncrat(j,i)=ir.uncrat(j)
ra.ratflg(j,i)=ir.ratflg(j)
enddo
ra.noe(i)=tr.noe
do j=1,tr.noe
ra.z(j,i)=tr.z(j)
ra.a(j,i)=tr.a(j)
ra.id(j,i)=tr.id(j)
ra.mc(j,i)=tr.mc(j)
ra.err(j,i)=abs(tr.err(j)*tr.mc(j)/100.0)    !Convert to absolute error.
enddo
nfiles=i
rtimes(i)=p.rtime/100.0    !Convert from centiseconds to seconds.
enddo
C
C GET DATA FROM HEADER BLOCK

```

```

C
50  SDATE=p.collect_date      !SAMPLE DATE
    STIME=p.collect_time      !SAMPLE TIME
    WRITE(UNIT=STM(1:4),FMT='(I4)')STIME
C  ELIMINATE LEADING SPACES
    IF(STM(1:3).EQ.' ')STM(1:3)='000'
    IF(STM(1:2).EQ.' ')STM(1:2)='00'
    IF(STM(1:1).EQ.' ')STM(1:1)='0'
    CDATE=p.start_date        !COUNT DATE
    CTIME=p.start_time        !COUNT TIME
    WRITE(UNIT=CTM(1:4),FMT='(I4)')CTIME
C  ELIMINATE LEADING SPACES
    IF(CTM(1:3).EQ.' ')CTM(1:3)='000'
    IF(CTM(1:2).EQ.' ')CTM(1:2)='00'
    IF(CTM(1:1).EQ.' ')CTM(1:1)='0'
C
C  OPEN FILE TO CONTAIN THE PRINTED OUTPUT
    CALL SPOOL
C
C  BEGIN PRINTING OF SUMMARY SHEETS
C
    WRITE(6,8000)curdat
    WRITE(6,8010)HEADER(61:160)
    WRITE(6,8020)chline(1:numchr)
    WRITE(6,8030)SDATE,STM
    WRITE(6,8040)CDATE,CTM
    chline=' ' !Reset CHLINE.
    do i=1,nfiles
        write(chline(((i-1)*10+1):((i-1)*10+10)),fmt='(f10.2)')
        rtimes(i)
    enddo
    numchr=INDEX(P.VOLUME_UNITS,' ')-1
    WRITE(6,8050)p.volume,P.VOLUME_UNITS(1:numchr),
    1 chline(1:(nfiles*12)),chlin2(1:(nfiles*12))
C
C  Print (and save if needed for the PAN$RATIOS file) the grand weighted
C  average isotopic ratio results, without outlier testing and ignoring which
C  set of ratios was calculated.
C
    write(6,2350)
C
C  Reset all of the PAN$RATIOS file ratios to zero; each of these that

```

```

C      ends with P is the ratio of the specified radionuclide to Pu-239; the
C      last one is the Am-241 : U-235 ratio.
C

```

```

rP238P=0.0    !Ratios.
eP238P=0.0    !Absolute errors in the ratios.
rP240P=0.0
eP240P=0.0
rP241P=0.0
eP241P=0.0
rP242P=0.0
eP242P=0.0
rA241P=0.0
eA241P=0.0
rU235P=0.0
eU235P=0.0
rA241U=0.0
eA241U=0.0

```

```

C
C      Find the first file that can be used (some may have no ratios because
C      of the presence of a lead liner).
C

```

```

do k=1,nfiles
  if(ra.ratflg(1,i)(1:1).ne.'L')then    !No lead liner present.
    i=k    !Use this as the first file.
    goto 240
  endif
enddo
write(6,2355)
goto 300    !None to print.

```

```

C
C      Calculate the grand weighted averages.
C

```

```

240 do 244 j=1,ra.nratio(i)
  ndatum=0
  if((ra.rat_a1(j,i).ne.0).and.(ra.ratflg(j,i)(3:4).eq.'  '))then
    ndatum=1
    values(1)=ra.ratios(j,i)
    errors(1)=ra.uncrat(j,i)
  endif
  do 242 k=1,nfiles    !Look for matches in subsequent files.
    if(k.eq.i)goto 242    !This file is already treated above.
    do l=1,ra.nratio(k)
      if((ra.rat_a1(l,k).eq.ra.rat_a1(j,i)).and.
        . (ra.rat_a2(l,k).eq.ra.rat_a2(j,i)).and.
        . (ra.rat_i1(l,k).eq.ra.rat_i1(j,i)).and.
        . (ra.rat_i2(l,k).eq.ra.rat_i2(j,i)).and.
        . (ra.rat_a1(l,k).ne.0).and.

```

```

      (ra.ratflg(1,k)(3:4).eq.' '))then    !Match found.
        ndatum=ndatum+1
        values(ndatum)=ra.ratios(1,k)
        errors(ndatum)=ra.uncrat(1,k)
      endif
    enddo
242   enddo
      if(ndatum.eq.0)goto 244    !No values to average.
      call wavg_o(values,errors,ndatum,noout1,mean,stdev,wmean,
        wstdev,rchisq,ifvald)
      write(6,2360)ra.rat_i1(j,i)(1:2),ra.rat_a1(j,i),
      ra.rat_i2(j,i)(1:2),ra.rat_a2(j,i),sigfig(wmean,wstdev)

C
C
C
      Is this a ratio to be saved for the PAN$RATIOS file?

      if((ra.rat_i1(j,i)(1:2).eq.'PU').and.(ra.rat_a1(j,i).eq.238)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rP238P=wmean
        eP238P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'PU').and.(ra.rat_a1(j,i).eq.240)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rP240P=wmean
        eP240P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'PU').and.(ra.rat_a1(j,i).eq.241)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rP241P=wmean
        eP241P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'PU').and.(ra.rat_a1(j,i).eq.242)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rP242P=wmean
        eP242P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'AM').and.(ra.rat_a1(j,i).eq.241)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rA241P=wmean
        eA241P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'U ').and.(ra.rat_a1(j,i).eq.235)
      .and.

```

```

      (ra.rat_i2(j,i)(1:2).eq.'PU').and.(ra.rat_a2(j,i).eq.239))then
        rU235P=wmean
        eU235P=wstdev
      endif
      if((ra.rat_i1(j,i)(1:2).eq.'AM').and.(ra.rat_a1(j,i).eq.241)
      .and.
      (ra.rat_i2(j,i)(1:2).eq.'U ').and.(ra.rat_a2(j,i).eq.235))then
        rA241U=wmean
        eA241U=wstdev
      endif
244   enddo
C
C   Calculate the Pu-239 mass fractions and write the appropriate records to
C   the PAN$RATIOS file.  If any of the calculated mass fractions are zero,
C   then use the default (as per L. V. East's draft EDF of January 26, 1994)
C   and scale proportionally the non-default mass fractions and uncertainties
C   (and if all are zero, skip this correction; note that if all radionuclides
C   for which mass ratios can be calculated have default mass fractions,
C   Pu-239 will automatically).
C
      fPu239=1.0/(rP238P+rP240P+rP241P+rP242P+1.0)
      ePu239=fPu239**2*sqrt(eP238P**2+eP240P**2+eP241P**2+eP242P**2)
      fPu238=rP238P*fPu239
      ePu238=sqrt((fPu239*eP238P)**2+(rP238P*ePu239)**2)
      fPu240=rP240P*fPu239
      ePu240=sqrt((fPu239*eP240P)**2+(rP240P*ePu239)**2)
      fPu241=rP241P*fPu239
      ePu241=sqrt((fPu239*eP241P)**2+(rP241P*ePu239)**2)
      fPu242=rP242P*fPu239
      ePu242=sqrt((fPu239*eP242P)**2+(rP242P*ePu239)**2)
      if((mfrac(1).ne.0.0).or.(mfrac(2).ne.0.0).or.(mfrac(3).ne.0.0).or.
      . (mfrac(4).ne.0.0).or.(mfrac(5).ne.0.0))then
        do i=1,5
          if(mfrac(i).eq.0.0)then      !Use the default and scale the others.
            mfrac(i)=dmfrac(i)
            fPu239=fPu239/(1.0+mfrac(i))      !Correct the Pu-239 mass fraction and uncertainty.
            ePu239=ePu239/((1.0+mfrac(i))**2)
          do j=1,5
            if((mfrac(j).ne.fPu239).and.      !Correct the non-Pu-239, . . .
            . (emfrac(j).ne.0))then      ! . . . non-default mass fractions.
              mfrac(j)=mfrac(j)/(1.0+mfrac(i))
              emfrac(j)=sqrt((fPu239*emrati(j))**2+
              . (mrati(j)*ePu239)**2)
            endif
          enddo
        enddo
      endif
    enddo
  enddo
enddo

```

endif

Find the barrel's barcode id - if the header was prepared by GC, the barcode id should follow the word "Barrel" and precede a colon. If this combination is found, use the DOS limit of at most the first eight non-blank characters of the id; if this combination is not found, use the default file name prefix of UNKNWNID.

```
i=index(header,'Barrel ')
contr='UNKNWNID' !Assume the default will be used.
numchr=8
if(i.ne.0)then !Now see if a colon following 'Barrel ' is there.
  i=i+7 !Move to the character following 'Barrel '.
  j=index(header(i:512),':')
  if(j.ne.0)then !Now extract the barcode id.
    j=index(header(i:512),' ') !See if there is a blank before the colon.
    if((j.eq.0).or.(j.gt.8))j=8 !Use the first eight characters following 'Barrel '.
    contr=header(i:(i+j-1))
    numchr=j
  endif
endif
```

Now open the file, write the data to it, then close the file.

```
open(7,file=('pan$ratios: '//contr(1:numchr) '//'.gam'),
. carriagecontrol='list',status='new')
write(7,2420)contr(1:numchr),curdat,
. ((237+i),mfrac(i),emfrac(i),i=1,5),(238),rP238P,eP238P,
. (240),rP240P,eP240P,(241),rP241P,eP241P,(242),rP242P,eP242P,
. (241),rA241P,eA241P,(235),rU235P,eU235P,(241),rA241U,eA241U
close(7,status='keep')
```

Print the isotopic ratio results, grouping them by which ones are observed by which detector. (Used, i.e. already displayed, ratios will have the first radionuclide's A value set to zero.)

```
do 290 i=1,nfiles
  ifirst=.true. !No heading yet printed for this detector.
  if(ra.ratflg(1,i)(1:1).eq.'L')then !Lead liner present.
    write(6,9200)i
    goto 290 !None to print.
  endif
  do j=1,ra.nratio(i)
    if(ra.rat_al(j,i).ne.0)then !Use only those that have not been used before.
      ndatum=1
      values(1)=ra.ratios(j,i)
      errors(1)=ra.uncrat(j,i)
```



```

indexs(1)=i
jindexs(1)=j
if(ra.ratflg(j,i)(2:4).ne.' ')goto 255 !Process this by itself.
do 250 k=(i+1),nfiles !Look for matches in subsequent files.
  do l=1,ra.nratio(k)
    if((ra.rat_a1(l,k).ne.0).and. !Use only those that have not been used before.
      ((ra.rat_a1(l,k).eq.ra.rat_a1(j,i)).and.
      (ra.rat_a2(l,k).eq.ra.rat_a2(j,i)).and.
      (ra.rat_i1(l,k).eq.ra.rat_i1(j,i)).and.
      (ra.rat_i2(l,k).eq.ra.rat_i2(j,i)).and.
      (ra.ratflg(l,k).eq.ra.ratflg(j,i))))then !Match found.
      ndatum=ndatum+1
      values(ndatum)=ra.ratios(l,k)
      errors(ndatum)=ra.uncrat(l,k)
      indexs(ndatum)=k
      jindexs(ndatum)=l
    endif
  enddo
250 enddo
255 call wavg_o(values,errors,ndatum,dooutl,mean,stdev,wmean,
  wstdev,rchisq,ifvald)
do k=1,ndatum !Is the primary (i.e., i) detector's value valid?
  if((indexs(k).eq.i).and.(.not.ifvald(k)))then !Resubmit just the invalid ones.
    nvalus=0 !Track "new" NDATUM.
    do l=1,ndatum
      if(.not.ifvald(l))then !Use this one.
        nvalus=nvalus+1
        values(nvalus)=values(l)
        errors(nvalus)=errors(l)
        indexs(nvalus)=indexs(l)
        jindexs(nvalus)=jindexs(l)
      endif
    enddo
    ndatum=nvalus
    goto 255 !Try WAVG_O again.
  endif
enddo
if(.not.ifirst) !Has the ratio set flag changed?
  ifirst=ra.ratflg(j,i)(1:1).ne.ra.ratflg(jlast,i)(1:1)
if(ifirst)then !Print the heading.
  if(ra.ratflg(j,i)(1:1).eq.'U')write(6,9202)i !Excessive-uranium set.
  if(ra.ratflg(j,i)(1:1).eq.'S')write(6,9204)i !Standard set
  if(ra.ratflg(j,i)(1:1).eq.'A')write(6,9206)i !Excessive-ameridium set.
  if(ra.ratflg(j,i)(1:1).eq.'T')write(6,9208)i !Test set.
  ifirst=.false. !Do not print the heading again for this detector.
endif
spcusc=blanks(1:16) !Construct the list of detectors used.

```

```

numchr=0
do k=1,ndatum
  if(ifvald(k))then
    if(numchr.eq.0)then
      spcusc(1:1)=char(indexs(k)+48)
      numchr=1
    else
      spcusc((numchr+1):(numchr+3))=', '//'
      char(indexs(k)+48)
      numchr=numchr+3
    endif
  endif
enddo
if(ra.ratflg(j,i)(3:4).eq.' ')then !Valid ratio, so print it.
  write(6,9210)ra.rat_il(j,i)(1:2),ra.rat_al(j,i),
  ra.rat_i2(j,i)(1:2),ra.rat_a2(j,i),
  sigfig(wmean,wstdev),spcusc
  if(ra.ratflg(j,i)(2:2).eq.'1')write(6,9215)
  'THE FIRST LINE USED WAS FIT BY LINEAR METHODS '
  if(ra.ratflg(j,i)(2:2).eq.'2')write(6,9215)
  'THE SECOND LINE USED WAS FIT BY LINEAR METHODS'
  if(ra.ratflg(j,i)(2:2).eq.'B')write(6,9215)
  'BOTH LINES USED WERE FIT BY LINEAR METHODS '
else !Invalid ratio, so say why.
  write(6,9210)ra.rat_il(j,i)(1:2),ra.rat_al(j,i),
  ra.rat_i2(j,i)(1:2),ra.rat_a2(j,i),
  'RATIO NOT DETERMINABLE ',spcusc
  if(ra.ratflg(j,i)(3:3).eq.'1')write(6,9220)
  ('THE FIRST LINE USED HAS A NON-TRUE-POSITIVE '//
  'ACTIVITY')
  if(ra.ratflg(j,i)(3:3).eq.'2')write(6,9220)
  ('THE SECOND LINE USED HAS A NON-TRUE-POSITIVE '//
  'ACTIVITY')
  if(ra.ratflg(j,i)(3:3).eq.'B')write(6,9220)
  ('BOTH LINES USED HAVE NON-TRUE-POSITIVE '//
  'ACTIVITIES')
  if(ra.ratflg(j,i)(4:4).eq.'1')write(6,9220)
  ('THE FIRST LINE USED IS INVALID '//blanks(1:28))
  if(ra.ratflg(j,i)(4:4).eq.'2')write(6,9220)
  ('THE SECOND LINE USED IS INVALID '//blanks(1:28))
  if(ra.ratflg(j,i)(4:4).eq.'B')write(6,9220)
  ('BOTH LINES USED ARE INVALID '//blanks(1:28))
endif
do k=1,ndatum !Mark as used those that were valid.
  if(ifvald(k))ra.rat_al(jndexs(k),indexs(k))=0
enddo
jlast=j

```

```

        endif
    enddo
290    enddo
C
C    Print the radionuclide activities, grouping them by which ones are
C    observed by which detectors. (Used, i.e. already displayed, activities
C    will have the radionuclide's Z value set to zero.)
C
300 do 390 i=1,nfiles
    ifirst=.true.    !No heading yet printed for this detector.
    IF(ra.NOE(i) .LE. 0)THEN
        WRITE(6,8105)i
        if(ra.ratflg(1,i)(1:1).eq.'L')write(6,8107)    !Lead liner present.
        GOTO 390
    ENDIF
    pu239a=0.0    !If no Pu-239 if found, print all others.
    do j=1,ra.noe(i)    !Find the Pu-239 activity for later use.
        if((ra.a(j,i).eq.239).and.(ra.id(j,i).eq.'PU '))then    !Found, so save it.
            pu239a=ra.mc(j,i)
            endif
        enddo
    do 380 j=1,ra.noe(i)
        if(ra.z(j,i).ne.0)then    !Use only those that have not been used before.
            do l=1,8
                if((ra.a(j,i).eq.dxc_a(1,1)).and.
                    (ra.id(j,i).eq.dxc_id(1,1)))goto 310    !TRU, so use it.
            enddo
            if(ra.mc(j,i).lt.(0.05*pu239a))goto 380    !Other, and too little of it.
310        ndatum=1
        values(1)=ra.mc(j,i)
        errors(1)=ra.err(j,i)
        indexs(1)=i
        jindexs(1)=j
        do k=(i+1),nfiles    !Look for matches in subsequent files.
            do l=1,ra.noe(k)
                if((ra.z(1,k).ne.0).and.    !Use only those that have not been used before.
                    ((ra.a(1,k).eq.ra.a(j,i)).and.
                    (ra.id(1,k).eq.ra.id(j,i)).and.
                    (ra.z(1,k).eq.ra.z(j,i))))then    !Match found.
                    ndatum=ndatum+1
                    values(ndatum)=ra.mc(1,k)
                    errors(ndatum)=ra.err(1,k)
                    indexs(ndatum)=k
                    jindexs(ndatum)=1
                endif
            enddo
        enddo
    enddo
enddo

```

```

355      call wavg_o(values,errors,ndatum,doout1,mean,stdev,wmean,
      wstdev,rchisq,ifvald)
      do k=1,ndatum      !Is the primary (i.e., i) detector's value valid?
      if((indexs(k).eq.i).and.(.not.ifvald(k)))then      !Resubmit just the invalid ones.
      nvalus=0      !Track "new" NDATUM.
      do l=1,ndatum
      if(.not.ifvald(l))then      !Use this one.
      nvalus=nvalus+1
      values(nvalus)=values(l)
      errors(nvalus)=errors(l)
      indexs(nvalus)=indexs(l)
      jindexs(nvalus)=jindexs(l)
      endif
      enddo
      ndatum=nvalus
      goto 355      !Try WAVG_O again.
      endif
      enddo
      if(ifirst)then      !Print the heading.
      numchr=INDEX(P.VOLUME_UNITS,' ')-1
      WRITE(6,8100)i,'uCi',P.VOLUME_UNITS(1:numchr)

      ifirst=.false.      !Do not print the heading again for this detector.
      endif
      spcusd=blanks(1:16)      !Construct the list of detectors used.
      numchr=0
      do k=1,ndatum
      if(ifvald(k))then
      if(numchr.eq.0)then
      spcusd(1:1)=char(indexs(k)+48)
      numchr=1
      else
      spcusd((numchr+1):(numchr+3))=', '//
      char(indexs(k)+48)
      numchr=numchr+3
      endif
      endif
      enddo
      WRITE(6,8110)ra.Z(j,I),ra.ID(j,I),ra.A(j,I),
1      wmean,abs((wstdev/wmean)*100.0),spcusd      !Error back to percent.
      do k=1,ndatum      !Mark as used those that were valid.
      if(ifvald(k))ra.z(jindexs(k),indexs(k))=0
      enddo
      endif
380      enddo
390      enddo

```

C

C End of the main routine.

C

CALL EXIT

END

C.17 DCL Command File for Linking BATCH_GIDXC_SUMMARY

```
$ LINK/EXE=INEL$EXE:BATCH_gidxc_summary BATCH_gidxc_summary,wavg_o,sigfig, -  
GAP$LIB/LIB,VT100$LIB/LIB
```

C.18 DCL Command File for Executing GIDXC within a Batch Queue

```
$!BATCH ANALYSIS COMMAND FILE ISSUED FROM GC PROGRAM  
$!   EWK 3/4/92  
$!  
$ ! Based on E. W. Killian's BATCH_DXC_ANALYSIS.COM to perform GIDXC analysis.  
$ ! Modified to analyze all spectra specified in the /PARAMETER qualifier (up  
$ ! to six spectra can be handled) and then generate a summary if all of the  
$ ! specified spectra were successfully analyzed.  D. A. Femec, June 14, 1993.  
$ !  
$ ! Modified so that the first parameter passed (p1) indicates whether a batch  
$ ! summary should be generated (set to Y to do so).  D. A. Femec, July 1, 1993.  
$ !  
$ ! Modified to look for the spectrum in the GE$SAVE directory, and if it is  
$ ! not found there, to look for it in the OLDSWEPP$SAVE directory.  If it is  
$ ! found there, reset GE$SAVE and BK$SAVE appropriately.  Also modified to  
$ ! scan the spectrum p<n> parameters in order rather than duplicating the code  
$ ! for each n.  D. A. Femec, January 27, 1994.  
$ !  
$ ! Inserted a new first parameter to specify whether the individual detector  
$ ! analyses should be printed; the user_mode logical IF$PRINT is used to pass  
$ ! this parameter to BATCH_GIDXC_ANALYSIS.  D. A. Femec, February 24, 1994.  
$ !  
$ ! There needs to be at least one spectrum.  
$ !  
$ if p3 .eqs. "" then exit  
$ !  
$ ! For each spectrum, temporarily stored in p9, . . .  
$ !  
$ index = 1  
$ p9 = p3  
$ next:
```

```

$ if p9 .eqs. "" then goto summary
$ if f$search("ge$save:''p9'.dgc",1) .eqs. ""
$ then
$     if f$search("oldswepp$save:''p9'.dgc",1) .eqs. ""
$     then
$         write sys$output "SPECTRUM " + p9 + " NOT FOUND"
$         exit
$     else
$         define/user_mode ge$save oldswepp$save
$         define/user_mode bk$save oldswepp_bk$save
$     endif
$ endif
$ DEFINE/USER_MODE A1$SPECTRUM$ID 'p9'
$ DEFINE/USER_MODE IF$PRINT 'p1'
$ RUN INEL$EXE:BATC_H_GIDXC_ANALYSIS
$ index = index + 1
$ if index .eq. 2 then p9 = p4
$ if index .eq. 3 then p9 = p5
$ if index .eq. 4 then p9 = p6
$ if index .eq. 5 then p9 = p7
$ if index .eq. 6 then p9 = p8
$ if index .eq. 7 then goto summary
$ goto next
$ !
$ ! Compile the summary report if requested.
$ !
$ summary:
$     if p2 .nes. "Y" then exit
$     @inel$exe:batch_gidxc_summary 'p3' 'p4' 'p5' 'p6' 'p7' 'p8'
$ !
$ ! Exit
$ !
$EXIT

```

C.19 DCL Command File for Generating a GIDXC Container Summary

```

$ ! Routine to generate a summary report for SWEPP barrel analysis (using up to
$ ! six detectors). The spectrum id(s) are passed to this routine via p<n>
$ ! parameters, and are passed to the executable via A<n>$SPECTRUM$ID logicals.
$ !
$ ! Written by D. A. Femec, January 11, 1994.
$ !
$ set verify
$ if p1 .nes. "" then define/user_mode A1$SPECTRUM$ID 'p1'
$ if p2 .nes. "" then define/user_mode A2$SPECTRUM$ID 'p2'

```

```

$ if p3 .nes. "" then define/user_mode A3$SPECTRUM$ID 'p3'
$ if p4 .nes. "" then define/user_mode A4$SPECTRUM$ID 'p4'
$ if p5 .nes. "" then define/user_mode A5$SPECTRUM$ID 'p5'
$ if p6 .nes. "" then define/user_mode A6$SPECTRUM$ID 'p6'
$ !
$ ! Compile a summary report.
$ !
$ ! summary:
$ run inel$exe:batch_gidxc_summary
$ !
$ ! Exit
$ !
$EXIT

```

C.20 AT_TABLE Include File

```

C STRUCTURE FOR THE ATTENUATION TABLE
C STRUCTURE WILL BE LOADED WITH THE VALUES IN EITHER THE Si OR Fe table
C
C Expanded to accept 100 energy-attenuation factor pairs. D. A. Femec,
C January 5, 1994.
C

```

```

STRUCTURE /ATABL/
  INTEGER*4 NP !NUMBER OF POINTS IN TABLE
  REAL*4 ATF(100) !ATTENUATION FACTORS
  REAL*4 ENERGY(100) !ENERGY VALUES
END STRUCTURE
RECORD /ATABL/AT
COMMON /AT_TABLE/AT

```

C.21 BRANCH Include File

```

C STRUCTURE CONTAINS THE BRANCHING RATIOS AND UNCERTAINTIES FROM THE NUCLIDE
C LIBRARY DATA BASE
C

```

EWK 12/13/93

```

STRUCTURE /BRANCH_RATIO/

```

```

      CHARACTER*4 ID
      INTEGER*4  Z
      INTEGER*4  A
      INTEGER*4  NR      !NUMBER OF LINES FOR THIS NUCLIDE
      REAL*4      ENERGY(400) !LINE ENERGY
      REAL*4      BR(400)      !BRANCHING RATIO
      REAL*4      BR_UC(400)   !UNCERTAINTY IN BR
END STRUCTURE
STRUCTURE /U_BR/
  UNION
    MAP
      RECORD /BRANCH_RATIO/E(500)
    END MAP
  END UNION
END STRUCTURE
RECORD /U_BR/BRU

```

C.22 FIND_LINE_STRUCT Include File

```

C STRUCTURE CONTAINING THE RESULTS FROM A FIND_LINE
C
C Modified to also include the half-life (see FIND_LINE.FOR for code changes).
C D. A. Femec, July 22, 1993.
C
C Modified to also include the decay correction (see FIND_LINE.FOR for code
C changes). D. A. Femec, January 25, 1994.
C

```

```

  STRUCTURE /FIND_LINE_DATA/
    REAL*4  LMC
    REAL*4  BR
    REAL*4  DCA
    REAL*4  ERR
    REAL*4  FLAG
    REAL*4  CPS
    REAL*4  EFF
    REAL*4  HL
  END STRUCTURE
RECORD /FIND_LINE_DATA/FL
COMMON /FIND_LINE_DATA_COMMON/FL

```