UCRL-CR--107467-Vol.3

DE92 009456

Plutonium Isotopic Analysis System for Plutonium Samples Enriched in ^{238}Pu in EP 60/61 and Fuel-Clad Containers

Volume 3
Part 2: Software Listings

Wayne D. Ruhter

July 1, 1991

Prepared under the auspices of WestinghouseHanford Company in partial fulfillment of Memorandum Purchase Order MJL-XMV-012823.

LAWRENCE LIVERMORE NATIONAL LABORATORY
University of California • Livermore, California • 94551

DISCLAIMER

DISCLAIMER

PREFACE

This software manual is addressed to the Westinghouse Hanford's Radioisotope Power Systems Facility personnel (programmers and supervisors) who maintain the software on the Pu-238 isotopic analysis system. The document is divided into two parts. Part 1 describes the computer codes that control the system, analyze the spectral data, and determine the relative plutonium abundances. Part 2 contains the software listings of the analysis codes.

In addition to this manual, two other manuals complete the description of the Pu-238 isotopic analysis system. The Pu-238 Isotopic Analysis Users Manual (UCRL-CR-107467, Vol. 1) explains how to operate and calibrate the system. The Pu-238 Isotopic Analysis Hardware Manual (UCRL-CR-107467, Vol. 2) describes the system components and provides service and maintenance information. In that volume are detailed instructions on required settings for the nuclear instrument modules.

All commercial items mentioned in this manual are assumed to be functioning correctly for the purposes of system operation. Users are referred to individual equipment manufacturers' manuals for details of operation, troubleshooting, and maintenance of this commercial equipment.

NOTE: If any of the Pu-238 Isotopic Analysis codes are subsequently changed, users are asked to print out the changed listing and replace the listing now in this manual. In that way, this document will be kept current.

CONTENTS

PART 2

Chapter 11

Chapter 12

Chapter 13

```
C*****************************************************************
C
C  ACQR38       (ACQR90 subroutine for PU238 instrument)
C
C      This routine is a special adaptation of the ACQR90
C      routine for the Pu238 isotopic analysis instrument.
C      Routine is instrument specific to Canberra Series 90
C      data collection.
C      Acquire data on the Series 90
C      Set up to send 'AS' (suspend acquisition) and 'AR'
C      (resume acquisition) to VAX
C
C      SSJ 20-JUL-79
C      Rev. 19-jun-84 lc
C      Rev. 25-sep-84 cms
C      Rev. 26-OCT-84 MJC      Split ACQR90 and CMPLTN subroutines
C      Rev. 31-OCT-84 MJC      Deleted LOGICAL * 1 TKEY & added it
C                              CONTRL.CMN
C      Rev. 21-DEC-84 WDR      Modified for the solid isotopic instrument
C      Rev. 03-JAN-85 MPK      Restored DATTIM option argument
C                              Added VAX comm for suspend and resume
C      Rev. 15-JAN-85 MPK      Disabled T key during MC runs
C      Rev. 21-JAN-85 MPK      Check VAXUP in comm calls
C      Rev. 04-FEB-85 MPK      Add sleep time in ADC/terminal
C                              polling loop
C                              Put END= and ERR= returns in terminal
C                              read after suspend, to prevent operator
C                              from bombing the program
C      Rev. 08-MAR-85 MPK      Include CVAX common
C      Rev. 27-JAN-86 WDR      Test for rotation before turning on ADCs
C                              to make sure it is rotation is enabled
C      Rev. 02-APR-86 WDR      Modified for Micro VAX II
C      Rev. 05-JUL-87 RDP      Modified for uVAX II w/ ADAC Drivers
C      Rev. 15-JUL-87 RDP      Using Single Key Input for VAX
C      Rev. 18-AUG-87 RDP      Included calls to control Load/Unload,
C                              Rotation, and SET lights.
C      REV. 04-SEP-87 RDP      Include a resume fcn, if assay is
C                              suspended due to rotation error.
C      Rev. 19-Jan-88 WDR      Sleep call changed to real parameter
C      Rev. 30-Aug-89 WDR      Modified for Pu238 instrument
C      Rev. 02-May-90 WDR      Removed COMM calls for Pu238 instrument
C      Rev. 08-May-91 WDR      Added test for sample type in elevator
C                              motion check
C*****************************************************************
C
      SUBROUTINE ACQR90
C-----------------------------------
C*****************************************************************
C Real Time Interrupt Variables for Keyboard Interrupt
C
      COMMON/RTI/       INKEY,INCHAR
      INTEGER           INKEY
      CHARACTER         INCHAR*1
C
      LOGICAL*1 EFLG, RFLG, ROTERR, ELERR
      CHARACTER ACHAR*1
      INTEGER*2 ON6,OFF6
      INTEGER*4 ISTAT,IMASK
C
C  The following common block files are generic for all instruments
```

```
C
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'ADAC.CMN'
      INCLUDE 'CTIM.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
      INCLUDE 'CONT90.CMN'
      INCLUDE 'CSPECT.CMN'
C
C Disable LOAD/UNLOAD sample capability; enable rotation
C
      CALL ROTMON(RFLG,EFLG)
      IMASK = 512
      ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
      CALL ERROR_CHECK(ISTAT,IOSB)
C
C  We are generating live data; set the live flag
C  Reset the termination flag
C
      LIVE = .TRUE.
      TKEY = .FALSE.
C
C  Clear the time
C  Erase the existing spectrum unless this is a non-first cycle
C  of a cumulative run

 12   CONTINUE
      CALL CTIM(IADC,,IERR)
      CALL CHK90('CTIM  ',IERR)
      CALL CDAT(IADC,,IERR)
      CALL CHK90('CDAT  ',IERR)
C
C
C  Preset (clock) time
C  Start collecting
C
 13   CONTINUE
      CALL PSET(IADC,,1,1,TIMEPR,0.0,0,0)
C
C  Check to see that sample is rotating
C
      RFLG = .TRUE.
      EFLG = .TRUE.
      CALL ROTMON(RFLG,EFLG)
C     TYPE *,'EFLG IN ACQR38',EFLG,MATTYP(1)
      IF (RFLG) GOTO 14
      TYPE 1003
1003  FORMAT(//,' !! Sample not rotating.  Enable rotation !!')
      TYPE1004,'Press ENTER to retry (type A and a <CR> to ABORT) ->'
      READ (5,1006,END=13,ERR=13) ACHAR
      IF ((ACHAR.EQ.'A').OR.(ACHAR.EQ.'a')) THEN
         TYPE*,' Abort Process -- Problem with Rotation Motor'
         ABORT = .TRUE.
         GOTO 400
```

```
        ENDIF
1004    FORMAT($,' ',A)
1006    FORMAT(80A1)
 14     CONTINUE
        IF (.NOT.EFLG .AND. MATTYP(1).EQ.'E') THEN
        TYPE 1113
1113    FORMAT(//,' !! Sample elevator not working.  Enable elevator !!')
        TYPE1004,'Press ENTER to retry (type A and a <CR> to ABORT) ->'
        READ (5,1006,END=13,ERR=13) ACHAR
        IF ((ACHAR.EQ.'A').OR.(ACHAR.EQ.'a')) THEN
           TYPE*,' Abort Process -- Problem with Sample Elevator'
        ABORT = .TRUE.
           GOTO 400
        ENDIF
        ENDIF
 16     CONTINUE
        CALL CMPLTN
C
C       Now turn on ADCs
C
 15     CONTINUE
        CALL COLL(IADC,,IERR)
        CALL CHK90('COLL  ',IERR)
C
C  Check the status of the ADC
C  If it is idle, acquisition has completed
C
50      I = INQADC(IADC,ISTAT,MEMSIZ,IGAIN,IRANGE,IOFF,INPUTS)
        CALL CHK90('INQADC',I)
        IF (I .EQ. 0 ) GO TO 100
        CALL SLEEP(0.5)

C  Check sample rotation and elevator; if stopped, terminate measurement
C
        ROTERR=.FALSE.
        ELERR =.FALSE.
        CALL ROTMON(RFLG,EFLG)
        IF (RFLG .AND. EFLG)GO TO 51              !No problem
        CALL DATTIM(1)                          !Rotation stopped; suspend meas.
        IF (.NOT. RFLG)THEN
        TYPE 1005,NOWTIM
1005    FORMAT(' SAMPLE NOT ROTATING.  MEASUREMENT SUSPENDED AT ',A8//)
        ACHAR = 'S'
        ROTERR= .TRUE.                          !Suspended by Rotation Error
        GOTO 55
        ELSE
C       TYPE *,EFLG,MATTYP(1)
        IF (MATTYP(1) .EQ. 'E')THEN
        TYPE 1016,NOWTIM
1016    FORMAT(' SAMPLE ELEVATOR STOPPED.  MEASUREMENT SUSPENDED AT ',A8//)
        ACHAR = 'S'
        ELERR = .TRUE.                          !Suspended by elevator problem
        GOTO 55
        ENDIF
        endif
C
   Acquisition not complete
L  Check for quit (Q) or suspend (S) or terminate (T) key on console
C
 51     CONTINUE
```

```
         IF (INKEY.NE.0) THEN
            ACHAR = INCHAR                  ! SET KEY VALUE
            INKEY = 0                       ! RESET INKEY FLAG
         ELSE
            GOTO 50
         ENDIF
C
C  Terminal input seen - check whether action required
C
 52      CONTINUE
         IF (ACHAR .EQ. 'Q') GOTO 55
         IF (ACHAR .EQ. 'S') GOTO 55
         IF (ACHAR .NE. 'T') GOTO 50
C
C  If we are here, T key was seen.  Allow termination unless MC
C  run is in progress.
C
         IF (.NOT. (MP.OR.MB)) GO TO 55
         TYPE 1002
1002     FORMAT(/20X,' T option ignored during meas. control;'
     1                 'use Q to quit.'/)
         GOTO 50
C
C  Quit (Q) or suspend (S) or terminate (T) key recognized
C  Turn on the keyboard and stop the analyzer
C
  55     CONTINUE
         CALL ADCOFF(IADC,,IERR)
         CALL CHK90('ADCOFF',IERR)
         IF (ACHAR .EQ. 'Q') GOTO 70
         IF (ACHAR .EQ. 'T') GOTO 60
         IF (.NOT. RFLG .OR. (.NOT. EFLG .AND.
     +   MATTYP(1).EQ.'E')) GOTO 56
C
C  Suspended by S key
C
         TYPE 1008,NOWTIM
 1008    FORMAT(//20X,' Acquisition suspended by S key at ',A8//)
         TYPE *
 56      TYPE 1058
 1058    FORMAT('$To resume, press RETURN (press A to abort) ->')
 57      IF (INKEY.NE.0) THEN                !Wait for Key Hit
            INKEY = 0
            IF ((INCHAR.EQ.'A').OR.(INCHAR.EQ.'a')) THEN
               IF (.NOT. RFLG) THEN
                 TYPE*,' Abort Process -- Problem with Rotation Motor'
               ELSE
                 TYPE*,' Abort Process -- Problem with Sample Elevator'
               ENDIF
               ABORT = .TRUE.
               GOTO 75              !exit acquistion and clean up
            ELSE
               GOTO 58              !resume
            ENDIF
         ELSE
            CALL ROTMON(RFLG,EFLG)
            IF (RFLG) THEN
               CALL SLEEP(2)                !Wait a sec to make sure
               CALL ROTMON(RFLG,EFLG)
            ENDIF
```

```
            IF (RFLG.AND.ROTERR) THEN     !If rotation has restarted
               ROTERR=.FALSE.
               GOTO 58
            ELSE
            IF ((EFLG.and. ELERR) .and. RFLG) THEN
               ELERR = .FALSE.
               GOTO 58
            ENDIF
            ENDIF
            GOTO 57
         ENDIF
 58      CONTINUE                          !Resume
1001    FORMAT(A1)
         CALL DATTIM(1)
C        IF (VAXUP) CALL COMM('AR')        !*Communications - assay resumed
         TYPE 1007,NOWTIM
 1007    FORMAT(//20X,' Acquisition resumed at ',A8//)
         GO TO 15
C
C  Terminated by T key
C  Stop cycling
C
   60    CALL DATTIM(1)
         TYPE 1019,NOWTIM
 1019    FORMAT(//20X,' Acquisition terminated by T key at ',A8//)
         TKEY = .TRUE.
         STPCYC = .TRUE.
         GO TO 200
C
C  Aborted
C
   )     ABORT = .TRUE.
         TYPE 1000,NOWTIM
 1000    FORMAT(//20X,' Acquisition aborted by Q key at ',A8//)
 75      CONTINUE
C Turn off rotation, enable LOAD/UNLOAD
C Enable LOAD/UNLOAD
400      IMASK = 512
         CALL = BCLRW(IMASK,H_DATA,IOSB,,,)
         CALL ERROR_CHECK(ISTAT,IOSB)
         GO TO 500
C
C
 100     CONTINUE
C
C  Acquisition stopped
C  Retrieve the live (0) and clock (1) times
C
 200     CALL FROMCA(IADC,2,0,SPCTRM,IERR,)
         CALL CHK90('FROMCA',IERR)
         TIMELV = SPCTRM(1)
         TIMECL = SPCTRM(2)
C
C
 500     RETURN
         END
```

```
C*************************************************************************
C
C ADATE
C
C       Set the date and time for assay

C       16-Jan-85 MPK
C
C*************************************************************************
C
        SUBROUTINE ADATE
C
        INCLUDE 'CDATE.CMN'
C
        CALL DATTIM(1)
        ASDATE=NOWDAT
        ASTIME=NOWTIM
        RETURN
        END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       ADAC_ERR
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C
        LOGICAL*1 FUNCTION ERROR_CHECK(ISTAT,IOSB)
C
C FUNCTION TO REPORT I/O ERRORS WITH ADAC CALLS.
C
        INTEGER*2       IOSB(4)
        INTEGER*4       ISTAT
        CHARACTER*80    MESSAGE
C
        IF (ISTAT.EQ.1) GOTO 100
        CALL LIB$SYS_GETMSG(ISTAT,,MESSAGE,,)
 50     TYPE*, MESSAGE
        ERROR_CHECK = .FALSE.
        RETURN
C
 100    I = IOSB(1)
        IF (I.EQ.1) GOTO 200
        CALL LIB$SYS_GETMSG(I,,MESSAGE,,)
        GOTO 50
 200    ERROR_CHECK = .TRUE.
        RETURN
        END
```

```
C*******************************************************************************
C
C AEND
C
C       Do VAX communication at end of assay.
        Message type 'AE' is sent unless the assay was terminated
C       early, in which case 'AT' is sent.  'AQ' is not sent,
C       because it is generated elsewhere.
C
C       Enable sample UNLOAD and disable sample rotation.
C
C       16-Jan-84 MPK
C       25-Aug-87 WDR
C       11-Nov-87 WDR  Added test for autocycle
C
C*******************************************************************************
C
        SUBROUTINE AEND
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CVAX.CMN'
        INCLUDE 'CASSAY.CMN'
C
        INCLUDE 'ADAC.CMN'
C
C Test for autocycle and number of cycles.  If not done, don't do
C do the following
C
        IF (.NOT.AUTOCY)GO TO 5
        IF (ICYCLE.LT.NUMCYC)GO TO 7
C
C Turn off rotation, enable LOAD/UNLOAD
5       IMASK = 4096
C     Send HIGH to reset
C       ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
C       CALL ERROR_CHECK(ISTAT,IOSB)
C     Return to LOW
C       ISTAT = BCLRW(IMASK,H_DATA,IOSB,,,)
C       CALL ERROR_CHECK(ISTAT,IOSB)
C       IMASK = 2048
C     Send HIGH to reset
C       ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
C       CALL ERROR_CHECK(ISTAT,IOSB)
C     Return to LOW
C       ISTAT = BCLRW(IMASK,H_DATA,IOSB,,,)
C       CALL ERROR_CHECK(ISTAT,IOSB)
C Turn Off Rotation
        IMASK = 1
        ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C Enable LOAD/UNLOAD
        IMASK = 512
        CALL = BCLRW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
7       RETURN
        END
```

```
C*****************************************************************************
C
C  ASOPT
C
C       Responds to the assay option selected by the operator
        from the menu.  Calls to perform the assay, print the
C       results, and write data to files are performed here.
C       The data is sent to the VAX from this routine also.
C
C
C       CMS 10-MAY-84
C       REV.11-SEP-84 CMS
C       REV 11-NOV-84 WDR       Removed WRTDAT since already done for LLNL
C                               instruments.  Livermore systems always write
C                               data to disk.
C       Rev 29-Nov-84 MPK       Replace list-directed with
C                               formatted I/O
C       Rev 04-Jan-85 MPK       Remove VAX communication that is
C                               handled in ASSAY - change rest
C                               to meet specs
C       Rev. 18-JAN-85  MPK     Add calls to ADATE and AEVAX
C       Rev. 21-JAN-85  MPK     Check VAXUP in comm calls
C       Rev. 12-FEB-85  MPK     Redo logical flags
C       Rev. 20-FEB-85  MPK     Added assay type 'A','S' for log
C       Rev. 29-APR-85  MPK     Send 'MS' message to VAX
C       Rev. 02-APR-86  WDR     Modified for Micro VAX II
C       Rev. 13-Jul-87  RDP     Modified for uVax II
C
C*****************************************************************************
C
        SUBROUTINE ASOPT

C  The following common block files are generic for all instruments
C
        INCLUDE 'CDIAG.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CVAX.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CLOG.CMN'
        INCLUDE 'CMSG.CMN'
C
C  The following common block files are instrument specific for
C  gamma and x-ray instruments
C
        INCLUDE 'CSPECT.CMN'
C
        ASFLG = .TRUE.
        RTYP = ' Assay  '
C
C  Check that measurement control has been performed
C  within the required time limits and without
C  error
C
        CALL MCCHK
        IF(ABORT) GO TO 500
C
C  Get set up parameters from the dialog
```

```
C  Filename returned in common if writing to disk
C
      CALL ADATE
      CALL DIALOG
      IF (ABORT) GOTO 500

C  Announce assay to VAX and to operator
C
      TYPE 1000
 1000 FORMAT (/'   Assay beginning')
C     IF (VAXUP) CALL COMM('SA')      !***communications - start assay
  100 CALL ASSAY
      CALL AEND
      IF(ABORT)GO TO 500
C
C  Print results of assay - if diagnostic error flag is set
C  write result with error message
C
      CALL PRINT1                     !instrument specific
C
C  Check if VAX is up - send assay results to VAX
C  If VAX is down and writing raw data - write a temporary file
C
      IF (.NOT.VAXUP) GOTO 400
C 300 CALL COMM('RA')                 !*communications-send results
  300 CONTINUE
C     IF(VAXUP.AND.RAWDTA)CALL COMM('RD')!*communications-send raw data
      ICHAR = 0
C
C  Write assay results and ancillary information to the assay
C  log file 'ASLOGF.INS'

  400 LVFLAG = VAXUP
      ASTYPE(1) = 'A'
      ASTYPE(2) = 'S'
      CALL WRITLG
  500 RETURN
      END
```

```
C
C*********************************************************************
C
C  ASSAY38      (ASSAY routine for Pu238 isotopic instrument)
C
      Routine performs assay runs for assay,background,
C     measurement control, calibration, from disk options.
C     It then calls the analysis routines and the data
C     diagnostic routines.
C
C     CMS  26-APR-84
C     Rev. 05-OCT-84 CMS ADDED ABORT=.TRUE. IF ERROR IN OPENING FILE
C     Rev. 22-OCT-84 MJC Changed 'BACKGR.DAT' to BKFIL in opening DKLUN1
C     Rev. 01-NOV-84 wdr adapted to LIVERMORE needs for solid isotopic
C                        instrument
C     Rev. 17-DEC-84 WDR modified for solid isotopic instrument
C     Rev. 03-JAN-85 MPK Restored DATTIM option argument
C                        Revised COMM calls to follow specs
C     Rev. 18-JAN-85 MPK Removed calls to send 'AE' and 'AT'
C                        messages, which will be handled elsewhere.
C                        Sending 'AQ' is still done here.
C                        Removed dead code at 501.
C     Rev. 13-FEB-85 MPK Redo logical flags
C     Rev. 20-FEB-85 MPK Remove Ref to CS90.CMN
C                        Continue line that was being cut off
C     Rev. 02-APR-86 WDR Modified for Micro VAX II
C     Rev. 30-Aug-89 WDR Modified for Pu238 instrument
C
C*********************************************************************
C
C
      SUBROUTINE ASSAY
      LOGICAL * 1 IANS
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CTIM.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CLOG.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'CMCONS.CMN'
      INCLUDE 'CBACKG.CMN'
C
C  The following common block files are instrument specific for
C  gamma ray instruments
C
      INCLUDE 'CSPECT.CMN'
      INCLUDE 'CONT90.CMN'
      INCLUDE 'CONTRS.CMN'
C
C  Set spectrum pointer to guarantee fresh data

      IFIRST = -500            !instrument specific
C
C  Check if assay is from disk - open file for analysis
```

```
C
      IF(AC) GO TO 200          !current data in MCA
      IF(.NOT. FROMDK) GO TO 130
      OPEN(UNIT=DKLUN,NAME = FILNAM,ACCESS = 'DIRECT',TYPE = 'OLD',
     1       RECORDSIZE = 128,ERR = 504)
      GO TO 200
  504 CALL ERRMSG(1,FILNAM)
      ABORT = .TRUE.
      GO TO 500
C
C  Read data from disk file or read current data
C  to retrieve information needed for analysis
C
  200 CALL DATA90(0,4,SUM)      !DATA90 is instrument specific
        TIMELV = SPCTRM(1)      !Livetime
        TIMECL = SPCTRM(2)      !Real time
      IF(AC ) GO TO 180
      GO TO 184                 !Assay data from disk
C
C  Do the actual data collection
C
  130 CALL DATTIM(1)
      CALL ACQR90                       !collect data - instrument specific
      CALL DATTIM(1)
      IF(.NOT. ABORT)GO TO 180
C      IF (VAXUP) CALL COMM('AQ')       !*communications-assay quit
      GO TO 500
180   CALL WRTDAT     ! write data to disk file - write ADC #1
      CLOSE(UNIT=DKLUN)
184   CALL P238GRP    !instrument specific routine
      CLOSE(UNIT=DKLUN)
  500 RETURN
      END
```

```
C*******************************************************************************
C
C  ASYP238
C
C        main program for the SRP_PU238 program

C        MPK 21-Nov-84
C
C*******************************************************************************
C
         PROGRAM ASYP238
         CALL EXEC
         END
```

```
C
C*********************************************************************
C
C  AUTCYC
C
C       Responds to the operator request for assay in autocycle
C       mode where assays are collected repeatedly for the number
C       of times input in the dialog mode.  This routine also
C       responds to autocycle from disk
C
C       CMS 10-MAY-84
C
C       Rev. 25-SEP-84 CMS
C       Rev. 16-NOV-84 WDR      modified standard deviation calculation
C       Rev. 04-JAN-85 MPK      changed VAX COMM to conform to specs
C       Rev. 18-Jan-85 MPK      Add calls to ADATE and AEVAX
C                               Add logic to handle 'T' option
C       Rev. 22-Jan-85 MPK      Check VAXUP in COMM calls
C                               correct statistics
C       Rev. 12-Feb-85 MPK      Redo logical flags
C
C       Rev. 20-Feb-85 MPK      Declared XSSQ,XMEAN,XSUM as REAL*8
C                               Added assay type 'A','S' for log
C       Rev. 02-May-90 WDR      Removed AEVAX and COMM calls
C
C*********************************************************************
C
       SUBROUTINE AUTCYC
       REAL*8 XSSQ, XMEAN, XSUM
       LOGICAL * 1 SUMMRY
C
C  The following common block files are generic for all instruments
C
       INCLUDE 'CONTRL.CMN'
       INCLUDE 'CVAX.CMN'
       INCLUDE 'CFILES.CMN'
       INCLUDE 'CDIAG.CMN'
       INCLUDE 'CBACKG.CMN'
       INCLUDE 'CLUNS.CMN'
       INCLUDE 'CASSAY.CMN'
       INCLUDE 'CLOG.CMN'
       INCLUDE 'CQSUMS.CMN'
C
C  The following common block files are instrument specific for
C  gamma ray instruments
C
       INCLUDE 'CSPECT.CMN'
C
       RTYP = 'Autocycl'
       AUTOCY= .TRUE.
       ASFLG = .TRUE.
C
       XMEAN = 0.0
       STDEV = 0.0
       VAR   = 0.0
       XSQ   = 0.0
       ICYCLE = 0
C
C  Check that measurement control has been performed within
C  the required time and without error
```

```
C
        IF(FROMDK) GO TO 100
        CALL MCCHK
        IF(ABORT) GO TO 500
 100    CALL DIALOG
        IF (ABORT) GO TO 500
        CALL NEWNAM(0)
C
C  Collect assays for the number of times returned from dialog
C
 200    DO 10 I = 1, NUMCYC
        CALL ADATE
C       IF (VAXUP) CALL COMM('SA')              !*communications - start assay
        CALL ASSAY
        IF(ABORT)GO TO 500
        ICYCLE = I
C
C  If autocycle from disk - don't send results to VAX
C
        IF(FROMDK) GO TO 30
C
C  Write assay log entry
C
   35   LVFLAG = VAXUP
        ASTYPE(1) ='A'
        ASTYPE(2) ='S'
        CALL WRITLG
C
C  Print results
C
  30    CALL PRINT1                             !instrument specific routine
        IF (.NOT.TKEY) GO TO 32
        ICYCLE = ICYCLE -1                      !Reset ICYCLE
        GO TO 15
C
C  Accumulate statistics
C
32      XMEAN = XMEAN + ASANS
        XVAL(I)= ASANS          !save answer for standard deviation calc
C
C  Create next filename
C
        IF(I .NE. NUMCYC) CALL NEWNAM(1)
  10    CONTINUE
C
C  Calculate average and standard deviation from assay results
C
15      IF (ICYCLE.LE.1)GO TO 500
        XMEAN = XMEAN/ICYCLE
        DO 50 J = 1,ICYCLE
        XSQ = XSQ + (XVAL(J) - XMEAN)**2
50      CONTINUE
        STDEV = SQRT(XSQ)/(ICYCLE-1)
C
        NUMCYC = ICYCLE
        SUMMRY = .TRUE.
        CALL HEADER(SUMMRY)
        WRITE(LOUT, 1000)ICYCLE,XMEAN, STDEV
 1000   FORMAT(' Average from  ',I3,' runs  =  ',F10.4,
     1          ' standard deviation =  ',F10.4)
```

```
500     CALL AEND
        RETURN
        END
```

```
C       C       C       C       C       C       C       C       C       C

        SUBROUTINE CALEFF(EFCOF,EPK,EMVLOG,DEFLOG)

        DIMENSION EFCOF(2)

        IF(EPK .GT. EFCOF(12))GO TO 20  !Test vs. Xover energy.
        DEFLOG = EFCOF(7)
        DO 10, I = 8,11
10      DEFLOG = DEFLOG + EFCOF(I) * EMVLOG**(I-7)
        GO TO 50
20      DEFLOG = EFCOF(1)
        DO 40, I = 2,6
40      DEFLOG = DEFLOG + EFCOF(I) * EMVLOG**(I-1)
50      RETURN
        END
```

```
C
      SUBROUTINE CALGPM (EPK,VALUE,PCERR)
C     Last edit 6-Mar-85 (J.B.N.)
C     Last edit 27-Aug-85 (R.G.)  To self- absorption correction
C     Edit 12-Sep-88 (r.g.) SSMU
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'

      INTEGER*2 MTRL
      DATA MTRL/'GE'/

      ELOG = ALOG (.001 * EPK)
      SMU = 0.0
      GAMAS = VALUE
      IF (NF1 .EQ. 0)GO TO 10
      CALL CALMU (NF1,SMU,EPK,ELOG,MTLZ,CMPOS,NSMBL,EDGE,COFMU)

10    SSMU = SMU
C     IF (RUNFLG(4) .EQ. 0)THEN
      IF (((DCNST(1).EQ.0) .AND. (DCNST(12).GT.0)) .OR.
     1   (GEOM - DCNST(15) .LE. 0))THEN
        VALUE = 1.0    !Skip efficiency corrections, but allow
        IF (SMU .GT. .001)  GAMAS = GAMAS *SMU / (1. -EXP(-SMU))
        SSMU = 2.*SMU !To increase the error assoc. with value
        GO TO 20      !absorber corrections.
      ENDIF

C     [DONT change DEPTH in common-- GPM will use 1/2 depth.]
      CALL GPM (EPK,ELOG,GEOM,SURFC,DEPTH,SMU,VALUE,GAMAS,DCNST)
      IF (EPK .GT. 500.)GO TO 20

C     CORRECT FOR GE "DEAD LAYER" ATTENUATION.

      CALL CALMU (1,XMU,EPK,ELOG,MTRL,SMU,NSMBL,EDGE,COFMU)
      GAMAS = GAMAS * EXP (XMU)
20    IF (NF2 .EQ. 0)GO TO 30          !No absorber present.
      ATN = 0.0

      CALL CALMU (NF2,AMU,EPK,ELOG,NABS,ABSRB,NSMBL,EDGE,COFMU)
C
C     ADJUST ABSORBER ATTENUATION OF PHOTONS TO ACCOUNT FOR GAMMAS
C     TRAVERSING THE ABSORBER AT SKEWED ANGLES.
C
C     WRITE (*,*)'AMU',EPK,AMU
      AMU = AMU * VALUE
      IF (AMU .LT. 6.9)ATN = EXP (AMU)
      GAMAS = GAMAS * ATN
C
C     CALCULATE ERROR IN ATTENUATION USING AN ERROR OF 2%
C     IN ABSORBER THICKNESSES
C
C     ADRA =  2.*AMU
      ADRA = 1.3
      PCERR = SQRT(PCERR**2 + SSMU**2 + ADRA**2)
30    VALUE = GAMAS
      RETURN
      END
```

```
        SUBROUTINE CALIB
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       GRPANL sub to calculate GAIN, ZERO, & standard reference points
   corrected for linearity. Called once for each grouping.
C
C               By J.B.Niday    Last edit 25-Sept-87 (JBN)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        INCLUDE 'GRPLGCM.PMS'

        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CDIAG.CMN'

        DIMENSION XC(2),TREFCH(2)
C       DATA LP/-1/
        DATA LP/6/
        IF(NUSHPS .NE. 0) THEN          !Parameter specification has precedence.
                NOC = 1
                EXP1 = ATAIL
                GO TO 10
        ENDIF

        IF(SHAPC(1) .NE. 0)THEN
                NOC = 0                 !Shape constants on file.
                TSHP5 = SHAPC(5)
        ELSE
                NOC = 2
        ENDIF


10      BTAIL = TSHP5 * GAIN
        EXP2=BTAIL
        GAN = GAIN
        ZRO = ZERO

        IF (LONGPR)WRITE(LOUT,1005)KTGRP+1,EST,EEND
1005    FORMAT(///' Group',I3,' includes',F7.1,' to',F7.1,' keV')

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       NGAIN = 1 is a (byte) flag indicating fixed gain; otherwise, two
C  internal reference peaks are to be used to establish energy scaling.
C  If the first reference peak's energy = 0, this is interpreted to request
C  calculation of the energy of channel zero from ZERO & GAIN

        IF(NGAIN .GT. 0)GO TO 150

        IF((STD(1).EQ.OLSTD1) .AND. (STD(2) .EQ. OLSTD2))GO TO 150

        OLSTD1 = STD(1)
        OLSTD2 = STD(2)

        IF (LONGPR)WRITE(LOUT,1010)STD(1),STD(2)
1010    FORMAT(' Recalibrate Gain using energy standards at',F9.3,
     +  ' and',F9.3,' keV')
        GSQ = GAIN * GAIN

        DO 100 J=1,2
```

```
        TREFCH(J) = 0.0
        XC(J) = 0.
        IF(STD(J) .EQ. 0)GO TO 100

        IF(NOC .EQ. 0) THEN
                TFWHM = SQRT(SHAPC(1) + SHAPC(2) * STD(J))
        ELSE
                TFWHM = FWHM
        ENDIF

        IF(TFWHM .EQ. 0) THEN
                T = 0.3
                IF(GAIN .LT. 0.15) T = 0.03      !For low-noise detectors.
                TFWHM = SQRT(T +.002*STD(J)) !Estimate FWHM.
        ENDIF

        ALFA=-2.7726*GSQ/TFWHM**2
        HW = SQRT(.69315/-ALFA)
        XLIM = 20. * HW
        REF=(STD(J)- ZERO)/GAIN

C       Look for calibration peaks using the approx. GAIN & ZERO
        DO 90 K = 1,6
        IF(XLIM .GT. (MXDP-10))XLIM = MXDP-10
        NCH = XLIM + 0.5
        ST1 = REF - XLIM/2.0
        IST1 = ST1 + 0.5
        ST1 = IST1
        IF (LONGPR)WRITE(LOUT,1014)NCH,IST1,STD(J)
1014    FORMAT(' CALIB searching',I4,' chans from',I5,' for ',
     +  F7.1,' keV peak')

        CALL RDCHNS(Y,IST1,NCH,INPDEV)
15      NS = REF - ST1 - 2.0 * HW + 0.5
        IF(NS .LT. 1)NS = 1
        NE = REF - ST1 + 2.0 * HW + 0.5
        IF(NE .GT. NCH)GO TO 90
        IF(K .GT. 1)TYPE 1015,NS+IST1-1,NE+IST1-1
1015    FORMAT(' Raise limits to chans:',I5,' and',I5)

C       Locate highest & lowest points of STD peaks.
17      CALL MAXVAL (NS,NE,JPK,YMAX,Y)
        IF((JPK .GT. NS) .AND. (JPK .LT. NE))GOTO 25
        HW = 1.5 * HW
        GO TO 15
25      CALL MINVAL (1,JPK,MS,BG1,Y)
        CALL MINVAL (JPK,NCH,ME,BG2,Y)
        YTST = YMAX * 0.9
        IF((JPK .LE. MS) .OR. (JPK .GE. ME))GO TO 80
        IF((YTST .GT. BG1) .AND. (YTST .GT. BG2))GO TO 35
        GO TO 80

35      CALL AVE (MS,ME,SUMY,AV,Y)
        YS = 0.0
        DBG = BG2 - BG1
        DO 40 I = MS,ME
        YS = YS + Y(I)
        Y(I) = Y(I) - BG1 - DBG * YS/SUMY
        IF(Y(I) .LE. 0)Y(I) = 0.1
40      CONTINUE
```

```
        YTST = Y(JPK)*0.5
        M = 0
        DO 45, I = JPK+1,ME
        M = M + 1
        IF(Y(JPK+M) .LE. YTST)GO TO 50
45      CONTINUE
        GO TO 80

50      IF((JPK-M) .LT. MS)GO TO 80

55      IF(NUSHPS .NE. 0)GOTO 60
        SG = SHAPC(1)/GSQ + SHAPC(2) *(ST1/GAIN +ZERO/GSQ)+.462
        ALFA=-2.7726/SG
        EXP1=EXP(SHAPC(3) +STD(J) *SHAPC(4))

C       DETERMINE PEAK POSITIONS OF REFERENCE PEAKS

60      CALL GFIT(JPK,M,ALFA,EXP1,EXP2,PKPO,PHT,ER,Y)
        IF(ER .EQ. -2.)GO TO 78

        TREFCH(J) =  PKPO + ST1 - 1.
        CALL LINCOR(CLIN,TREFCH(J),XC(J))

        GO TO 100

78      TYPE 1078,ER
1078    FORMAT(' Error in GFIT; flag =',F4.0)

80      TYPE 1080,(JPK+IST1-1),NCH,IST1
1080    FORMAT(' Reject peak at chan',I5,' in',I4,' chan grp',
     +  ' beginning at',I5)
        XLIM = XLIM * 1.5       !Widen search limits & retry.
90      CONTINUE

        TYPE 1090,KTGRP+1
        IF(IOFLG(3) .EQ. 0)WRITE(LP,1090),KTGRP+1
1090    FORMAT(' CALIBRATION FAILURE -- ON Group #',I3,
     +  '; will use the preexisting GAIN')
        GO TO 150

100     CONTINUE

        TMP1 = STD(1)
        IF(TMP1 .EQ. 0) TMP1 = ZERO     !Use E of chan 0.
        GAIN = (STD(2) - TMP1) / (TREFCH(2) - TREFCH(1))
        GAN = (STD(2) - TMP1) / (XC(2) - XC(1))
        IF(STD(1) .NE. 0) THEN
                ZERO = STD(1) - GAIN * TREFCH(1)
                ZRO = STD(1) - GAN * XC(1)
                ZROC = TREFCH(1) - STD(1)/GAIN
        ENDIF
        IF ((ABS(ZROC) .GT. 5.) .AND. (ABS(GAIN-0.25).GT. .0025))THEN
        WRITE (LOUT,1140)
1140    FORMAT(' ***WARNING: Zero-Channel intercept or gain incorrect.
     +  Please center 152.7-keV peak at 612 and the 766.4-keV peak at
     +  channel 3063.  Check stabilizers')
        DGFLG(2) = .true.
        ABORT = .TRUE.
        END IF
```

```
      IF(INTFLG .NE. 0)INTFLG = INTFLG + 1     !Flag the new GAIN

150   ST = (EST - ZERO) / GAIN
      END = (EEND - ZERO) / GAIN
      IST = ST + 0.5
      IEND = END + 0.5
C     Correct for system nonlinearity by correcting the reference
C     energy value
      REFCH = (IST + IEND) / 2.
      CALL LINCOR(CLIN,REFCH,PP)
      REFEN = ZRO + GAN * PP
      IF (LONGPR)WRITE(LOUT,1155) IST,EST,EEND,ST,REFEN,ZERO,GAIN
1155  FORMAT(' St-Ch  Energy   End energy   start    Refen    Zero',
     +  '    Gain'/I6,6F10.4)
C     TYPE 9999, IST,EST,EEND,ST,REFEN,ZERO,GAIN
C9999 FORMAT (I5,6F10.4)

      RETURN

      END

C     -       -       -       -       -       -       -       -
C     CORRECT FOR ANY EFFECT OF SYSTEM NONLINEARITY ON ENERGY
C     MEASUREMENT (CLIN(I) VALUES =0 FOR LINEAR RESPONSE).

      SUBROUTINE LINCOR(CLIN,CH,X)

      DIMENSION CLIN(2)

      PP = CH
      PKP = PP
      DO 10, I=1,4
      PKP = PKP * CH
10    PP = PP + CLIN(I) * PKP
      X = PP
      RETURN
      END
```

```
C*****************************************************************
C CHK90
C
C       Routine is instrument specific for gamma ray instrument
C       using Series 90
C       Check error return on Series 90 call
C
C       CALLING ARGUMENTS:
C            ROUTIN = Routine from which the error was returned
C            IERR = Error number
C
C       SSJ 01-NOV-79
C       REV. 14-May-84 lc
C
C*****************************************************************
C
        SUBROUTINE CHK90 (ROUTIN,IERR)
C
        REAL*8 ROUTIN
C
C
        IF(IERR .GE. 0) GO TO 100
        TYPE 1000,IERR,ROUTIN
 1000   FORMAT(//20X,'**** Error number ',I3,' returned from ',A6//)
C
  100   RETURN
        END
```

```
C*****************************************************************
C CMPLTL        (CMPLTN subroutine for Livermore use)
C
C       Estimate completion time for the assay,
C       given the current time & the preset time
C
C       SSJ  25-OCT-82
C       Rev. 20-Jun-84 lc
C       REV. 26-OCT-84 MJC  Split from ACQR90
C                           Deleted argument to CALL DATTIM
C       REV. 13-DEC-84 WDR  Modified to determine start time of
C                           spectrum and pass results using CTIM.CMN
C       REV. 03-JAN-85 MPK  Restored DATTIM option argument
C       Rev. 02-APR-86 WDR  Modified for Micro VAX II
C
C*****************************************************************
C
      SUBROUTINE CMPLTN
C
      INTEGER*2 HOURS,MINS
      LOGICAL*1 ENDTIM(5)
      CHARACTER*8 WHEN
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CMONTH.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CTIM.CMN'
C
C  Get current date & time
      CALL DATTIM(1)
C
C  Get start time in Julian days and seconds since midnite
C
      CALL JULDAT(NOWDAT,NOWTIM,SDAT,STIM)
C
C  Get seconds since midnight
C
      SEC = SECNDS(0.0)
C
C  Add preset time in seconds
C  Take modulo seconds in a day
C
      SEC = SEC + TIMEPR
      DAY = 24. * 3600.
      NDAYS = SEC / DAY
      SEC = AMOD(SEC,DAY)
C
C  Compute hours & minutes
C
      HOURS = SEC / 3600.
      MINS = AMOD(SEC,3600.) / 60.
C
C  Encode time & zero-fill spaces
C
      ENCODE(5,1000,ENDTIM) HOURS,MINS
 1000 FORMAT(I2,':',I2)
      IF(ENDTIM(1) .EQ. ' ') ENDTIM(1) = '0'
      IF(ENDTIM(4) .EQ. ' ') ENDTIM(4) = '0'
```

```
C
C  Determine when - today, tomorrow, or how many days
C
        IF(NDAYS .EQ. 0) WHEN = '   Today'
        IF(NDAYS .EQ. 1) WHEN = 'Tomorrow'
        IF(NDAYS .GT. 1) WHEN = '    days'
        IF(NDAYS .GT. 1) ENCODE(2,1004,WHEN) NDAYS
 1004   FORMAT(I2)
C
        TYPE 1005, TIMEPR,NOWTIM,NOWDAT,WHEN,ENDTIM
 1005   FORMAT(/' Starting a run of ',F7.0,' s  at ',A,' ('A,')',
     X     3X,'Ends ',A,' at ',5A1/)
C
        RETURN
        END
```

```
C
C*****************************************************************
C
C  CURDTA
C
      Responds to the operator request for assay from
C     current data in MCA
C
C     CMS 10-MAY-84
C     REV.11-SEP-84 CMS
C     Rev. 08-Jan-85  MPK     remove reset of AC flag
C                             (now done by INTFLG)
C     Rev. 02-Apr-86  WDR     Modified for Micro VAX II
C
C*****************************************************************
C
      SUBROUTINE CURDTA
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'CBACKG.CMN'
C
C  The following common block files are instrument specific for
C  gamma ray and x-ray multichannel instruments
C
      INCLUDE 'CSPECT.CMN'

      RTYP = 'From MCA'
      AC = .TRUE.
      CALL DIALOG
      IF(ABORT)GO TO 500
      CALL ASSAY
      IF(ABORT) GO TO 500
      CALL PRINT1                        !instrument specific routine
500   RETURN
      END
```

```
C*****************************************************************
C DATA90
C       Return Region of Interest data from the Series 90
C
C       Calling arguments:
        NFIRST - First channel to retrieve
C       NLAST  - Last channel to retrieve
C       SUM    - Returns the sum of the channels
C
C
C       SSJ 03-JUL-79
C       REV. 10-May-84 LC
C       REV. 26-OCT-84 MJC  CHANGED LAST TO MEMSIZ FROM 4095
C       REV. 02-APR-86 WDR  Modified for Micro VAX II
C       REV. 05-JUL-87 RDP  Modified for uVAX II w/ ADAC Drivers
C
C*****************************************************************
C
        SUBROUTINE DATA90 (NFIRST,NLAST,SUM)
C
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
C
C  The following common block files are instrument specific for
C  (gamma ray) instruments
C
        INCLUDE 'CSPECT.CMN'
        INCLUDE 'CONT90.CMN'
C
C
C
C  Check the validity of the input channel numbers
C
        IF(NFIRST .LT. 0 .OR. NLAST .GE. 4095) GO TO 500
        NUMCHN = NLAST - NFIRST + 1
        IF(NUMCHN .GT. 128) GO TO 500
C
C  Check if it is necessary to get new data or whether the data
C  exists in /cspect/ already
C
        IF(NFIRST .GE. IFIRST .AND. NLAST .LE. IFIRST+255) GO TO 50
C
        IF(FROMDK) GO TO 20
C
C  Get the data from the Series 90
C  Get 256 channels
C
        LAST = NFIRST+127
        IF(LAST .GT. MEMSIZ) LAST = MEMSIZ
C       TYPE 1200,NFIRST,LAST
C1200   FORMAT(' Debug - FROMCA',215)
        CALL FROMCA(IADC,LAST,NFIRST,SPCTRM,IERR,)
        CALL CHK90('FROMCA',IERR)
        IFIRST = NFIRST
        GO TO 50
C
```

```
C  Get data directly from a disk file
C  Check which records the first and last channels are in
C
 20     IRECF = NFIRST/128 + 1
        IRECL = NLAST/128 + 1
        IF(IRECF .LT. IRECL) GO TO 25
C
C  First and last channels are in the same record
C  Read the whole record
C
        READ(DKLUN'IRECF) SPCTRM
        IFIRST = (IRECF-1) * 128
        GO TO 50
C
C  Get half the data from the first record
C  and half from the second
C
 25     READ(DKLUN'IRECF) (DUM,I=1,128),(SPCTRM(I),I=1,128)
        READ(DKLUN'IRECL) (SPCTRM(I),I=129,256)
        IFIRST = (IRECF-1) * 256 + 128
C
C  Sum the channels asked for
C
 50     J = NFIRST - IFIRST + 1
        SUM = 0.0
        DO 60 I=1,NUMCHN
          SUM = SUM + SPCTRM(J)
 60       J = J + 1
        GO TO 600
C
C  Error in input parameters

 500    TYPE 1000,IADC,NFIRST,NLAST
 1000   FORMAT(//20X'**** DATA90 INPUT ERROR - ADC#',I4/
     X          '   NFIRST = ',I5,'    NLAST = ',I5,' ****'/)
C
 600    CONTINUE
D       WRITE(LOUT,1600) NFIRST,NLAST,SUM
D1600   FORMAT(2I6,F10.0)
        RETURN
        END
```

```
C*****************************************************************
C DATTIM
C
C       Retrieve, verify or change the system date & time
C
C       SSJ 25-JAN-79
C       REV. 13-SEP-83 LC
C       Rev 02-Jan-85 MPK       Include option to sync to date and
C                                time received from VAX and stored in
C                                the CVAX common. Adapt to the NSR/FMF
C                                generic program. Add automatic operator
C                                input of date and time if date and time
C                                are found to be unreasonable.
C       Rev 14-Jul-87 RDP       Modified for uVAX II
C       Rev 18-Sep-87 RDP       Remove Operator from initializing date
C                               and time.(IOPT=0)
C
C       Options:
C
C               0 - Get initial date and time from operator input
C                   and give to monitor.
C               1 - Update date and time from monitor.
C               2 - Update and let operator verify or change.
C               3 - Sync to VAX date and time (note that these
C                   must have been set up by a call to VAX
C                   communication immediately before calling
C                   DATTIM).
C
C*****************************************************************
C
        SUBROUTINE DATTIM(OPTION)

        INTEGER OPTION
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CMONTH.CMN'
        INCLUDE 'CVAX.CMN'
C
        CHARACTER*9 INPSTR
        CHARACTER*23 DATE_TIME
C
        LOGICAL*1 VFD
C
        INTEGER*4 STATUS, SYSTIM(2), SYS$BINTIM
C
        INTEGER*2 SYS$SETIME
C
        EXTERNAL SYS$BINTIM, SYS$SETIME
C
        DATA AMONTH /'JAN','FEB','MAR','APR','MAY','JUN',
        1             'JUL','AUG','SEP','OCT','NOV','DEC'/
        DATA ALOWDT /0./          !Julian date for 01-Jan-1900
        DATA AHIDAT /36525./      !Julian date for 31-Dec-99
C
C  Set internal option variable.  If it is zero, force operator
C  input.  If it is 3, sync to date and time from VAX.
C
        IOPT=OPTION
```

```
        IF (IOPT.LE.0) GOTO 100
        IF (IOPT.GE.3) GOTO 300
        VFD=.FALSE.
C
C  Update alphanumeric date and time.
C
  .0    CALL DATE (NOWDAT)
        CALL TIME (NOWTIM)
C
C  If the date and time have been operator verified, return now.
C
        IF (VFD) GOTO 500
C
C  For option 1, return if date and time from VAX are reasonable;
C  otherwise, make the operator input them by setting the internal
C  option to 0.
C
        IF (IOPT.NE.1) GOTO 40
        CALL JULDAT (NOWDAT,NOWTIM,ADAT,ATIM)
        VFD=.TRUE.
        IF ((ADAT.GE.ALOWDT).AND.(ADAT.LE.AHIDAT).AND.
     1      (ATIM.GE.0.0).AND.(ATIM.LE.86399.0)) GOTO 500
        IOPT=0
        GOTO 100
 40     CONTINUE
C
C  Print out date and time for operator verification.
C
        TYPE 1000,NOWDAT,NOWTIM
 1000   FORMAT(/1X,A,2X,A)
        VFD=.TRUE.

C  Allow operator to change date.
C
 100    WRITE(*,1003)
 1003   FORMAT(' '/'$Enter Date in the form 01-FEB-85    ->')
        IF (IOPT.NE.0) WRITE(*,1006)
 1006   FORMAT('$ (or RETURN for no change)')
        READ(5,1014,END=100,ERR=100) INPSTR
C!      READ(5,1014,END=100,ERR=100) NCHAR,INPSTR
C! 1014 FORMAT(Q,A)
 1014   FORMAT(A)
        IF ((IOPT.NE.0).AND.(INPSTR.EQ.' ')) GOTO 200
        IF (INPSTR(3:3).NE.'-') GOTO 100
C!      IF (NCHAR.NE.0) GOTO 100
        IF (IOPT.EQ.0) GOTO 110
        GOTO 200
C
C  We appear to have a date from the operator.  Call JULDAT to
C  parse it, and check its components for reasonableness.
C
 110    VFD=.FALSE.
        CALL JULDAT(INPSTR,NOWTIM,ADAT,ATIM)
        IF (IYEAR.GE.84 .AND. IYEAR.LE.99) GOTO 120
        TYPE 1024
 1024   FORMAT (//20X,'**** INCORRECT YEAR NUMBER ****'/)
        GOTO 100
 120    IF (IMONTH.GE.1 .AND. IMONTH.LE.12) GOTO 130
        TYPE 1025
 1025   FORMAT (//20X,'**** INCORRECT MONTH NAME ****'/)
```

```
        GOTO 100
 130    IF (IDAY.GE.1 .AND. IDAY.LE.31) GOTO 170
        TYPE 1020
 1020   FORMAT (//20X,'**** INCORRECT DAY NUMBER ****'/)
        GOTO 100
  170   DATE_TIME(1:8)=INPSTR(1:8)
        J = INDEX(DATE_TIME,'-')+1
        IF (J.GT.3) GOTO 175
        TYPE 8000
8000    FORMAT(//20X,' **** INCORRECT FORMAT  04-APR-86 ****')
        GOTO 100
  175   J = INDEX(DATE_TIME(J+1:),'-')+1+J
        DATE_TIME(J:J)='1'
        DATE_TIME(J+1:J+1)='9'
        DATE_TIME(J+2:J+2)=INPSTR(J:J)
        DATE_TIME(J+3:J+3)=INPSTR(J+1:J+1)
C
C  Allow operator to change time.
C
 200    WRITE(*,1004)
 1004   FORMAT(' '/'$Enter Time in the form 01:47:59    ->')
        IF (IOPT.NE.0) WRITE(*,1006)
        READ(5,1014,END=200,ERR=200) INPSTR
        IF ((IOPT.NE.0).AND.(INPSTR.EQ.' ')) GOTO 500
        IF (INPSTR(3:3).NE.':') GOTO 200
        IF (IOPT.EQ.0) GOTO 210
        GOTO 300
C
C  We appear to have a time from the operator.  Call JULDAT to
C  parse it, and check its components for reasonableness.
C
  210   VFD=.FALSE.
        CALL JULDAT (NOWDAT,INPSTR,ADAT,ATIM)
        IF (IHOUR.GE.0 .AND. IHOUR.LE.23) GO TO 220
        TYPE 1034
 1034   FORMAT (//20X,'**** INCORRECT HOUR NUMBER ****'/)
        GO TO 200
 220    IF (IMIN.GE.0 .AND. IMIN.LE.59) GO TO 230
        TYPE 1036
 1036   FORMAT (//20X,'**** INCORRECT MINUTE NUMBER ****'/)
        GO TO 200
 230    IF (ISEC.GE.0 .AND. ISEC.LE.59) GO TO 270
        TYPE 1038
 1038   FORMAT (//20X,'**** INCORRECT SECOND NUMBER ****'/)
        GO TO 200
  270   CONTINUE
C
C  Load DATE_TIME character string with time and pad with blanks
        DATE_TIME(12:12)=' '
        DATE_TIME(13:20)=INPSTR(1:8)
        DATE_TIME(21:23)='   '
C       TYPE 9000,DATE_TIME
9000    FORMAT(1X,23A)
        STATUS = SYS$BINTIM(DATE_TIME,SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
        STATUS = SYS$SETIME(SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
C
C  Repeat update and verification.
C
```

```
        IOPT=2
        GOTO 10
C
C  Sync to the VAX time.
C
 )0     CALL JULDAT(VAXDAT,VAXTIM,ADAT,ATIM)
C. Get VAX time from comm
C!      CALL COMM('TS')
C  Load DATE_TIME character string with time and date (modified date
C   to meet Date String requirements.
        DATE_TIME(1:9)  = VAXDAT
        DATE_TIME(10:11)= DATE_TIME(8:9)
        DATE_TIME(8:9)  ='19'
        DATE_TIME(12:12)=' '
        DATE_TIME(13:20)= VAXTIM
        DATE_TIME(21:23)='   '
C       TYPE 9000,DATE_TIME
C Update Time
        STATUS = SYS$BINTIM(DATE_TIME,SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
        STATUS = SYS$SETIME(SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
 500    RETURN
C
C
        END
```

```
C*****************************************************************
C DIALGS        (DIALOG subroutine for solid isotopic system)
C
C       Assay and Background dialogue with operator for the solid
C       isotopic instrument
C
C       ***This routine needs to be modified for the communications
C            package ***
C
C       SSJ 17-AUG-79
C       REV. 31-May-84 lc
C       REV. 05-Sept-84 TES  10:28
C       REV. 01-NOV-84 WDR  CREATE FILENAME FOR CURRENT DATA
C       REV. 11-OCT-84  MJC: CHANGED ACCEPT TO READ WITH ERR TO BRANCH
C                            BACK TO 'ENTER NUMBER OF CYCLES/CALIBRATIONS'
C       REV. 29-NOV-84 MPK Replace list-directed with formatted I/O
C       REV. 19-DEC-84 WDR Modified for solid isotopic instrument
C       REV. 14-FEB-85 MPK      Rewrite to do "real" VAX comm
C                               Replace all ACCEPTS with READS having
C                               END and ERR exits
C                               Use YESNO to ask yes/no questions
C       REV. 20-FEB-85 MPK      Check ABORT after calling GETSMP
C       Rev. 23-OCT-85 WDR      Modified communications with the VAX
C                               installed 'LO' message to VAX
C       Rev. 21-NOV-85 WDR      Get Cadmium absorber thickness when
C                               calibrating
C       Rev. 02-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 13-Jul-87 RDP      Modified for uVax II
C                               Convert BYTE Strings to CHARACTER strings
C       Rev. 13-Aug-87 RDP      Preliminary Hardware and Acquistion
C                               check added before performing ASSAY.
C       Rev. 02-May-90 WDR      Removed COMM call for Pu238 instrument
C
C*****************************************************************
C
        SUBROUTINE DIALOG
C
        LOGICAL*1 YESNO,MTL
        CHARACTER*15 CURDTA
        REAL*4 MONTH
        INTEGER SAMLEN
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CVAX.CMN'
        INCLUDE 'CLOG.CMN'
        INCLUDE 'CCAL.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CMCONS.CMN'
        INCLUDE 'CMC.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CMONTH.CMN'
        INCLUDE 'P238COM.CMN'
C
C  The following common block files are instrument specific for
```

```
C  gamma ray instruments
C
      INCLUDE 'CSPECT.CMN'
C
      DATA SAMLEN/9/
      DATA CURDTA/'CURDTA.   '/
C
C  Get operator and sample ID
C
      CALL GETOPS(MTL)
      IF(ABORT)RETURN
C
C  Get filename if from disk
C
    8 IF( .NOT.(AC .AND. WRITNG .OR. FROMDK))GO TO 85
    9 TYPE 1059
 1059 FORMAT(' '/'$Enter the full filename [ FILENM.EXT ] -> ')
      READ (5,1001,END=9,ERR=9)FILNAM
      IF(FILNAM(1:1) .EQ. ' ') GO TO 450
C
C  Get number of cycles for autocycling
C
 85   NUMCYC = 1
      ICYCLE = 1
      IF(.NOT. AUTOCY )GO TO 245
 86   TYPE 1020
 1020 FORMAT(/'$Enter number of cycles -> ' )
 1013 FORMAT(I)
      READ(5,1013,END=86,ERR=86)NUMCYC
      IF(NUMCYC .LT. 0) GO TO 400
      IF(NUMCYC .EQ. 0) GO TO 86
      IF(FROMDK)GO TO 500
C
C  Skip rest of code if no data is to be taken
C
 245  IF( FROMDK)GO TO 500
      IF(WRITNG .OR. AUD)GO TO 295
C
C      Create a dummy filename for current data
C      This is required for Livermore solution analysis code
C      Get start date for data
C
247   FILNAM = CURDTA
      CALL NEWNAM(0)
      GOTO 350
C
C  Set filename to sample id if writing or VAX is not up
C
295   IF(.NOT. WRITNG .AND. VAXUP)GO TO 350
      IF (AC) GO TO 500
      K = 1
      DO 300 I = 1, SAMLEN
      FILNAM(K:K) = SAMPID(I:I)
      K = K + 1
 300  CONTINUE
      CALL NEWNAM(0)

C  Final instructions for inserting the sample
C
 350  IF(AC .OR. FROMDK) GO TO 500
```

```
C Perform Preliminary Check on Acquistion System
        CALL PRECHK
        IF (ABORT) GO TO 400
        IF (BACKG) THEN
          TYPE 1057
 1057     FORMAT(/'$Remove any sample, then press RETURN  -> ')
          ACCEPT 1001
 1001     FORMAT(80A)
        ENDIF
        CALL ADATE
        GO TO 500
C
C  Abort the measurement because of negative input
C
 400    TYPE 1019
 1019   FORMAT(/' Measurement aborted by operator input')
C
 450    ABORT = .TRUE.
C
 500    RETURN
        END
```

```
      INTEGER*4 FUNCTION ERR_HANDLER(SIGARGS, MECHARGS)
C
C Error Handler for Math Errors in Data analysis routines
C ABORT flag is set and a SYS$UNWIND is preformed.
C
      INTEGER*4       SIGARGS(*), MECHARGS(5)
      INCLUDE         '($MTHDEF)'
      INCLUDE         '($SSDEF)'
      INCLUDE         'CONTRL.CMN'
C
C Assume not Math Error...
      ERR_HANDLER = SS$_RESIGNAL
C
      IF (LIB$MATCH_COND(SIGARGS(2),SS$_FLTOVF,
     c                                SS$_FLTOVF_F,
     c                                SS$_FLTDIV,
     c                                SS$_FLTDIV_F,
     c                                SS$_INTDIV,
     c                                SS$_INTOVF).NE.0) THEN
         ABORT = .TRUE.
         WRITE(*,*)
         WRITE(*,*) 'ERROR -- Divide by Zero Encountered'
         WRITE(*,*) 'Analysis Invalid... <ABORT>'
         WRITE(*,*)
         CALL SYS$UNWIND(,)
      ELSE IF (LIB$MATCH_COND(SIGARGS(2),MTH$_SQUROONEG).NE.0) THEN
         ABORT = .TRUE.
         WRITE(*,*)
         WRITE(*,*) 'ERROR -- Square Root of a Negative Number'
         WRITE(*,*) 'Analysis Invalid... <ABORT>'
         WRITE(*,*)
         CALL SYS$UNWIND(,)
      ELSE IF (LIB$MATCH_COND(SIGARGS(2),MTH$_FLOOVEMAT).NE.0) THEN
         ABORT = .TRUE.
         WRITE(*,*)
         WRITE(*,*) 'ERROR -- OverFlow in Math Function'
         WRITE(*,*) 'Analysis Invalid... <ABORT>'
         WRITE(*,*)
         CALL SYS$UNWIND(,)
      ELSE IF (LIB$MATCH_COND(SIGARGS(2),MTH$_LOGZERNEG).NE.0) THEN
         ABORT = .TRUE.
         WRITE(*,*)
         WRITE(*,*) 'ERROR -- Zero or Negative Argument of Logarithm'
         WRITE(*,*) 'Analysis Invalid... <ABORT>'
         WRITE(*,*)
         CALL SYS$UNWIND(,)
      ENDIF
      RETURN
      END
```

```
C*********************************************************************
C  ERRMSG
C
C       Read appropriate error message from the
C       error message file and output to user terminal
        Calling arguments:
C               INUM = Message calling argument
C               ARG = ASCII argument passed to be printed (15A1)
C       File format:
C               INUM (Message call number) = I4
C               IARG (ASCII argument passed) = A1
C               MESSAG (ASCII error message) = 72A1
C
C       21-Apr-84 LC
C       REV. 12-OCT-84 MJC:     CLOSED DKLUN1 IF ARG =' '
C       Rev. 19-Apr-85  MPK     Remove redundant initialization
C                               of DKLUN and DKLUN1
C       Rev. 29-Apr-85  MPK     Put MESSAG and ICHAR in CMSG common
C       Rev. 02-Apr-86  WDR     Modified for Micro VAX II
C       Rev. 13-Jul-87  RDP     Modified for uVax II
C                               Convert BYTE strings to CHARACTER strings
C
C*********************************************************************
C
C
        SUBROUTINE ERRMSG(INUM,ARG)
C
C
        CHARACTER ARG*15,IARG*1
C
C  The following common block files are generic for all instruments

        INCLUDE 'CFILES.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CMSG.CMN'
C
C
C  Open error message file
C  Locate proper error message
C
        OPEN (UNIT=DKLUN1,NAME=ERRFIL,TYPE='OLD',ERR=25)
 10     READ(DKLUN1,1000,ERR=20,END=20)NUM,IARG,ICHAR,
     X           (MESSAG(I),I=1,ICHAR)
 1000   FORMAT(I4,A1,Q,72A1)
        IF (NUM .EQ. -999)GO TO 20
 11     IF (NUM .EQ. INUM) GO TO 15
        GO TO 10
C
C  Print error message
C
 15     TYPE 1015, NUM, (MESSAG(I),I=1,ICHAR)
 1015   FORMAT(/5X,'**** ERROR # ',I3/5X,72A1,' ****')
        IF(IARG .NE. ' ')GO TO 17
        CLOSE(UNIT=DKLUN1)
        GO TO 30
 17     TYPE 1016, ARG
 1016   FORMAT(7X,A15)
        CLOSE(UNIT=DKLUN1)
        GO TO 30
C
```

```
C  Error number not found in file
C
 20     TYPE 1020, INUM, ERRFIL
 1020   FORMAT(/5X,' Error # ',I4,' not located in ',A15)
        CLOSE(UNIT=DKLUN1)
        GO TO 30
C
C  Error opening file
C
 25     TYPE 1025, ERRFIL
 1025   FORMAT(/5X,'**** UNABLE TO OPEN ',A15,' ****')
C
C
  30    RETURN
        END
```

```
C
C*********************************************************************
C
C  EXEC238      (EXEC routine for Pu238 isotopic system)
C
C       Executive main program to perform the assay measurements for
C       the NDA instruments under the FMF/NSR projects for the
C       Savannah River Plant.
C       The program initializes the hardware for the NDA instrument
C       and checks the status of the hardware.  It also accepts
C       operator input for the menu selection and executes the
C       subroutine for the option selected.
C
C       CMS 25-APR-84
C
C       REV. 2-OCT-84    CMS     Deleted form feeds to line printer
C                                before closing
C                                Changed 'ASLOGF' to 'ASYLOG'
C       REV.11-OCT-84    MJC     Added counter IENTNR near label 1008 to
C                                track number of entries in ASYLOG
C       REV. 29-NOV-84   MPK     Replace list-directed with formatted I/O
C       REV. 17-DEC-84   WDR     Modified for solid isotopic system
C       REV. 03-JAN-85   MPK     Restore DATTIM option argument
C       REV. 15-JAN-85   WDR     Changed from main program to subroutine
C       REV. 18-JAN-85   WDR     Modified option input to test LN monitors
C       Rev 22-Jan-85    MPK     Send log entries to COMM using
C                                pseudo-message-type 'LA'
C       Rev 25-Jan-85    MPK     Add end and err exits to term input
C                                Reset ctrl/O before option prompt
C       REV. 25-JAN-85   WDR     Changed from main program to subroutine
C       REV. 25-JAN-85   WDR     Modified option input to test LN monitor
C       Rev 30-Jan-85    MPK     Handle sending files to VAX by
C                                calling LSLOG in auto mode
C       Rev 31-Jan-85    MPK     Add argument to call to LMCLOG
C                                (this "hook" for automatic sending
C                                of MC files to the VAX is not actually
C                                used at this time)
C       Rev 08-Jan-85    MPK     Redo management of logical flags
C       Rev 06-Mar-85    MPK     Put VAX comm init and time sync in
C       Rev 07-Mar-85    MPK     Remove VAX init (now done in RESET)
C       Rev 19-Apr-85    MPK     Remove dead code
C       Rev 02-Apr-86    WDR     Modified for Micro VAX II; made main
C                                program
C       Rev 05-Jul-87    RDP     Modified for uVAX II w/ ADAC Drivers
C                                Convert PROGRAM to SUBROUTINE
C                                Convert BYTE strings to CHARACTER strings
C       Rev 30-Aug-89    WDR     Modified for Pu238 isotopic instrument
C       Rev 02-May-90    WDR     Removed CALIB and COMM for Pu238 instrument
C       Rev 07-May-91    WDR     Modified for Hanford 238 instrument
C       Rev 04-Jun-91    WMB     added support for stabilizer control
C*********************************************************************
C
      SUBROUTINE EXEC
C-----------------------------------------
C*********************************************************************
C Routine loops until a input for the keyboard aborts the process

      COMMON/RTI/       INKEY,INCHAR
      INTEGER           INKEY
      CHARACTER         INCHAR*1
```

```
      INTEGER*4       ISTATUS,VKID,PBID
C
C System Routines
      INTEGER*4       SMG$CREATE_PASTEBOARD
      INTEGER*4       SMG$CREATE_VIRTUAL_KEYBOARD
      INTEGER*4       SMG$ENABLE_UNSOLICITED_INPUT
      INTEGER*4       SMG$DISABLE_UNSOLICITED_INPUT
      INTEGER*4       SMG$DELETE_VIRTUAL_KEYBOARD
      INTEGER*4       SMG$DELETE_PASTEBOARD
C
      EXTERNAL        GET_KEY
C
      INCLUDE         '($SYSSRVNAM)'
C
C-----------------------------------------------
      LOGICAL * 1 NOPT,SUPV,AUTSND
      CHARACTER   OPT(25)*4, OPTIN*4,OPTR*3
      CHARACTER   SAVFIL*15, ACHAR*1
      INTEGER OPTLEN,MENPRM,ON6,OFF6
      EQUIVALENCE (OPTR(1:1),OPTIN(2:2))
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'CLOG.CMN'
      INCLUDE 'CPASS.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CMONTH.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CMC.CMN'
      INCLUDE 'CBACKG.CMN'
      INCLUDE 'CONFIL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CMCONS.CMN'
      INCLUDE 'CCAL.CMN'
      INCLUDE 'p238com.cmn'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
      INCLUDE 'CSPECT.CMN'
      INCLUDE 'CONT90.CMN'
C
      DATA MAXASY,LOGREC /100,22/
      DATA JSW /"44/, ON6 /"100/, OFF6 /"177677/
      DATA MCA /1/, IADC /1/, MEMSIZ /4096/    !CHANGE FOR INSTRUMENT
      DATA MCMAX,MCREC /100,22/
      DATA DKLUN /1/, DKLUN1 /2/, LOUT /6/
      DATA MENPRM /25/, OPTLEN /4/
      DATA ASLOG  /'ASYLOG.WH8'/
      DATA CONFIL /'PARMTR.WH8  '/
      DATA BKFIL  /'BACKGR.WH8  '/
      DATA MCLOG  /'MCLOGF.WH8  '/
      DATA NAMSFL /'CONSTA.WH8  '/
      DATA ERRFIL /'ERRMSG.INS  '/
C-----------------------------------------------------
```

```
C ** Single Key Interupt for VAX input
C Create PasteBoard
      ISTATUS = SMG$CREATE_PASTEBOARD(PBID,'SYS$INPUT')
      IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C
C Create Virtual Keyboard
      ISTATUS = SMG$CREATE_VIRTUAL_KEYBOARD(VKID,'SYS$INPUT')
      IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C
C Enable Virtual Input
      ISTATUS = SMG$ENABLE_UNSOLICITED_INPUT(PBID,GET_KEY,VKID)
      IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C
C-----------------------------------------------------
C  Announce startup
C
      TYPE *,' Starting WHC_ASY238 program Version 1.02'
C
C  Initialize logical flags and hardware for the instrument
C  If problem with the hardware -  abort flag is set
C
      CALL START
      IF (ABORT) GOTO 500
C
C  Reset default settings for the Assay options
C  Argument 0 resets default flags and parameters
C  Argument 1 permits operator to select flags and parameters
C
      CALL RESET(0)
C
C  set values for 8232 stabilizer

      CALL STABSET
C
C  Get time sync from VAX and set date and time
C
C! Get sync time from VAX and set time and date.
C! If VAX is down test if time and date are reasonable.
C! if not, get time and date from user.
      IOPT = 3
C     CALL COMM('TS')
      IF (.NOT.VAXUP) GO TO 15
      CALL DATTIM(IOPT)
C
C  Initialize menu option list
C
   15 DO 20 I = 1, MENPRM
         OPT(I) = ' '
   20 CONTINUE
C
C  Close line printer if in use
C
   30 IF (LOUT.NE.6) GOTO 100
      CLOSE(UNIT=LOUT,ERR=100)
C
C  Open a non-existent file on the system disk to clear
   the directory, in case floppies are switched by the
C  operator;  applies to noswap
C
  100 OPEN(UNIT = DKLUN, NAME = 'NOFILE.DAT', TYPE = 'OLD',
```

```
     1        ERR = 105)
         CLOSE(UNIT=DKLUN)
  105    CONTINUE
C
C  Check status of hardware - determines if hardware is okay
C    to run
C
C
  110    ABORT = .FALSE.
         CALL STATUS(0)            !instrument specific routine
         IF (ABORT) GOTO 500
C
C  Re-initialize logical flags
C
         CALL INTFLG
C
C  Get system date/time
C
  120    CALL DATTIM(1)
C
C  Prompt operator for menu option
C  Menu options are:
C  202  A    -  Assay
C  206  MB   -  Measurement control - Bias
C  208  MP   -  Measurement control - Precision
C  210  H    -  Help - display menu
C  212  AC   -  Assay Current data in MCA
C  214  AD   -  Assay data from Disk
C  216  AU   -  AUtocycle
C  218  AUD  -  AUtocycle from Disk
C  222  C    -  Calibration
C  224  CD   -  Calibration from Disk
C  226  D    -  Default
C  228  LA   -  List Assay log
C  230  LM   -  List Measurement control log
C  232  ST   -  Status
C  234  R    -  Read data from disk
C  236  W    -  Write data to disk
C  238  OU   -  change OUtput listing device
C  240  HS   -  Help Supervisor mode
C  500  X    -  eXit
C
         TYPE 1003, NOWDAT, NOWTIM
 1003    FORMAT(' '/,1X,A9,2X,A8,3X,'Enter OPTION ( H to see Menu) ->  ',$)
C
C  Check LN monitors while waiting for operator input.
C  Read operator input - if error in input, output a
C  message and prompt again
C
C
125      CALL LNMN38
         IF (ABORT) GOTO 120
C
C-----------------------------------------------------
C
C Test if Key has been press...
         IF (INKEY.NE.0) THEN
            ACHAR = INCHAR         ! Get Character
            INKEY = 0              ! Reset INKEY Flag
            TYPE 2001,ACHAR        ! Echo Character
```

```
 2001       FORMAT('+',A,$)
          ELSE
             CALL SLEEP(0.5)
             GOTO 125                ! Return to loop
          ENDIF

C-------------------------------------------------------
          READ (5,1004,END=120,ERR=120) OPTR
  1004    FORMAT(A3)
          OPTIN(1:1) = ACHAR
          IF (OPTIN(1:1).EQ.' ') GOTO 120
C
C  Menu searches a table of options available and matches the
C  input to a menu option in the table - if there is not a
C  match, a flag is set and the operator is prompted again.
C  When a match is found menu returns a number of the routine
C  to be called.
C
          CALL MENU(OPTIN,MNUM,NOPT)
          IF (ABORT) GOTO 130
          IF (NOPT)  GOTO 120
          GOTO 140
    130   CALL ERRMSG(102)
          ABORT = .FALSE.
          GOTO 110
    140   IF (MNUM .GT. 5) GO TO 211
          GOTO (202,204,206,208,210) MNUM
C
C  The following is a list of the subroutines that can
C  be executed from the menu option list - these menu options
C  do not require a password to access them

    202   CALL ASOPT
           GO TO 260
C
    204   CONTINUE          !Background option removed
C
    206   CALL MCBIAS
           GO TO 260
C
    208   CALL MCPREC
           GO TO 260
C
    210   SUPV=.FALSE.
          CALL HELP(SUPV)
           GO TO 260
C
C  These menu options require a password
C
    211   CALL PASS(OK)
          IF(.NOT. OK) GO TO 110
          MPNUM = MNUM - 5
          GO TO (212,214,216,218,220,222,224,226,228,230,
       1          232,234,234,238,240,500) MPNUM
C
    212   CALL CURDTA
           GO TO 260
C
    214   AD = .TRUE.
           GO TO 2201
```

```
C
  218   AUD = .TRUE.
        FROMDK = .TRUE.
C
  216   CALL AUTCYC
         GO TO 260
C
  220   CONTINUE          !Background from disk removed
        GO TO 260
 2201   CALL FRDSK
         GO TO 260
C
  224   CALDSK = .TRUE.
        FROMDK = .TRUE.
C
  222   CONTINUE          !CALIBRATION OPTION REMOVED
*C  222  CALL CALIBS
         GO TO 260
C
  226   CALL RESET(1)
         GO TO 260
C
  228   AUTSND=.FALSE.
        CALL LSLOG(AUTSND)
         GO TO 260
C
  230   AUTSND=.FALSE.
        CALL LMCLOG(AUTSND)
          GO TO 260
C
  232   CALL STATUS(1)           !instrument specific routine
         GO TO 260
C
C  Prompt for filename for reading and writing data to/from disk
C
  234   SAVFIL=FILNAM
  231   TYPE 2005
 2005   FORMAT(/'$Enter full filename [FILENM.EXT] -> ')
        READ (5,2006,END=231,ERR=231) FILNAM
 2006   FORMAT(A15)
        IF(FILNAM(1:1) .EQ. ' ') GO TO 260
        IF (OPTIN(1:1).EQ.'W') GOTO 236
        CALL PUT90(IADC,0)              !instrument specific routine
        GO TO 237
  236   CALL GET90(IADC)                !instrument specific routine
  237   FILNAM=SAVFIL
C       IADC = IADCSV
         GO TO 260
*C
  238   TYPE 2007
 2007   FORMAT(' '/'$Enter 6 for line printer, 5 for terminal -> ')
        READ (5,2008,END=238,ERR=238) LOUT
 2008   FORMAT(I)
        IF(LOUT .EQ. 0) LOUT = 5
        IF(LOUT .NE. 6 .AND. LOUT .NE. 5) GO TO 238
        GO TO 110

C
  240   SUPV = .TRUE.
        CALL HELP(SUPV)
```

```
        GO TO 110
C
  260   IF(ABORT) GO TO 30
C
C  If the Vax is up, send all files that have not yet been sent
C
        IF(.NOT. VAXUP) GO TO 30
        TYPE 2010
2010    FORMAT(/' Files being sent to VAX ')
        AUTSND =.TRUE.
        CALL LSLOG(AUTSND)
C
C
        GO TO 110
  500   CONTINUE
C
C   Exit from program
C
C       CALL COMM('OS')
C-------------------------------------------------------
C *** Remove Single Key Input
C Disable Unsolicited Virtual Keyboard Input
        ISTATUS = SMG$DISABLE_UNSOLICITED_INPUT(PBID)
        IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C
C Delete Virtual Keyboard
        ISTATUS = SMG$DELETE_VIRTUAL_KEYBOARD(VKID)
        IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C
C Delete Pasteboard
        ISTATUS = SMG$DELETE_PASTEBOARD(PBID)
        IF (.NOT.ISTATUS) CALL LIB$SIGNAL(%VAL(ISTATUS))
C-------------------------------------------------------
        RETURN
        END
```

```
      SUBROUTINE FIT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     FIT modified to use MATINV and linear peak heights.
C             Last edit (R.G.) Statement 464 correction
              Last edit for VAX 13-OCT-87
L             Modified for Pu238 instrument WDR 1-Sep-89
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPRFL.PMS'
C  The following common is required for the instrument

      INCLUDE 'CONTRL.CMN'
C

      BYTE NFLG1,NFLG2,NFLG3,NFLG4,NFLG5
      EQUIVALENCE (NFLG1,NFLG(1)),(NFLG2,NFLG(2))
      EQUIVALENCE (NFLG3,NFLG(3)),(NFLG4,NFLG(4)),(NFLG5,NFLG(5))
      EQUIVALENCE (KK,NWRDS(2))        !A fudge to transfer KK to OUT.


      DIMENSION AX(MXFP)
      EQUIVALENCE (AX,VM)
      DOUBLE PRECISION APHA(MXFP,MXFP),XSUM(MXFP)

      IF(MAXITR .EQ. 0)MAXITR = 10     !Max iterations may be specified
      LFLG2 = 0
      LFLG4 = 0
      FIXFLG2 = 0
      FIXFLG4 = 0
      ITFLG=0            !INIT ITERATION COMPLETION FLAG
      ASLP = SHAPC(2)/GAIN

C     C     C     C     C     C     C     C     C     C
C     Five NFLG's flag the freedom of ALPHA, short-term tail size &
C     slope, and long-term tail size & slope, respectively.
C     Reiteration comes back to here.

1     KK = NFLG1 + NFLG2 + NFLG3 + NFLG4 + NFLG5

      SG = -2.7726 / ALPHA
      CX = SQRT(-ALPHA)
      XPTS = NDPTS

C     CALCULATE PEAK TAILING INTENSITIES AT THE GROUP BOUNDARIES.
C     THESE VALUES USED LATER TO ADJUST NET COUNTS OF PEAK
C     GROUPING.

      BG(1) = 0.
      BG(2) = 0.
      TLS = EXP4/GAIN
      IF ((EN(1) .GT. 4500.).AND.(TLS .LT. .1).AND.(TLS .GT. 0.))
     +  GOTO 56  !Assume Alpha Spectrum where long term tail = BG

      DO 55 J = 1,NP
      AMP1 = EXP1
      AMP2 = EXP3
      XI = PKPOS(J) + XTRAPL   !Correct for remote bkgds.
```

```
      DO 50 L = 1,2
      CC = 0.0
      IF(GAMA(J) .EQ. 0.)GO TO 10
      WL = GAMA(J) * CX
      CC = CX * XI
      EXPNT = ALPHA * XI * XI
      EXPN1 = 0.
      IF(EXPNT .GT. -14.)EXPN1 = EXP(EXPNT)
      CALL BWF2(WL,CC,EXPNT,EXPN1)
10    BG(L)=BG(L)+PKHT(J)*(CC+AMP1*EXP(-EXP2*XI)
     1 * (1.- EXP(.4*ALPHA*XI*XI)) + AMP2*EXP(-EXP4*XI))
      AMP1 = 0.
      AMP2 = 0.
      XI = XPTS - (PKPOS(J) + XTRAPH)
50    CONTINUE
55    CONTINUE

56    YP = 0.
      DBG = BG(2) - BG(1)
      CALL AVE(1,NDPTS,YNSUM,AV,YNET)

C     ZERO COEFFICIENT ARRAY

      DO 60 J=1,MXFP
      XSUM (J) = 0.
      DO 60 I=1,MXFP
60    APHA(I,J)=0.
      DO 61 J=1,NP
61    SUM(J) = 0.0
      CHISQ = 0.

      FORM MATRIX OF LINEAR EQUATIONS

      DO 200 I = 1,NDPTS
      DO 65 JK = 1,MXFP
65    AX(JK) = 0.
      XI = I
      FX = 0.0
      DALFA = 0.0
      DBETA = 0.0
      DGAMA = 0.0
      DDELT = 0.0
      DCHI = 0.0
      YP = YP + YNET(I)
      DELY(I) = YNET(I) + BG(1) + YP * DBG / YNSUM
      RM(I) = DELY(I)
      TYNET = YNET(I)
      IF (TYNET .LE. 0) TYNET = TYNET + 1.
      EY = ABS(TYNET) + AVEBG + 1.0
      IF (EY .LT. 0.)  EY = 1.0
      EY = SQRT(EY) + 0.001*EY          !Channel width uncertain

C     Prepare for profile plotting only if requested.
      IF(MFLGS(4) .EQ. 0)GO TO 75
      KPHF = 0          !Clear ctr. for free peak profiles.
      SVYNET(I) = DELY(I)
      ERYNET(I) = EY
      DO 70 JK = 1,MXNP         !Clear profile arrays.
      KKB(JK) = 0
70    PKPRFL(JK) = 0.
```

```
        LBF = 0          !Buffer #
        XEN(I) = EST + (I-1)*GAIN

C       DETERMINE WEIGHTING FACTOR
75      WT = 1.0 / EY
        KPK = KK

C       CALCULATE COEFFICIENTS FOR EACH OF "NP" PEAKS
        DO 170 J = 1,NP
        DLTAX = XI - PKPOS(J)
        DX = DLTAX
        DLX = DLTAX
        DLXSQ = DLTAX * DLTAX
        FX2 = 0.0
        FX3 = 0.0
        CB1 = 0.
        CB2 = 0.
        EXPN1 = 0.0
        PHT = PKHT(J)
        ALFA = -2.7726 / (ASLP * PKPOS(J) + SG)
        EXPNT = ALFA * DLXSQ
        EXPN3 = 0.
        IF (EXPNT .GT. -14.)EXPN1 = EXP(EXPNT)

C       CALCULATE GAUSSIAN COMPONENT OF PEAK
        IF(DLTAX .LT. 0)EXPN3 = EXP (0.4 * EXPNT)
        FX1 = EXPN1
        IF(GAMA(J) .LE. 0)GO TO 80

C       COMPUTE MODIFIED SHAPE FOR X-RAY PEAK
        CX = SQRT(-ALFA)
        WL = GAMA(J) * CX
        CC = CX * DLTAX
        CALL BWF2(WL,CC,EXPNT,EXPN1)
        FX1 = CC
80      IF(NFLG1 .GT. 0) THEN    !Is peak width parameter free?
            DALFA = DALFA + PHT * DLXSQ * EXPN1          !Yes
        ENDIF
        IF (DLTAX .GE. 0)GO TO 130
        EXPN = EXP2 * DLTAX
        IF ((EXPN + 10.) .LT. 0)GO TO 110
        EXPN2 = EXP (EXPN)
        CB = PHT * EXPN2 * (1.0 - EXPN3)
        CB1 = EXP1 * EXP2 * CB
        FX2 = EXP1 * CB
        IF (NFLG2 .GT. 0) THEN   !Is short term tailing amplitude free?
            DBETA = DBETA + CB   !Yes
        ENDIF

        IF (NFLG3 .LE. 0)GO TO 110

C       SHORT TERM TAILING SLOPE FREE
        DGAMA = DGAMA + DLTAX * CB

110     IF(EXP3 .LE. 0)GO TO 120
        EXPN = EXP4 * DLTAX
        IF((EXPN + 10.) .LT. 0)GO TO 120
        CB = PHT * EXP(EXPN) * (1. - EXPN3)
        CB2 = EXP3 * EXP4 * CB
        FX3 = EXP3 * CB
```

```
        IF(NFLG4 .GT. 0)  THEN  !Is long term tailing amplitude free?
            DDELT = DDELT + CB  !Yes
        ENDIF

        IF(NFLG5 .LE. 0)GO TO 120

L       LONG TERM TAILING SLOPE FREE

        DCHI = DCHI + DLTAX * CB
120     IF (NFLG3+NFLG5 .EQ. 0) THEN
          IF (NFLG2 .EQ. 0)  FX = FX + FX2
          IF (NFLG4 .EQ. 0)  FX = FX + FX3
        ELSE
          FX = FX + FX2 + FX3
        ENDIF

130     IF(NYFLG(J) .GT. 0)THEN !Is peak size freed?
                KPK = KPK + 1   !Yes, count free param.
                IF(MFLGS(4) .NE. 0)KPHF = KPHF + 1      !Count the buffer
                L=KPK
                VALU=FX1*WT
                GO TO 140
        ENDIF

C   Peak size is not free.  [NYFLG is not positive.]
        IF (HIGHT(J) .GE. 0.)  THEN
                FX = FX + FX1 *PHT      !Peak size is fixed absolutely.
                IF(MFLGS(4) .EQ. 0) GO TO 160
                MFLGS(4) = 2            !Flag: a profile stored at MXNP
                LBF = MXNP              !Assign 1 buf. to all fixed peaks
                PKPRFL(MXNP) = PKPRFL(MXNP) + FX1*PHT + FX2 + FX3  !Store
                IF(ITFLG .EQ. 0) GO TO 160
                L = MXNP
                GO TO 150
         ENDIF
C     Peak size is fixed relative to another peak.
         LPK = -NYFLG(J)                !Find index of primary peak.
         IF (HIGHT(LPK) .GT. 0.) THEN
                FX = FX-HIGHT(J)*PKHT(LPK)*FX1  !Primary is fixed also.
                GOTO 160                        !Skips plotting!
         ENDIF

         VALU=-HIGHT(J)*FX1 *WT
         L = KK
         IF(LPK-1 .GT. 0) THEN
           DO 135 M=1,LPK-1     !Skip stg. space already used up.
           IF(NYFLG(M) .GT. 0)L= L+1
           IF(NXFLG(M) .GT. 0)L=L+1
135        CONTINUE
        ENDIF
        L= L+1          !Locn. indexing profile of primary .


140     AX(L) =VALU + AX(L)
        IF(MFLGS(4) .EQ. 0)GO TO 160    !No plots

        Store peak profile of each free peak.
        PKPRFL(L) = PKPRFL(L) + FX1 * PHT + FX2 + FX3
        LBF = KPHF
```

```
      IF(ITFLG .EQ. 0)GO TO 160

C     Iterations are finished, clean up plotting.
      IF(NYFLG(J) .LT. 0) THEN          !Is peak size freed?
       IF(KKB(L) .EQ. 0)GO TO 160       !No. Look for index of relative
        LBF = KKB(L)                    !Substitute this profile.
        GO TO 150
      ENDIF

       DO 148, K = J,NP
       IF(-NYFLG(K) .EQ. J)GO TO 153
148    CONTINUE
150    PRFILS(I,LBF) = PKPRFL(L)
       GO TO 160
153    KKB(L) = LBF

C     C       C       C       C
C     PEAK POSITION FREE?
160   IF (NXFLG(J) .GT. 0)THEN
        KPK = KPK + 1                   !Yes. Count the free parameter
        AX(KPK)= (-2.*ALFA * DX *EXPN1 *PHT -CB1-CB2) * WT
      ENDIF

      SUM(J) = SUM(J) + FX1 * PHT + FX2
      RM(I) = RM(I) - FX1 * PHT - FX2 - FX3
170   CONTINUE

      L = 0
      IF (NFLG1 .GT. 0) THEN
              L = L + 1
              AX(L) = DALFA * WT
      ENDIF
      IF (NFLG2 .GT. 0) THEN
              L = L + 1
              AX(L) = DBETA * WT
      ENDIF
      IF (NFLG3 .GT. 0) THEN
              L = L + 1
              AX(L) = DGAMA * EXP1 * WT
      ENDIF
      IF(NFLG4 .GT. 0)THEN
              L = L + 1
              AX(L) = DDELT * WT
      ENDIF
      IF(NFLG5 .GT. 0) THEN
              L = L + 1
              AX(L) = DCHI * EXP3 * WT
      ENDIF

      DELY(I) = (DELY(I)-FX) * WT

      DO 190 KM = 1,KPK
      XSUM(KM) = XSUM(KM) + AX(KM)*DELY(I)
      DO 190 KL = 1,KPK
190   APHA(KM,KL) = APHA(KM,KL) + AX(KM)*AX(KL)

      RM(I) = RM(I) * WT
      CHISQ = CHISQ + RM(I)**2
200   CONTINUE
```

```
C       Matrix is now loaded.

        IF(ITER .EQ. 1)GO TO 210
        RCHISQ = CHISQ/(NDPTS-KPK)
        IF (LONGPR)WRITE (LOUT,1208)RCHISQ
 )8     FORMAT(' REDUCED CHI-SQUARE VALUE IS ',E15.6)

210     IF(ITFLG .EQ. 1) GO TO 500
        CALL MATINV(APHA,MXFP,KPK,DET)

        IF(DET .EQ. 0.) THEN
        TYPE *,'MATRIX IS SINGULAR'
        ABORT = .TRUE.
        RETURN
        ENDIF

        POS = NDPTS/2  !For FWHM at mid-energy
        IF (NP .EQ. 1)  POS = PKPOS(1)  !At peak position
        FWHM = GAIN * SQRT(ASLP*POS + SG - .462)
        EX2 = EXP2/GAIN
        EX4 = EXP4/GAIN
        IF (LONGPR)WRITE (LOUT,9040)
9040    FORMAT(/,' ITERATION  FWHM    EXP1    EXP2    EXP3    EXP4')
        IF (LONGPR)WRITE (LOUT,9030)ITER,FWHM,EXP1,EX2,EXP3,EX4
9030    FORMAT(I3,6X,F9.5,4F8.4)
        IF (LONGPR)WRITE (LOUT,9010)
9010    FORMAT(/,'  CHANNEL     KEV     PEAK HEIGHT')
        DO 280 J=1,NP
        PKP=PKPOS(J)+OFFSET
        PKE =(PKP - REFCH)*GAIN + REFEN
        IF (LONGPR)WRITE (LOUT,9020)PKP,PKE,PKHT(J)
 )      CONTINUE
9020    FORMAT(2F9.2,F14.3)

        DO 310 L = 1,KPK
        DEL(L) = 0.

D       IF(APHA(L,L) .LT. 0)TYPE 1310,L
D1310   FORMAT(/' *** Warning - Negative matrix element at L=',I3/)

        DM(L) = SQRT(ABS(APHA(L,L)))
        DO 310 J=1,KPK
310     DEL(L) = DEL(L) + XSUM(J)*APHA(L,J)
C
C     INCREMENT THE TRIAL VALUES AND CHECK FOR COMPLETION OF ITERATIVE
C     PROCESS.

        DO 320 J = 1,5
        IF(NFLG(J) .LE. 0) ERR(J+2) = 0.
320     CONTINUE

        L = 0
        LFLG = 0
        IF (NFLG(1) .LE. 0)GO TO 410

C       INCREMENT PEAK WIDTH PARAMETER

        L = L + 1
        ERR(3) = DM(L)
        ADEL = ABS(DEL(L))
```

```
      ALPHA = ALPHA + DEL(L) * (1. + ADEL/(ALPHA - ADEL))
      IF ((ADEL + .01 *ALPHA) .GT. 0) LFLG = 1
410   IF (NFLG(2) .LE. 0)GO TO 425
      L = L + 1
      ERR(4) = DM(L)

C     INCREMENT SHORT TAILING AMPLITUDE

      IF (NFLG3+NFLG5 .EQ. 0)  GOTO 419
      IF ((EXP1 + DEL(L)) .LE. 0)  DEL(L) = -.5 * EXP1
      ADEL = ABS(DEL(L))
      EXP1 = EXP1 + DEL(L) * (1. - ADEL/(EXP1 + ADEL))
      IF ((ADEL - DM(L) - .01 * EXP1) .GT. 0)  LFLG = 1
      GOTO 425

419   IF(DEL(L) .GT. 0.)GO TO 420
      IF (LONGPR)TYPE 1419
1419  FORMAT(' SHORT TERM TAILING AMPLITUDE NEGATIVE--SET TO 0.001')
      DEL(L) = 0.001
      LFLG2 = LFLG2 + 1
      IF(LFLG2 .GT. 4) THEN
              NFLG2 = 0
              LFLG = 1
              FIXFLG2 = 1
      ENDIF
420   IF (ABS(EXP1 - DEL(L)) - .5 * DM(L) .GT. 0)  LFLG = 1
      EXP1 = DEL(L)

425   IF (NFLG(3) .LE. 0)GO TO 435

C     INCREMENT SHORT TERM TAILING SLOPE

      L = L + 1
      ERR(5) = DM(L)
      IF ((EXP2 + DEL(L)) .LE. 0)DEL(L) =- .5 * EXP2
      ADEL = ABS(DEL(L))
      EXP2 = EXP2 + DEL(L) * (1. - ADEL / (EXP2 + ADEL))
      IF ((ADEL - DM(L) - .02* EXP2) .GT. 0)  LFLG = 1
435   IF(NFLG(4) .LE. 0)GO TO 440

C     INCREMENT LONG TERM TAILING AMPLITUDE

      L = L + 1
      ERR(6) = DM(L)
      IF (NFLG3+NFLG5 .EQ. 0)  GOTO 437
      IF ((EXP3 + DEL(L)) .LE. 0)  DEL(L) = -.5 * EXP3
      ADEL = ABS (DEL(L))
      EXP3 = EXP3 + DEL(L) * (1. - ADEL/(EXP3 + ADEL))
      IF ((ADEL - DM(L) - .01 * EXP3) .GT. 0)  LFLG = 1
      GOTO 440

437   IF(DEL(L) .GT. 0.)GO TO 438
      IF (LONGPR)TYPE 1437
1437  FORMAT(' LONG TERM TAILING AMPLITUDE NEGATIVE--SET TO .0001')
      DEL(L) = 0.0001
      LFLG4 = LFLG4 + 1
      IF (LFLG4 .GT. 4) THEN
              NFLG4 = 0
              LFLG = 1
              FIXFLG4 = 0
```

```
        ENDIF
438     IF (ABS(EXP3 - DEL(L)) - 0.5 * DM(L) .GT. 0)  LFLG = 1
        EXP3 = DEL(L)
440     IF(NFLG(5) .LE. 0)GO TO 445

        INCREMENT LONG TERM TAILING SLOPE

        L = L + 1
        ERR(7) = DM(L)
        IF((EXP4 + DEL(L)) .LE. 0)DEL(L) = -0.5 * EXP4
        ADEL = ABS(DEL(L))
        EXP4 = EXP4 + DEL(L) * (1. -ADEL / (EXP4 + ADEL))
        IF((ADEL - DM(L) - 0.02 * EXP4) .GT. 0)  LFLG = 1
445     K = L
        DO 460 J = 1,NP
        IF (NYFLG(J) .LE. 0)GO TO 450    !If fixed height

C       GET NEW PEAK HEIGHTS

        K = K + 1
        PKHT(J) = DEL(K)
        IF (DEL(K) .LT. 0.0) PKHT(J) = -1.0      !Damper
450     IF (NXFLG(J) .LE. 0)GO TO 460            !If fixed posn.

C       INCREMENT PEAK POSITIONS
        K = K + 1
        ADEL = ABS(DEL(K))
        PKPOS(J) = PKPOS(J) + DEL(K) * (1. - ADEL/(1.+ADEL))
        IF((ADEL - DM(K) - .01) .GT. 0)LFLG = 1
460     CONTINUE

        REESTABLISH PEAK ENERGY AND INTENSITY RELATIONSHIPS
C       SPECIFIED IN INPUT.

        DO 470, J = 1,NP
        IF(NYFLG(J) .GE. 0)GO TO 465
        IF(HIGHT(J) .GT. 0)GO TO 464
        IF(HIGHT(J) .EQ. 0)THEN
                PKHT(J) = 0      !Eliminated peak.
                GO TO 465
        ENDIF
        L = -NYFLG(J)
        PKHT(J) = -HIGHT(J) * PKHT(L)
        GO TO 465
464     PKHT(J) = PKHT(J) *HIGHT(J) / SUM(J)

465     IF(NXFLG(J) .GE. 0)GO TO 470
        L = -NXFLG(J)
        PKPOS(J) = PKPOS(L) -(EN(L) - EN(J))/GAIN
470     CONTINUE
        IF(LFLG .EQ. 0)ITFLG = 1
        IF (ITER .GE. MAXITR)  ITFLG = 1
        IF(ITFLG .EQ. 0)ITER = ITER + 1

C       ITERATE CALCULATION AGAIN

        GO TO 1

500     IF(ITER .GE. MAXITR.AND. LONGPR)TYPE 1500
1500    FORMAT(' MAXIMUM ITERATIONS REACHED.')
```

```
C       Check for missing peaks. If found, set flags & retry fitting.
        L = 0
        DO 530, J=1,NP
        IF(PKHT(J) .GE. 0) GO TO 530
        L = L + 1
        HIGHT(J) = 0.
        PKHT(J) = 0.
        NYFLG(J) = -J
        NXFLG(J) = 0
        SUM(J) = 0.
530     CONTINUE
        IF(L .EQ. 0)    RETURN

        ITFLG = 0
        ITER = 1
        GO TO 1          !Try again.

        END



CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       VERSION 29-OCT-82

        SUBROUTINE BWF2(WL,X,EXPNT,EXPN1)

        DIMENSION CA(5),CB(6)
        DATA CA/.729281,.0344557,.127639,-.0325239,.00642257/
        DATA CB/.933301,-4.45603,1.16191,-.141124,-.0352464,.00744387/

        C1 = 1. - .5642*WL
        C2 = .15915 * WL
        C3 = .25 * WL**2
        C4 = .6366 * WL
        RT = C1 + C3
        X = ABS(X)

        IF (X .GE. 2.2)  GOTO 10
        XX = X * X
        AX = 1.
        SUM = 0.0
        DO 4 I = 1,5
        AX = AX * XX
4       SUM = SUM + CA(I) * AX
        X = SUM * EXP (-XX)
        GO TO 20
10      XLOG = ALOG (X)
        SUM = CB(1)
        DO 12 I = 2,6
12      SUM = SUM + CB(I) * XLOG ** (I-1)
        X = EXP (SUM)
20      AA = (C1 - C2*EXPNT + C3*(1. + 2.*EXPNT))*EXPN1 + C4*X
        X = AA / RT

        RETURN
        END
```

```
C*****************************************************************
C
C  FRDSK
C
C       Responds to operator selection of assay from disk and
        background from disk
C
C       CMS 10-MAY-84
C       REV.11-SEP-84  CMS
C       Rev 29-Nov-84 MPK       Replace list-directed with
C                               formatted I/O
C       Rev. 12-FEb-85  MPK     Redo logical flags
C       Rev. 04-Apr-86  WDR     Modified for Micro VAX II
C       Rev. 13-Jul-87  RDP     Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C
C*****************************************************************
C
C
        SUBROUTINE FRDSK
        CHARACTER   SAVFIL*15
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CFILES.CMN'
C
C
        FROMDK = .TRUE.
        RTYP = 'Fromdisk'
        IF(BD) TYPE 1000
 1000   FORMAT ('  Background from disk')
        IF(AD) TYPE 1001
 1001   FORMAT ('  Assay from disk')
C
C  Get parameter set ups from dialog
C
        CALL DIALOG
        IF(ABORT)GO TO 500
C
C  Save original filename
C
        SAVFIL = FILNAM
C
C  Perform assay
C
        CALL ASSAY
        IF (ABORT) GOTO 500
C
C  Restore filename
C
        FILNAM = SAVFIL
C
C
   Print results
C
        CALL PRINT1                         !instrument specific routine
500     RETURN
```

END

```
C-----------------------------------------------------------------------

      SUBROUTINE GETIME(NDEV,LIVTIM,RLTIM,DEDTIM)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     A sub for  picking live,real, & start times out of first
C     two channels of spectrum.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL*4 LIVTIM,RLTIM

      DIMENSION BUF(2)

      CALL RDCHNS (BUF,1,2,NDEV)
      LIVTIM = BUF(1)
      RLTIM = BUF(2)
      DEDTIM = 100. *(RLTIM - LIVTIM)


      RETURN
      END
```

```
        SUBROUTINE GET_KEY(PBID,VKID)
C*************************************************************************
C Routine gets input from the virtual keyboard buffer.
C It sets a char detected flag (INKEY = 1) and the character
C detected (INCHAR). The character is echoed to the display.

        INCLUDE '($TRMDEF)'
C
        COMMON/RTI/     INKEY,INCHAR
        INTEGER         INKEY
        CHARACTER       INCHAR*1,INPUT*4
        INTEGER*4       STATUS,VKID,PBID,LENGTH,TIMEOUT
C
        INTEGER*4       SMG$READ_STRING
C
C Read Key Input
        LENGTH  = 1
        TIMEOUT = 3
        STATUS = SMG$READ_STRING(VKID,
     c                           INPUT,
     c                           ,
     c                           LENGTH,
     c                           TRM$M_TM_NOECHO+TRM$M_TM_CVTLOW,
     c                           TIMEOUT)
        IF (.NOT.STATUS) THEN
           INCHAR= ' '
           INKEY = 0
           CALL LIB$SIGNAL(%VAL(STATUS))
        ELSE
           INCHAR= INPUT(1:1)
           INKEY = 1
        ENDIF
C
        END
```

```
C*****************************************************************
C
C  GETOPS
C       Get operator and sample ID
C
        Output argument
C
C       MTL (LOGICAL*1)          .TRUE. if material type and
C                                isotopics are needed
C
C       31-May-84 lc
C       Rev 14-Feb-85 MPK        Put in "real" VAX communications
C                                Supply end and err exits for
C                                terminal input
C       Rev 08-Oct-85 MPK        Adapted to neutron coinc counters
C                                GETOP and GETSMP combined
C       Rev 21-Oct-85 MPK        Pad "canned" sample ID properly
C       Rev 23-OCT-85 WDR        Revised for solution assay system
C       Rev 04-Apr-86 WDR        Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE GETOPS(MTL)
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CONTRL.CMN'
C
        LOGICAL*1 YESNO,MTL
        CHARACTER BSAMP*4,FSAMP*4
        DATA BSAMP /'BKG '/
        DATA FSAMP /'FOIL'/
C
C  Get operator ID; set abort flag if it is null
C
   10   TYPE 1000
 1000   FORMAT (' '/'$Enter operator ID (up to 8 chars)',
        1       ' or RETURN to escape -> ')
        READ (5,1002,END=10,ERR=10) OPERID
 1002   FORMAT(A80)
        ABORT = OPERID.EQ.' '
        IF (ABORT) RETURN
C
C  See whether sample ID, material type, and isotopics are needed
C
        IF (FROMDK) RETURN
        IF (AC) RETURN
        IF (BACKG) GO TO 100
        IF (.NOT.MP) GO TO 120
        IF (MP .OR. MB)GO TO 120
C
C  Supply "canned" sample ID for background or meas control
C
  100   MTL=.FALSE.
        SAMPID=FSAMP
        IF (BACKG) SAMPID=BSAMP
        SAMPID=' '
        RETURN
C
```

```
C  Get sample ID; set abort flag if it is null
C
  120    MTL=.TRUE.
   20    TYPE 1001
 1001    FORMAT (' '/' Read sample barcode or enter sample ID'
     .   1           /'$                    (RETURN to escape) -> ')
         READ (5,1002,END=20,ERR=20) SAMPID
         ABORT = SAMPID.EQ.' '
         RETURN
         END
```

```
C*****************************************************************
C GET90
C
C       Routine specific to gamma ray instruments using
C       Canberra Series 90
        Get a spectrum from the Series 90 and write to disk;
C       Read chunks of 256 channels at a time
C
C       SSJ 28-JUN-79
C       Rev. 13-Jun-84 lc
C       REV. 12-Dec-85 WDR      Put start time and date in channels
C                               3 and 4
C       Rev. 19-Feb-85 MPK      Change ACCEPT to READ with ERR and
C                               END exits
C       Rev. 20-MAR-85 WDR      Modified to do FROMCA rather than READ80
C       Rev. 04-APR-86 WDR      Modified for Micro VAX II
C       Rev. 05-JUL-87 RDP      Modified for uVAX II w/ ADAC Drivers
C
C*****************************************************************
C
        SUBROUTINE GET90(NADC)
C
        LOGICAL*1 ANS,SMALL
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CTIM.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CDATE.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray and x-ray multichannel instruments
C
        INCLUDE 'CSPECT.CMN'
        INCLUDE 'CONT90.CMN'
C
C
C  Initialize for writing
C
        NSPEC = MEMSIZ
 5      NRECS = NSPEC/128
        KTIMES = 0
        SMALL = .FALSE.
        DUM = -999.0
D       WRITE(5,5568)NSPEC,NRECS,NADC
D5568   FORMAT(' NSPEC = ',I5,'NRECS = ',I5,'NADC = ',I5)
C
C  Open the output file on disk first
C  If any problems occur (such as the program bombing
C  due to no disk), we do not want the Series 90
C  to be opened up yet
C
   Allocate channel, free it when through
C
 15     CONTINUE
        OPEN(UNIT=DKLUN1,NAME=FILNAM,ACCESS='DIRECT',TYPE='NEW',
```

```
     X           RECORDSIZE=128,MAXREC=NRECS+1,ERR=20)
         IREC = 1
         GO TO 25
C
C  Problem in opening the file
C
 20      CALL ERRMSG(1,FILNAM)
C
C  Free channel IOCHAN
C
C         CALL IFREEC(IOCHAN)
C
         KTIMES = KTIMES + 1
         IF(KTIMES .LE. 1) GO TO 15
         TYPE 1005
 1005    FORMAT(/'$Enter A to abort writing spectra to disk',
     X           '   -> ')
         READ (5,1007,END=15,ERR=15)ANS
 1007    FORMAT(A1)
         IF (ANS .EQ. 'A') WRITNG = .FALSE.
         IF (ANS. EQ. 'A') GOTO 500
         GO TO 15
C
C  Read one 256-channel block
C  Check for errors in the read
C
25       ICHNF = 0
         ICHNL = 127
30       CONTINUE
         CALL FROMCA(NADC,ICHNL,ICHNF,SPCTRM,IERR,)
         CALL CHK90('FROMCA',IERR)
         NUMCHN = IERR
         IERR = IERR-128
         IF(IERR .EQ. 0 .AND. IREC .EQ.1) GO TO 32
         IF(NUMCHN .NE. 128) TYPE 1002,NUMCHN
 1002    FORMAT(//20X,'****', I6,' CHANNELS READ - EXPECTED 128 ****')
         IF(IREC .GT. 1) GO TO 35
         GO TO 500
C
C  First record
C  Check to see if file is big enough
C  Change number of records for actual spectrum size
C
 32        TYPE 1003,NSPEC,NSTART
1003    FORMAT(/I5, ' Channels starting at channel # ',I5/)
         IF(NSPEC/128 .GT. NRECS) GO TO 33
         NRECS = NSPEC/128
         GO TO 35
C
C  This file is not large enough
C  Go on to close the file and the series 90
C  and start over
C
 33      SMALL = .TRUE.
         GO TO 100
C
C   In the case of a cumulative run, insert the cumulative
C   live and clock times in place of SPCTRM(1) & (2)
C
 35      IF((.NOT. CUMLTV) .OR. (IREC .NE. 1)) GO TO 50
```

```
        SPCTRM(1) = TIMELV
        SPCTRM(2) = TIMECL
C
C       Insert the start time and date in SPCTRM(3) and (4)
C       Make sure it is the first record

        IF (IREC .NE. 1) GO TO 55
        SPCTRM(3) = STIM
        SPCTRM(4) = SDAT
C       TYPE *,'SDAT IN GET90',SDAT
C
C  Write this 128 channel block to disk
C
55        WRITE(DKLUN1'IREC)SPCTRM
        IREC = IREC + 1
        ICHNF = ICHNL + 1
        ICHNL = ICHNL + 128
        IF(IREC .LE. NRECS) GO TO 30
C
C  Close the ADC and the disk file
C  Write current background data to disk
C
100       IF(SMALL) GO TO 5
        WRITE(DKLUN1'IREC)DUM,BK,SGBK,TIMEBK,BKDAT,BKTIM,
     X           BKOP,BKSFIL,SAMPID,OPERID,SYSID,ASDATE,ASTIME,
     X           MATTYP(1)
 101    CLOSE(UNIT=DKLUN1)
C       CALL IFREEC(IOCHAN)
        NBLKS = IREC
        TYPE 1006,NBLKS,FILNAM
 1006   FORMAT(/' Spectrum retrieval completed --',
     X           I6,' blocks written to ',15A)
500    RETURN
        END
```

```
C*****************************************************************************
C HEADR38       (HEADER subroutine for PU-238 isotopic system)
C
C       Print assay for PU-238 isotopic instrument
C
C       Calling arguments:
C            SUMMRY = Logical flag for writing summary for cycles
C
C       07-Jun-84 lc
C       04-Sept-84      TES  16:38
C       11-SEP-84       CMS
C       06-NOV-84       WDR     Modified for solid isotopic instrument
C       27-APR-90       WDR     Modified for Pu-238 isotopic instrument
C
C*****************************************************************************
C
        SUBROUTINE HEADER(SUMMRY)
C
        LOGICAL*1  SAMPLE,SUMMRY
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CONFIL.CMN'
C
C
   Print the header showing date, time, operator, etc.
C
        WRITE(LOUT,1999)
 1999   FORMAT(/1X,80(1H*)/)
C
        WRITE(LOUT,1065)RTYP
 1065   FORMAT(1X,'PU-238 isotopic gamma assay - ',T37,'Run Type:  ',
     X      A8)
C
C
        IF(.NOT. SUMMRY .AND. (AUTOCY .OR. MP)) WRITE(LOUT,1007)ICYCLE
 1007   FORMAT(/' Cycle: ',I5)
        IF(SUMMRY) WRITE(LOUT,1077) NUMCYC
 1077   FORMAT(/19X,' Summary for ',I4,' Cycles')
C
        SAMPLE = LIVE .AND. .NOT. FROMDK
        IF(SAMPLE) WRITE(LOUT,1001) SAMPID,ASDATE,ASTIME
        IF(.NOT. SAMPLE) WRITE(LOUT,1006) FILNAM,ASDATE,ASTIME
 1001   FORMAT(/' Sample ID:       ',A12,T35,'Measurement Date: ',A9,2X,
     X          A8)
 1006   FORMAT(/' Inout file:      ',A15,T35,'Measurement Date: ',A9,2X,
     X          A8)
C
        WRITE(LOUT,1005) OPERID,NOWDAT,NOWTIM
 1005   FORMAT(' Operator ID:      ',A8,T35,'Current Date:      ',A9,2X,A8)

        IF(WRITNG) WRITE(LOUT,1002) FILNAM
 1002   FORMAT(' Data file:        ',A15)
C
```

```
      WRITE(LOUT,1066) CONFIL, FILDAT, FILTIM
 1066 FORMAT(' Constants File: ',A15,T35,'Constants Date:    ',A9,2X,
     X        A8)
C
      WRITE(LOUT,1003) TIMELV,TIMECL,REMARK
 1003 FORMAT(' Live time (sec):',F8.0,T35,'Clock time (sec):',F8.0,
     X         //1X,80A1)
C
      WRITE(LOUT,1999)
C
  500 RETURN
      END
```

```
C
C*********************************************************************
C
C  HELP38       (HELP subroutine for Pu-238 isotopic instrument)
C
C       Outputs to the terminal the list of menu options for the
C       operator.  If the supv flag is true, supervisor options
C       are listed.
C
C       Calling arguments:
C       SUPV = Logical flag passed from calling program
C              If SUPV = .TRUE. then operator can select from
C                 supervisor options
C              If SUPV = .FALSE. then operator is not allowed
C                 to select from supervisor options
C
C       CMS  26-APR-84
C       REV. 07-Sep-84 CMS
C       REV. 26-JUN-85 WDR      Modified for solids isotopic system
C                               Removed B and BD options
C       REV. 04-APR-86 WDR      Modified for Micro VAX II
C
C*********************************************************************
C
        SUBROUTINE HELP(SUPV)
        IMPLICIT INTEGER (A - Z)
        CHARACTER*1 REPLY
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'

C  Output the list of options - does not include supervisor options
C
        WRITE(5,1000)
 1000   FORMAT(/
     X          ' A   -  Assay'/
     X          ' MB  -  Measurement control - bias'/
     X          ' MP  -  Measurement control - precision'//
     X          '          Options during data acquisition:'//
     X          ' Q   -  Quit assay - no results '/
     X          ' S   -  Suspend assay temporarily'/
     X          ' T   -  Terminate assay with results'/)
C
C  If supervisor mode output the rest of the options
C
        IF(SUPV)THEN
        TYPE 900,'Press RETURN to continue ->'
900     FORMAT(1X,A,$)
        READ (5,920)REPLY
920     FORMAT(A1)
        WRITE(5,1001)
 1001   FORMAT(/   '  SUPERVISOR OPTIONS '/
     X          '                     '/
     X          ' AC -  Current data in MCA '/
     X          ' AD -  From disk'/
     X          ' AU -  Autocycle'/
     X          ' AUD-  Autocycle from disk'/
     X          ' D  -  Default'/
     X          ' LA -  List assay log'/
```

```
     X           ' LM -  List measurement control log'/
     X           ' OU -  Change output listing device'/
     X           ' ST -  Status'/
     X           ' R  -  Read data from disk'/
     X           ' W  -  Write data to disk'/
     X           ' X  -  Exit from program'/)
         ENDIF
C
C
         RETURN
         END
```

```
C*****************************************************************
C INIT90
C
C       Routine specific to initializing Series 90
C
C       SSJ 13-AUG-79
C
C       REV 16-FEB-84 LC
C       REV 22-OCT-84 MJC       CHANGED 'PUT MCA IN REMOTE . . .'
C                               TO 'PUT SERIES 90 ON LINE . . .'
C                               IN LABEL 1020
C       Rev 19-Feb-85 WDR       Change ACCEPT to READ with END and
C                               ERR exit
C       Rev 04-Apr-86 WDR       Modified for Micro VAX II
C       Rev 05-Jul-87 RDP       Modified for uVAX II w/ ADAC Drivers
C
C*****************************************************************
C
        SUBROUTINE INIT90
C
        LOGICAL*1 ANS
        INTEGER IARY(12)
C
C  The following common block files are instrument specific for
C  gamma ray instruments
C
        INCLUDE 'CONT90.CMN'
C
        DATA MCA/1/
C

C  Try initializing the series 90 a maximum of 3 times
C
        ITIMES = 0
 3      ITIMES = ITIMES + 1
        CALL INIT(MCA,,IERR)
        IF(IERR .EQ. 0) GO TO 10
        IF(ITIMES .GE. 3) GO TO 8
C
C  Some problem
C  Check if MCA is not in remote (ierr = -1)
C
        IF(IERR .NE. -1) GO TO 3
        TYPE 1020
1020    FORMAT(' '/'$Put Series 90 on line - hit ''return'' when ready ',
     X          ' ->  ')
        READ (5,1001,END=3,ERR=3)
1001    FORMAT(A1)
        GO TO 3
C
C  Print error msg and terminate
C
 8      CALL CHK90('INIT  ',IERR)
        TYPE 1030
1030    FORMAT(//20X,'**** SERIES 90 INIT UNSUCCESSFUL 3 TIMES'//
     X           10X,'PROGRAM EXITING'/
     X           10X,'CHECK YOUR "AN" HANDLER ****')
        RETURN
C
```

```
C  Detach any tasks
C
 10     DO 11 I=1,3
 11       CONTINUE
C
   )0    RETURN
         END
```

```
      SUBROUTINE INITLP

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     GRPANL sub to begin line printer output of parameters & file names.
   Called from READIN (before any peak groupings)
L
C               By J.B.Niday
C               Last edit for VAX  14-Mar-88 (J.B.N.)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'

      INCLUDE 'CFILES.CMN'
      INCLUDE 'CLUNS.CMN'
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      BYTE IBUF(80)
      EQUIVALENCE (WM,IBUF),(WM(30),LTH)

      BYTE DETFNM(MXFNM),SHPFNM(MXFNM)
      EQUIVALENCE (DETFNM,RM),(SHPFNM,RM(MXFNM+1))

      DIMENSION XC(2)
C     DATA LP/-1/
C     DATA LP/6/
C     -     -     -     -     -     -     -     -
      DEDTIM = 100. * (RLTIME-TIMEL)/RLTIME    !Temp arg. only.

C     Tabulate some of initialization stuff on printer
      CALL VERS(1)
      WRITE(LOUT,1010)(DTE(J),J=1,9),(TBUF(I),I=1,8),FILNAM,
     + (ICFIL(I),I=1,12)
1010  FORMAT(/30X,'Analysis on ',9A1,' at ',8A1,' of '//
     + ' Spectral data file: ',A,'Parameter control file: ',12A1)

      DO 5, I = 1,LTH
      IF (IBUF(I) .NE. '/') GO TO 5
      II = I
      GO TO 12
5     CONTINUE
      GO TO 15
12    WRITE(LP,1012)(IBUF(I),I=II,LTH)
1012  FORMAT(4X,' Using run-time control options: ',30A1)

15    WRITE(LP,1015)(EXPTID,DTZERO,IZYR)
      WRITE(LP,1020)(SMPID(I),I=1,12),(DSNAME(I),I=1,6),DELTAT

      IF((DETFNM(1) .NE.0).AND.(DETFNM(1) .NE. ' ')) THEN
          WRITE(LP,1030)(DETFNM(I),I=1,MXFNM)
      ELSEIF ((DFILNM(1).NE.0).AND.(DFILNM(1).NE.' ')) THEN
          WRITE(LP,1035)(DFILNM(I),I=1,MXFNM)
      ENDIF

      IF (SHAPC(1) .EQ. 0) GO TO 20
      IF((SHPFNM(1).NE.0).AND.(SHPFNM(1).NE.' ')) THEN
              WRITE(LP,1040)(SHPFNM(I),I=1,MXFNM)
      ELSEIF ((SFILNM(1).NE.0).AND.(SFILNM(1).NE.' '))THEN
              WRITE(LP,1041)(SFILNM(I),I=1,MXFNM)
```

```
      ENDIF

20    WRITE(LP,1622)(BKGDFL(I),I=1,MXFNM-1)
      WRITE(LP,1625)TIMEL,DEDTIM
      WRITE(LP,1630)DSTTIM,ICTYR,DSTPTM,ISTPYR
   5  FORMAT(/' Expt: ',12A1,9X,'Zero time: ',F12.6,' (',I4,')')
1020  FORMAT(' Sample: ',12A1,6X,' Counted on: ',6A1,6X,'Decay',
     +  ' (days to midpt.): ',1PE11.5)

1030  FORMAT(' Efficiency data updated from file: ',30A1)
1035  FORMAT(' Efficiency data originally obtained from file: ',30A1)

1040  FORMAT(' Shape parameters updated from file: ',30A1)
1041  FORMAT(' Shape parameters originally obtd. from file: ',30A1)

1622  FORMAT(' Background peak file used: ',29A1)
1625  FORMAT(' Live time',F9.2,' minutes with',F7.3,'% dead time')
1630  FORMAT(' Starting day',F9.4,' (',I4,') and ending on',F9.4,
     +  ' (',I4,')')
      WRITE(LP,1635)MCHBUF,TXTBUF
1635  FORMAT(' Input text:',5X,32A1/1X,70A1)

      IF(RUNFLG(3) .GT. 0) WRITE (LP,1638)
1638  FORMAT(/' NOTE: Control file used was for a DIFFERENT detector',
     +  '-system!!!!!!'//)
      IF(RUNFLG(4) .EQ. 0) WRITE (LP,1640)
1640  FORMAT(/'  Photon calculations were not requested'//)

C     SUMMARIZE DATA FOR CALCULATION OF PHOTON INTENSITIES

      WRITE(LP,1643)DCNST(13),DCNST(14),(DSNAME(I),I=1,6)
      WRITE(LP,1645)GEOM,GEOM+DCNST(15),SURFC,DEPTH,RHO,SMPWT,WT100,
     +  ANSFCT

1643  FORMAT(/' Detector parameters: Height =',F7.3,' cm.; Radius =',
     +  F7.3,' cm. for ',6A1)
1645  FORMAT(' Sample midplane to detector window =',F7.3,' cm.;',
     +  ' total dist. used =',F7.3,' cm.'/' Sample surface area =',F6.2,
     +  ';  Depth =',F6.3,' cm.;  Density =',F7.3,' g/cm3'/' Weight = ',
     +  F11.6,'g;  100% weight =',F12.6,'g;  Normalization factor =',
     +  1PE12.4)

660   IF(NF1 .EQ. 0)GO TO 680
1660  FORMAT (/' Sample composition:' /' Element  Amount(%)')
      WRITE(LP,1660)
      DO 675, K=1,NF1
      WRITE(LP,1675)MTLZ(K),CMPOS(K)
675   CONTINUE
1675  FORMAT(3X,A2,F12.4)

680   WRITE(LP,1680)
1680  FORMAT (/' Absorbers used: ',$)
      IF(NF2 .EQ. 0) THEN
              WRITE(LP,1681)
              GO TO 700
      ENDIF
  31  FORMAT(' NONE'/)
      WRITE(LP,1682)
1682  FORMAT(/' Element & amount (g/cm2)')
      DO 685, K=1,NF2
```

```
685       WRITE(LP,1675)NABS(K),ABSRB(K)

700       IF(KTGRP.EQ.0)WRITE(LP,1001),GAIN,ZERO
1001      FORMAT(' Initialize at approx. GAIN of',F9.6,
     + ' keV/chan.; & chan. ZERO =',F9.4' keV'//)

          RETURN
          END
```

```
C*************************************************************************
C
C  INTFLG -
C
C       Initialize logical flags to be used in the assay
C       procedure.  This is called when the system is first
C       initialized and after each option has been exercised.
C
C
C       CMS   7-MAY-84
C
C       Rev    2-Oct-84 CMS      Added CCAL.CMN Include statements
C       REV   11-OCT-84 MJP      Added common block CCAL
C       Rev   08-Feb-85 MPK      Initialize all flags that need
C                                initialization
C       Rev 13-Jun-85 MPK        Change for NCC instruments
C       Rev 19-Jul-85 MPK        Make DGFLG an array; add CONDIT
C                                and BAD flags
C       Rev 04-Apr-86 WDR        Modified for Micro VAX II
C
C*************************************************************************
C
        SUBROUTINE INTFLG
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CCAL.CMN'
        INCLUDE 'CVAX.CMN'
        INCLUDE 'CDIAG.CMN'
        INCLUDE 'CMC.CMN'
        INCLUDE 'CASSAY.CMN'
C
C  initialize logical flags
C
        ABORT  = .FALSE.
        BACKG  = .FALSE.
        FROMDK = .FALSE.
        AUTOCY = .FALSE.
        OLDATA = .FALSE.
        CUMLTV = .FALSE.
        STPCYC = .FALSE.
        ASFLG  = .FALSE.
        AC     = .FALSE.
        AUD    = .FALSE.
        AD     = .FALSE.
        MB     = .FALSE.
        MP     = .FALSE.
        TKEY   = .FALSE.
        CHANG  = .FALSE.
        CHANG1 = .FALSE.
C
        BD     = .FALSE.
C
        CAL    = .FALSE.
        CALDSK = .FALSE.
C
        VAXUP  = .FALSE.
        NOGO   = .FALSE.
```

```
C
      DO 100 I=1,10
  100 DGFLG(I)=.FALSE.
      CONDIT = .FALSE.
      BAD    = .FALSE.

      MCERR  = .FALSE.
C
      RETURN
      END
```

```
C*****************************************************************
C  JULDAL       (Livermore version of JULDAT)
C
C       Using ASCII strings for date, time, compute Julian day
C       From Sue Johnson's Julday
C
C       Calling arguments:
C       IDAT - Input ascii string for date  30-apr-84
C       ITIM - Input ascii string for time  13:54:17
C       XDAT - Output Julian day
C       XTIM - Output time of day in seconds
C
C       12-APR-83 LC
C       REV. 02-MAY-84  LC      for generic package
C       REV. 31-AUG-84  CMS
C       REV. 13-DEC-84  WDR     made day 0 1/1/1900
C                               modified days calculation
C                               Note: Day 0 1/1/1900
C
C       Rev. 11-Oct-84  mjc:    IDAT is accepted as 8 characters (1-May-84)
C                               or 9 characters (10-May-84)
C       Rev. 30-Oct-84  mpk     Fix error in 11-Oct correction -
C                               tighten code
C       Rev. 31-Oct-84  mpk     Tighten code some more, and handle
C                               time format more flexibly.  This is
C                               done by copying date and time into
C                               an auxiliary array, and editing the
C                               array to convert all special characters
C                               into commas.  Then the DECODE can use
C                               the comma-delimited integer field
C                               feature.
C                               Also, correct Julian date algorithm
C                               to use integer arithmetic.
C                               In the correct algorithm, day 0 comes
C                               out to 30-OCT-79, not 28-OCT as
C                               stated previously.
C       Rev. 01-Nov-84  mpk     Leave spaces in date/time string
C                               rather than editing to commas.
C       Rev. 02-Nov-84  mpk     Latest installment in space saga:
C                               Ignore leading spaces and replace
C                               trailing spaces with commas; this
C                               makes it work right for time inputs
C                               like 11:23 (seconds omitted).
C       Rev. 10-DEC-85  WDR     Modified to calculate number of days
C                               from 1/1/1900 rather than as described
C                               above.
C       Rev.  3-Apr-86  WDR     Modified for Micro VAX II
C
C*****************************************************************
C
      SUBROUTINE JULDAT(IDAT,ITIM,XDAT,XTIM)
C
      CHARACTER XMONTH*3
      CHARACTER*1 IDAT(9),ITIM(8), DATTIM(23)
      LOGICAL*1 TERMFG
      DIMENSION NDAYS(12)

      DATA NDAYS /31,28,31,30,31,30,31,31,30,31,30,31/
C
C  The following common block files are generic for all instruments
```

```
C
      INCLUDE 'CMONTH.CMN'
C
C
C  Decode ASCII strings
      DO 5 I=1,9
 5    DATTIM(I)=IDAT(I)
      DO 6 I=1,8
 6    DATTIM(I+9)=ITIM(I)
D     TYPE 1998,DATTIM
C
C  Replace special chars ('-' or ':') and trailing spaces with commas
C
      TERMFG = .TRUE.
      DO 7 I=1,17
      IF (DATTIM(I).NE.' ') GO TO 71
      IF (TERMFG) GO TO 7
      GO TO 72
71    IF (DATTIM(I).GE.'0' .AND. DATTIM(I).LE.'9') GO TO 73
      IF (DATTIM(I).GE.'A' .AND. DATTIM(I).LE.'Z') GO TO 73
72    DATTIM(I)=','
      TERMFG = .TRUE.
      GO TO 7
73    TERMFG = .FALSE.
 7    CONTINUE
D     TYPE 1998,DATTIM
D1998 FORMAT (' EDITED DATE/TIME: ',17A1)
      IDAY = 0
      XMONTH = '   '
      IYEAR = 0
      IHOUR = 0
      IMIN = 0
      ISEC = 0
      DECODE (9,1000,DATTIM,ERR=75) IDAY,XMONTH,IYEAR
 1000 FORMAT (I3,A3,1X,I4)
75    DECODE (8,1001,DATTIM(10),ERR = 76) IHOUR,IMIN,ISEC
 1001 FORMAT (2I5,I2)
76    CONTINUE
C
C  Compare month to common
C
      IMONTH = 0
      DO 9 I=1,12
 9    IF(XMONTH .EQ. AMONTH(I))IMONTH = I
D     TYPE 1999,IMONTH,IDAY,IYEAR,IHOUR,IMIN,ISEC
D1999 FORMAT(' DATE:',3I3,'  TIME:',3I3)
C
C  Convert time to seconds
C
      XTIM = ISEC + 60. * (IMIN + 60. * IHOUR)
C
C  Convert date to days
C
C     IF (IYEAR .GT. 1900) IYEAR = IYEAR - 1900
      NMONTH = IMONTH - 1

 .0   XDAT = AINT (365.25 * IYEAR)
C     1    + AINT (30.6001 * NMONTH)
      2    + IDAY
```

```
C       3    + XTIM/86400.       !uncomment to include fractional day
C
C  Add the number of days up to this month
C
        IF (NMONTH.EQ.0)GO TO 16
        DO 15 J =1,NMONTH
15      XDAT = XDAT + NDAYS(J)
16      IF(NMONTH .LT. 3)GO TO 500
        YR = IYEAR
C
C  Include an extra day for leap year
C
        IF(AMOD(YR,4.).EQ. 0.)XDAT = XDAT + 1.
C
500     RETURN
        END
```

```
C*********************************************************************
C
C  LGMENU
C       Lists menu options for the list assay log and list
C       measurement control log
        Options are:
C               Return to main menu
C               Entries not sent to VAX
C               Entries between two dates
C               All entries
C               'N' entries
C
C       Returns number of option selected
C
C       Calling arguments:
C       Input:  LGNAME - name of log being listed, 23A1
C       Output:  NANS - menu option number, I
C
C       CMS 14-MAY-84
C       REV 10-SEP-84 mjc
C       Rev 24-Jan-85 MPK       Add menu option 0 - "return to main menu"
C       Rev 25-Jan-85 MPK       Add code to handle variable-length name
C       Rev 04-Feb-85 MPK       Reset control/O before menu output
C       Rev 04-Apr-86 WDR       Modified for Micro VAX II
C       Rev 04-Apr-90 WDR       Modified for Pu-238 instrument; 1st
C                               option searches for sample ID
C
C*********************************************************************
C
C
        SUBROUTINE LGMENU(NANS,LGNAME)

        LOGICAL * 1  LGNAME(23)
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
C
        DO 100 L=1,22
        IF (LGNAME(L+1).EQ.0) GO TO 110
  100   CONTINUE
        L=23
  110   TYPE 1000, (LGNAME(I),I=1,L)
 1000   FORMAT(/'  Menu of Options for Listing ',23A1)
        TYPE 1001
 1001   FORMAT(/'    0.  Return to main menu'
     X          /'    1.  List entries with specified sample ID'
     X          /'    2.  List entries between two dates'
     X          /'    3.  List all entries'
     X          /'    4.  List n entries')
   10   TYPE 1002
 1002   FORMAT(' '/'$ Enter option number -> ')
        READ(5,1003,END=10,ERR=10)NANS
 1003   FORMAT(I)
        IF(NANS .LT. 0 .OR. NANS .GT. 4) GO TO 10
        RETURN
        END
```

```
C
C*************************************************************************
C
C  LHEAD
C       Outputs header information for the assay log list option

C
C       CMS 14-MAY-84
C       Rev 23-Aug-84 mjc
C       Rev 11-Oct-84 cms       Changed title output from 'Data'
C                               to 'Assay'
C       Rev 04-Apr-86 WDR       Modified for Micro VAX II
C       Rev 05-Aug-87 WDR       Modified format statements for
C                               character output
C
C*************************************************************************
C
        SUBROUTINE LHEAD
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CLOG.CMN'
        INCLUDE 'CDATE.CMN'
C
        CALL DATTIM(1)
        WRITE(LOUT,1099)NOWDAT,NOWTIM
 1099   FORMAT(/'1',10X,'Assay log as of ',A9,2X,A8///
     X    1X,'Entry Date   Time  Sample   Operator File     ',
     X    '      Cycles  Results      240 Err'/22X,'Id',50X,
     X    'Flg',1X,'Flg'//)
        RETURN
        END
```

```
C*************************************************************************
C
C  LNMN38
C
C       This subroutine checks the signals from the Liquid Nitrogen
        Liquid Level Monitors (Ortec 729A) in the Pu238 isotopic
C       instrument through the ADAC 1616 CCI interface board.  If
C       bit #5 in the data register at address ZWA1: goes low,
C       this indicates a low liquid nitrogen level in detector.
C
C       REV. 06-AUG-87  RDP             Changes made to ADAC interface.
C
C*************************************************************************
C
        SUBROUTINE LNMN38
C
        INTEGER*4       VALUE,ISTAT,ISTATE
        INTEGER*2       IBUFF(2)
C
C  The following common block is generic to all instruments
C
        INCLUDE 'CONTRL.CMN'
C
C  Common Containing ADAC Channel Information
C
        INCLUDE 'ADAC.CMN'
C
        ABORT = .FALSE.
C
C  First, clear data register
C
        VALUE = 96
        ISTAT = BTSTW(VALUE,ISTATE,C_DATA,IOSB,,,)
        ESTATUS = ERROR_CHECK(ISTAT,IOSB)
        IF (.NOT.ESTATUS) TYPE*,'PROBLEM WITH ADAC (#)'
C
C  Now test bit #5
C
        VALUE = 32
        ISTAT = BTSTW(VALUE,ISTATE,C_DATA,IOSB,,,)
C       WRITE (*,*)'ISTATE',ISTATE
        ESTATUS = ERROR_CHECK(ISTAT,IOSB)
        IF (.NOT.ESTATUS) TYPE*,'PROBLEM WITH ADAC (#5)'
        IF (ISTATE.EQ.1) GOTO 500
C
C  Bit #5 is low, send warning message
C
        TYPE 900
900     FORMAT('                                        ')
        TYPE 999
999     FORMAT(/' !!!WARNING!!!')
        TYPE 1000
1000    FORMAT(' !!!Liquid Nitrogen level low in detector dewar!!!')
        TYPE 1500
1500    FORMAT(' Press RETURN to continue ')
        ACCEPT 1600
 )0     FORMAT(80A1)
        ABORT = .TRUE.
500     RETURN
        END
```

```
C*****************************************************************************
C
C  LMCLOG
C       List the measurement control log entries
C       Options for listing:
C               All entries
C               'N' entries
C               Entries not yet sent to VAX
C               Entries within a pair of dates
C       Entries are listed from current date
C
C       Calling argument:
C       AUTO    If .FALSE., the routine runs in interactive mode
C               and prompts for listing options.
C               If .TRUE., there is no operator interaction or listing;
C               all entries not yet sent to the VAX are sent.
C
C       CMS 14-MAY-84
C
C       REV.  7-SEP-84 mjc
C       REV. 18-OCT-84 MJC      closed LOUT if it equaled 6 (line printer)
C       Rev. 23-Jan-85 MPK      Remove date syntax checking, and
C                               use JULDAT to check dates for
C                               correctness
C                               Update communications to use pseudo-
C                               message types 'LB' (log bias) and
C                               'LP' (log precision)
C       Rev. 24-Jan-85 MPK      Allow multiple listings on a single
C                               entry
C                               Add end and error exits to all terminal
C                               input
C       Rev. 25-Jan-85 MPK      Change 'carr ret' in prompt to 'RETURN'
C                               Allow escaping from date mode
C       Rev. 28-Jan-85 MPK      Correct and clean up indexing and
C                               listing logic
C       Rev. 30-Jan-85 MPK      Add automatic mode for sending
C                               entries to VAX
C                               Add additional error checks for
C                               log file access
C                               Check VAXUP when sending to VAX
C                               Update MCVFLG in file when an
C                               entry is sent to the VAX successfully
C       Rev. 15-Feb-85 MPK      Use YESNO to ask yes/no questions
C       Rev. 10-Jul-85 MPK      Adapt to new log format
C                               In option for listing entries not sent
C                               to VAX, do not list background entries
C       Rev. 04-Apr-86 WDR      Modified Micro VAX II
C       Rev. 02-May-90 WDR      Removed COMM calls for Pu238 instrument
C
C*****************************************************************************
C
      SUBROUTINE LMCLOG(AUTO)
C
      CHARACTER*1 INPDTE(9)
      INTEGER*2 NANS,ENTNUM
      INTEGER*2 LPTR,FPTR,IPTR,INTPTR(100)
      LOGICAL*1 VAXFLG,DATFLG,STRDFL
      LOGICAL*1 AUTO,YESNO
C
C  The following common block files are generic for all instruments
```

```
C
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CLOG.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CMC.CMN'
      INCLUDE 'CMONTH.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CMEAS.CMN'
C
C  Open log file MCLOGF for reading
C
      IERR=1
      OPEN (UNIT=DKLUN,NAME=MCLOG,TYPE='OLD',ACCESS='DIRECT',
     1       RECORDSIZE=MCREC, ERR=90)
C
C  Read first record to get start and end pointers
C
  150 IERR=2
      READ (DKLUN'1,ERR=90) LPTR,FPTR
      IERR=5
      IF (LPTR.LT.2 .OR. LPTR.GT.MCMAX) GO TO 90
      IF (FPTR.LT.2 .OR. FPTR.GT.MCMAX) GO TO 90
      IF (AUTO) GO TO 350
C
C  Message to operator to select a listing option
C
      CALL LGMENU(NANS,'Measurement control log')
      IF (NANS.EQ.0) GO TO 500
      VAXFLG=.FALSE.
      DATFLG=.FALSE.
C
C  Option 201 goes to entries not yet sent to VAX
C  Option 202 goes to entries between two dates
C  Option 203 goes to all entries
C  Option 204 goes to N entries
C
      GO TO (201,202,203,204)NANS
C
C  List entries not yet sent to VAX
C
  201 VAXFLG=.TRUE.
      GO TO 203
C
C  List entries between two dates
C
  202 DATFLG=.TRUE.
C
C  Set flag for start date and get start date
C
  800 STRDFL=.TRUE.
      TYPE 1003
 1003 FORMAT(' '/'$ Enter start date (most recent)  [09-AUG-84] -> ')
      READ (5,1004,END=800,ERR=800) NCHAR,INPDTE
 1004 FORMAT(Q,9A)
      IF (NCHAR.EQ.0) GO TO 150
      CALL JULDAT(INPDTE,NOWTIM,SDAT,xTIM)
```

```
        GO TO 820
C
C  Reset flag for start date and get stop date
C
  810   STRDFL=.FALSE.
        TYPE 1005
 1005   FORMAT(' '/'$ Enter stop date (oldest)  [02-AUG-84] -> ')
        READ (5,1004,END=810,ERR=810) NCHAR,INPDTE
        IF (NCHAR.EQ.0) GO TO 150
        CALL JULDAT(INPDTE,NOWTIM,EDAT,XTIM)
C
C  Check the start or stop date for validity,
C  using the numbers JULDAT saves in IYEAR, IMONTH, and IDAY
C
  820   IF (IYEAR.GE.84 .AND. IYEAR.LE.99) GO TO 830
        TYPE 1020
 1020   FORMAT (//20X,'**** INCORRECT YEAR NUMBER ****'/)
        GO TO 850
  830   IF (IMONTH.GE.1 .AND. IMONTH.LE.12) GO TO 840
        TYPE 1021
 1021   FORMAT (//20X,'**** INCORRECT MONTH NAME ****'/)
        GO TO 850
  840   IF (IDAY.GE.1 .AND. IDAY.LE.31) GO TO 860
        TYPE 1022
 1022   FORMAT (//20X,'**** INCORRECT DAY NUMBER ****'/)
  850   IF (STRDFL) GO TO 800
        GO TO 810
  860   IF (STRDFL) GO TO 810
        IF (SDAT.GE.EDAT) GO TO 203
        TYPE 1023
 1023   FORMAT (//20X,'**** START DATE IS EARLIER THAN STOP DATE ****'/)
        GO TO 800
C
C  List N entries
C
  204   TYPE 1006,MCMAX-1
 1006   FORMAT(/'$ How many entries do you want listed ? ('
        1       I3,' maximum) -> ')
        READ (5,1000,END=205,ERR=205) N
 1000   FORMAT (I4)
        IF (N .EQ. 0) GO TO 150
        IF (N .GE. 1 .AND. N .LT. MCMAX) GO TO 220
C
C  Error message for improper input
C
  205   TYPE 1009
 1009   FORMAT(' '/
        1   '  **** ANSWER WAS NON-NUMERIC, NEGATIVE, OR TOO LARGE ****')
        GO TO 204
C
C  List all entries
C
  203   N = MCMAX - 1
C
C  Output header for listing
C
  220   CALL MCHEAD
C
C  Main loop for listing entries
C
```

```
        LGNUM = 1
        IPTR=LPTR
C
        DO 300 I = 1, N
        IERR=2
        READ (DKLUN'IPTR,ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
        2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
        IF (.NOT.DATFLG) GO TO 230
        CALL JULDAT(ASDATE,ASTIME,XDAT,XTIM)
        IF (XDAT.GT.SDAT) GO TO 240
        IF (XDAT.LT.EDAT) GO TO 400
  230   IF ( VAXFLG .AND. (MCVAX .OR. MTYP(1).NE.'M') ) GO TO 240
        WRITE (LOUT,1002) LGNUM,
        1       ASDATE,ASTIME,
        2       OPERID,MTYP,VALUE,SIGMA,VAR,
        3       MCRUNS,MCVAX,MCERR
 1002   FORMAT (I4,1X,A6,1X,A5,1X,A8,1X,2A,1PE13.5,
        1       1PE11.3,E14.6,I3,1X,L,1X,L)
        INTPTR(LGNUM) = IPTR
        LGNUM = LGNUM + 1
  240   IF (IPTR.EQ.FPTR) GO TO 400
        IPTR=IPTR-1
        IF (IPTR.LT.2) IPTR=MCMAX
  300   CONTINUE
C
C  If the line printer is open, close it in order to finish printing
C
  400   IF(LOUT .EQ. 6)CLOSE(UNIT = LOUT)
C
   Prompt for sending entries to VAX
C
C       IF (.NOT.YESNO('Send entry to VAX')) GO TO 150
C       IF (YESNO('All files not yet sent')) GO TO 350
  380   TYPE 1014
 1014   FORMAT(' '/'$ Type number of entry (RETURN to quit)=> ')
        READ (5,1000,END=380,ERR=380) ENTNUM
        IF (ENTNUM.EQ.0) GO TO 150
C
C  Check if number entered is in the range of entries listed
C
        IF (ENTNUM .GE. 1 .AND. ENTNUM .LT. LGNUM) GO TO 390
C
C  Error message for number less than one or
C  greater than maximum entries listed
C
        TYPE 1015,ENTNUM
 1015   FORMAT(/' **** ENTRY NUMBER',I4,' WAS NOT LISTED ****'/)
        GO TO 380
C
C  Read the entry from the file and send it to the VAX
C
  390   IERR=2
        READ (DKLUN'INTPTR(ENTNUM),ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
        2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
        IF (MTYP(2).NE.'B') GO TO 391
C       CALL COMM('RB')                  !*communications - log bias
```

```
        GO TO 393
  391   IF (MTYP(2).NE.'P') GO TO 392
C       CALL COMM('RP')                    !*communications - log precision
  393   IF (.NOT.VAXUP) GO TO 150
        IF (MCVAX) GO TO 380
        MCVAX=.TRUE.
        IERR=3
        WRITE (DKLUN'INTPTR(ENTNUM),ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
        2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
        GO TO 380
  392   TYPE 1030,MTYP
 1030   FORMAT (//20X,'**** ENTRY TYPE ',2A1,
        1       ' CANNOT BE SENT TO VAX ****'/)
        GO TO 380
C
C  All entries not sent to VAX yet will be sent here
C
  350   IPTR=LPTR
C
  355   IERR=2
        READ (DKLUN'IPTR,ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
        2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
        IF (MCVAX) GO TO 361
        IF (MTYP(2).NE.'B') GO TO 363
C       CALL COMM('RB')                    !*communications - log bias
        GO TO 364
  363   IF (MTYP(2).NE.'P') GO TO 361
        CALL COMM('RP')                    !*communications - log precision
  364   IF (VAXUP) GO TO 362
        IF (AUTO) GO TO 500
        GO TO 150
  362   MCVAX=.TRUE.
        IERR=3
        WRITE (DKLUN'IPTR,ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
        2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
  361   IF (IPTR.EQ.FPTR) GO TO 365
        IPTR=IPTR-1
        IF (IPTR.LT.2) IPTR=MCMAX
        GO TO 355
  365   IF (AUTO) GO TO 500
C
C  Check if more are to be sent
C
        IF (YESNO('Send more')) GO TO 380
        GO TO 150
C
C  ** Log file handling errors **
C
   90   CALL ERRMSG(IERR,MCLOG)
C
   Clean up and exit
C
  500   CLOSE (UNIT=DKLUN,ERR=501)
  501   RETURN
```

END

```
      SUBROUTINE LODGRP

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     Sub to load a specified peak grouping and remove its background.
      Optionally, fix any specified energy & intensity relationships
C     between peaks. In the case of single peaks specified for simple
C     integration only, sum around the centroid and return.
C             By J.B.Niday    Edited by R. Gunnink  1-June-84
C             Last edit 13-Sep-85 (R.G.) PKHT in DO 200 loop.
C             Last edit 17-Jun-87 (J.B.N.)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'

      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CLUNS.CMN'

C     Check for size of group (EDIGRP cannot do)
      OFFSET = IST - 1.0
      NDPTS = IEND-IST+1
      CHANS = NDPTS
      IF(NDPTS .GT. MXDP) THEN
             TYPE 1002,KTGRP,IST,IEND,MXDP
             EST = 0.
             GO TO 400
      ENDIF
1002   FORMAT (/' Group',I3,', (channels',I5,' to',I5,') exceeds limit',
     +  ' of',I5,' data points--REJECT IT!'//)

C *** Run-time modification of shape params that change with energy
      GSQ = GAIN * GAIN
      IF(NUSHPS .NE. 0)GO TO 13
      IF (SHAPC(1) .EQ. 0)GO TO 10    !Flag for SHAPC file
      AVEN = 0.5*(IST+IEND) * GAIN + ZERO
      SG = (SHAPC(1) + SHAPC(2)*EST)/GSQ +.462
      FWHM = SQRT(SG) * GAIN
      ATAIL = EXP(SHAPC(3) + AVEN * SHAPC(4))
      BTAIL = SHAPC(5) * GAIN
      EXP3 = SHAPC(7)
      EXP4 = SHAPC(8) * GAIN
      GO TO 16

10    IF (INTFLG .NE. 0) GO TO 13
      EST = 0.          !Set to abort the group!
      TYPE 1010,KTGRP
1010  FORMAT(' *** Cannot fit peaks without some shape',
     +  ' information!!!'/'     Abort group #',I4)
      IF(IOFLG(3) .EQ. 0) WRITE(LOUT,1010)KTGRP
      GO TO 400


13    IF(SHAPC(1) .EQ. 0) SHAPC(2) = .0017
C      Set a default for slope of peak width parameter.
      POS = NDPTS/2.  !For FWHM at mid-energy of grouping.
      IF (NP .EQ. 1)  POS = EN(1)/GAIN - OFFSET  !At peak position
      SG = FWHM**2/GSQ - SHAPC(2)*POS + .462
      BTAIL = TSHP5 * GAIN
      EXP4 = TSHP8 * GAIN
```

```
16      ALPHA = -2.7726/SG
        EXP1 = ATAIL
        EXP2 = BTAIL

        DO 19, I=1,NP
        Convert intrinsic widths from eV to channel values.
        GAMA(I) = GAMA(I) * .001/GAIN
19      CONTINUE

C       Determine peak grouping location and read in data,after first
C       determining the backgrounds to be subtracted.
C       Background areas were specified for channels adjacent to the
C       group data, somewhat separated ("remote"), or both. In the latter
C       case, the slope for extrapolation is to be determined.

        XTRAPL = 0      !Store extapolation dist. for remote cases.
        XTRAPH = 0
        NUMR = 0
        NUMS = BKLO / GAIN + 0.5        !Note: These values may be zero.
        NUME = BKHI / GAIN + 0.5

        IF((NUMS .LE. 0) .AND. (BKLO .NE. 0)) NUMS = 1
C       Modified to protect against high gain cases (J.B.N. 24-Jan-85)
        IF((NUME .LE. 0) .AND. (BKHI .NE. 0)) NUME = 1

C       Convert bkgrd slopes to fractions/channel.
        SLPLO = 0.01 * SLP(1) * GAIN    !Note: SLP may be zero
        SLPHI = 0.01 * SLP(2) * GAIN

C       Detn. average background level in front of peak grouping.
C       Correct sloping backgd to channel nearest data points

        IF(RBKEN(1) .NE. 0) GO TO 30
        ISTLO= IST-NUMS         !No remote background used on low E side.

        CALL RDCHNS(Y,ISTLO,NUMS,INPDEV)

        CALL AVE (1,NUMS,SM,AVBG1,Y)

        TLS = EXP4/GAIN
        IF (EN(1) .LT. 4500.) GO TO 36
        IF ((TLS .LT. 0.1).AND.(TLS .GT. 0.)) AVBG1 = 0.01  !Assume Alpha Spectr

        GO TO 36

C       A remote low E bkgrd has been specified.
30      ISTLO = ((RBKEN(1) - ZERO) / GAIN) + 0.5
        NUMR = RBKWID(1) /GAIN + 0.5
        NPTS = IST - ISTLO

        CALL RDCHNS (Y,ISTLO,NPTS,INPDEV)

        CALL AVE(1,NUMR,SM,AVBGLO,Y)

        IF(BKLO .NE. 0) THEN            !Both bkgds given; calc. slope
                CALL AVE(NPTS-NUMS,NPTS,SM,AVBG1,Y)
                ADIST = NPTS - NUMR/2. - NUMS/2.
                SLPLO = (AVBG1-AVBGLO) / (AVBGLO * ADIST)
                SLP(1) = SLPLO / (.01*GAIN)     !Preserve SLP(1) for printout in
        ELSE
```

```
                XTRAPL = NPTS - NUMR/2. !Remote bkgrd only!
                BGLO = AVBGLO * (1. + SLPLO * XTRAPL)
                SLPLO = SLPLO * AVBGLO  !Finally, convert to cts/chan
                NUMS = NUMR     !To fix BPTS & carry forward for OUT
                GO TO 40
        ENDIF

36      BGLO = AVBG1 * (1. + SLPLO * NUMS/2.)
        SLPLO = SLPLO * AVBG1   !Finally, convert to cts/chan

C       Now detn avg. bkgd. level on high E side of group
40      NUMR = 0
        IF(RBKEN(2) .NE. 0) GO TO 43

        CALL RDCHNS(Y,IEND+1,NUME,INPDEV)

        CALL AVE (1,NUME,SM,AVBG2,Y)
        GO TO 48

C       A remote bkgrd specified on high E side.
43      ISTHI = ((RBKEN(2) - ZERO) / GAIN) + 0.5
        NUMR = RBKWID(2) /GAIN + 0.5
        NDXHR = ISTHI - IEND
        NPTS = NUMR + NDXHR - 1

        CALL RDCHNS(Y,IEND+1,NPTS,INPDEV)

        CALL AVE(NDXHR,NPTS,SM,AVBGHI,Y)

        IF(BKHI .NE. 0) THEN
                CALL AVE(1,NUME,SM,AVBG2,Y)     !Both bgds used
                ADIST = NPTS - NUMR/2. - NUME/2.
                SLPHI = (AVBGHI - AVBG2) / (AVBGHI*ADIST)
                SLP(2) = SLPHI / (.01 * GAIN)
        ELSE

                XTRAPH = NPTS - NUMR/2. !Remote bkgd only!
                BGHI = AVBGHI * (1. - SLPHI * XTRAPH)
                SLPHI = SLPHI * AVBGHI
                NUME = NUMR
                GO TO 50
        ENDIF

48      BGHI = AVBG2 * (1. - SLPHI * NUME/2.)
        SLPHI = SLPHI * AVBG2


C       Remove background continuum from peak grouping; return net cts.

50      CALL RDCHNS(Y,IST,NDPTS,INPDEV)
52      CALL BKGRD(1,NDPTS,BGLO,BGHI,SLPLO,SLPHI,NPTS,SMY,Y,YNET)
C       TYPE *,'BGLO',BGLO,BGHI,SLPLO,SLPHI
        BPTS = NUMS + NUME
        AVEBG = (BGLO + BGHI) / 2.
        IF(SMY .LT. 0)SMY = 0.
        ER= SQRT(SMY + AVEBG*CHANS*(1. + CHANS/BPTS))
        ERR(1) = ER     !Used only if SMY equal to 0.0
        ER = ER / (SMY+1.) * 100.
        IF(ER .GT. 100.) SMY = 0.0      !Less than 50% confidence.
        IF(SMY .NE. 0)ERR(1) = ER
```

```
      PKPO = (EN(1) - REFEN)/GAIN + REFCH

      IF((SMY.EQ.0.) .OR. (ERR(1).GE.50)) THEN
            IF(INTFLG .NE. 0)GO TO 92       !Scheduled for integration.
            WRITE (LOUT,1068)KTGRP,ERR(1),SMY
            IF(IOFLG(3) .EQ. 0)WRITE (LOUT,1068)KTGRP,ERR(1),SMY
            GOTO 92                  !Even though area was to be fitted.
      ENDIF
 1068 FORMAT(/' !!    For group #',I3,' Error=',F9.3,' SumY=',F10.3/
     +   '  Will skip fitting & just integrate group!')

      IF(INTFLG .EQ. 0) GO TO 100     !Peak scheduled to be fitted.

C     C       C       C       C       C       C       C       C
C      Simply locate peak for integration.
      CALL MAXVAL(1,NDPTS,JPK,YMAX,YNET)
      IF((JPK .EQ. 1) .OR. (JPK .EQ. NDPTS))GO TO 80
      Y2 = 0.5 * YMAX
      M = 0
      DO 70, J = JPK,NDPTS
      M = M + 1
      IF(Y(JPK+M) .LE. Y2)GO TO 75
70    CONTINUE
      GO TO 80
75    IF((JPK-M) .GE. 1)GO TO 85
80    PKPO = JPK
      TYPE 1080,KTGRP
1080  FORMAT(' For group',I3,' peak position will be inexact')
      GO TO 90

      CALL GFIT(JPK,M,ALPHA,ATAIL,BTAIL,PKPO,PHT,ER,YNET)
      FWHM = GAIN* SQRT(-2.7726/ALPHA)
      IF(ER .EQ. -2.)GO TO 80

C     Assume slow variation within group of any non-linearity effect.
90    PKPO = PKPO + OFFSET

92    INTFLG = 1       !Turns on flag, in case peak scheduled for fitting.
      IF (LONGPR)WRITE (LOUT,1090)IST,IEND,PKPO,SMY,ERR(1)
1090  FORMAT(' By simple integration of region:',I5,' to',I5,/9X,
     + 'Peak chan =',F8.2,'; Net counts  =',F12.0,' +-',F7.2,'%')

C     Transfer results of integration now to simplify storage in OUT
      PKPOS(1) = PKPO
      SUM(1) = SMY
      ENJ(1) = REFEN + (PKPO-REFCH) * GAIN
      GO TO 400

C     C       C       C       C       C       C       C       C       C

100   DO 200 J = 1,NP
      PKPOS(J) = (EN(J) - REFEN)/ GAIN + REFCH - OFFSET
      NPK = PKPOS(J) + 0.5
      PKHT(J) = YNET(NPK)
      IF(HIGHT(J).LE.0.)  GOTO 102
      PKHT(J) = HIGHT(J) / SQRT(-3.14/ALPHA)
      WL = GAMA(J) * SQRT(-ALPHA)
      IF (WL .GT. 0.)  PKHT(J) = PKHT(J)*(1.-.5642*WL+.25*WL**2)
      GOTO 200
C     Adjust relative intensities for local variation of shapes
```

```
102     K = -NYFLG(J)
        S1 = SHAPC(1)
        IF(S1 .EQ. 0.)S1 = SG * GSQ
        HIGHT(J)=HIGHT(J)*SQRT((S1+SHAPC(2)*EN(K))/(S1+SHAPC(2)*EN(J)))
        RATIO = 0.0
        IF ((GAMA(K) .EQ. 0.) .AND. (GAMA(J) .EQ. 0.))  GOTO 200
        IF ((GAMA(K) .GT. 0.) .AND. (GAMA(J) .GT. 0.))  GOTO 200
C       X ray and Gamma Ray are being related.  Make height adjustment.
        WL = (GAMA(J)+GAMA(K))*SQRT(-ALPHA)  !One of GAMA's = 0.0
        RATIO = 1.-.5642*WL +.25*WL**2  !X ray/Gamma peak height ratio.
        IF (GAMA(K) .GT. 0.0)  RATIO = 1. / RATIO
        HIGHT(J) = HIGHT(J) * RATIO
200     CONTINUE
        FPKHT=0.


C       FIX THE INTENSITY RELATIONSHIPS THAT HAVE BEEN SPECIFIED
C       IN THE INPUT.  FIRST READJUST PEAK HEIGHT
C       ESTIMATES FOR INTERFERENCES FROM NEIGHBORING PEAKS.

        DO 250 J=1,NP
        IF(NYFLG(J) .GT. 0)GO TO 250
        IF(HIGHT(J) .GE. 0)GO TO 250
        K = -NYFLG(J)
        DX = PKPOS(J) - PKPOS(K)
        FCT = -HIGHT(J) * EXP(ALPHA * DX**2)
        IF (HIGHT(K) .LE. 0.)  PKHT(K) = PKHT(K) / (1. + FCT)
        PKHT(J) = -HIGHT(J) * PKHT(K)
250     CONTINUE

        DO 300 J = 1,NP
        IF(NYFLG(J) .LT. 0)GO TO 300
        L = J - 1
        CONST = 0.
        IF(L .LE. 0)GO TO 290
C       Is neighbor related?
270     IF(NYFLG(L) .GE. 0)GO TO 280
        IF(-NYFLG(L) .EQ. J)GO TO 290
280     DX = PKPOS(J) - PKPOS(L)
        EXPN = EXP(ALPHA * DX**2)
        PKHT(J) = PKHT(J) - PKHT(L) * EXPN *(1.- CONST * EXPN)
290     IF(L .GT. J)GO TO 300
        CONST = 0.5
        L = J + 1
        IF(L .LE. NP)GO TO 270
300     CONTINUE

C       FIX THE ENERGY RELATIONSHIPS THAT HAVE BEEN
C       SPECIFIED IN THE INPUT.

        DO 390 J =1,NP
        IF(NXFLG(J) .GE. 0)GO TO 390
        L = -NXFLG(J)
        PKPOS(J) = PKPOS(L) -(EN(L) - EN(J))/GAIN
390     CONTINUE
        ITER = 1

400     RETURN
        END
```

```
C*****************************************************************************
C
C  LSLOG8        (LSLOG subroutine for Pu-238 isotopic assay system.
C                Includes listing of isotopic abundances.)
C
      List the assay log entries
      Options for listing:
              All entries
              'N' entries
              Entries not yet sent to VAX
              Entries within a pair of dates
      Entries are listed from current date

      Calling argument:
      AUTO    If .FALSE., the routine runs in interactive mode
              and prompts for listing options.
              If .TRUE., there is no operator interaction or listing;
              all entries not yet sent to the VAX are sent.

      CMS 14-MAY-84

      Rev 27-SEP-84 mjc
      Rev 25-sep-84 cms
      Rev 18-OCT-84 MJC       closed LOUT if it equaled 6 (line printer)
      Rev 25-Jan-85 MPK       Remove date syntax checking, and
                              use JULDAT to check dates for
                              correctness
                              Update communications to use pseudo-
                              message type 'LA' (log assay)
                              Allow multiple listings on a single
                              entry
                              Add end and error exits to all
                              terminal input
                              Allow escaping back to menu from
                              most input prompts
      Rev 28-Jan-85 MPK       Correct and clean up indexing and
                              listing logic
      Rev 29-Jan-85 MPK       Add automatic mode for sending
                              entries to VAX
                              Add additional error cnecks for
                              log file access
                              Check VAXUP when sending to VAX
                              Update LGFLAG in file when an
                              entry is sent to the VAX successfully
      Rev 15-Feb-85 MPK       Use YESNO to ask yes/no questions
      Rev 20-Feb-85 CMS       Added error bar and assay type variables
                              to log entries
      Rev 27-Feb-85 MPK       Check that log entry has proper type
                              before sending to VAX
      Rev 20-Mar-85 MPK       Correct interchange of variables
                              in write statement
      Rev 08-Jul-85 MPK       Adapt to new log format
      Rev 10-Jul-85 MPK       In option for listing entries not
                              sent to VAX, list only "AS" entry type
      Rev 19-Sep-85 WDR       Added isotopic abundances to the output
      Rev 04-Apr-86 WDR       Modified for Micro VAX II
      Rev 13-Jul-87 RDP       Modified for uVAX II
                              Convert BYTE strings to CHARACTER strings
      Rev 20-Mar-90 WDR       Modified for Pu-238 system; changed
                              LVFLAG to C240FG
```

```
C       Rev 02-May-90 WDR       Removed COMM calls for Pu238 instrument
C
C*************************************************************************
C
      SUBROUTINE LSLOG(AUTO)

      CHARACTER*1 INPDTE(9)
      CHARACTER*8 SMPID
      INTEGER*2 NANS,ENTNUM
      INTEGER*2 LPTR,FPTR,IPTR,INTPTR(100)
      LOGICAL*1 VAXFLG,DATFLG,STRDFL,NAMFLG
      LOGICAL*1 AUTO,YESNO,CYCBUF(3)
C
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CLOG.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CMONTH.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CVAX.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CABUND.CMN'
C
C  Open log file ASYLOG for reading
C
      IERR=1
      OPEN (UNIT=DKLUN,NAME=ASLOG,TYPE='OLD',ACCESS='DIRECT',
     1       RECORDSIZE=LOGREC,ERR=90)
C
C  Read first record to get start and end pointers
C
  150 IERR=2
      READ (DKLUN'1,ERR=90) LPTR,FPTR
      IERR=5
      IF (LPTR.LT.2 .OR. LPTR.GT.MAXASY) GO TO 90
      IF (FPTR.LT.2 .OR. FPTR.GT.MAXASY) GO TO 90
      IF (AUTO) GO TO 350
C
C  Message to operator to select a listing option
C
      CALL LGMENU(NANS,'Assay log')
      IF (NANS.EQ.0) GO TO 500
      VAXFLG=.FALSE.
      DATFLG=.FALSE.
      NAMFLG=.FALSE.
C
C  Option 201 goes to entries not yet sent to VAX
C  Option 202 goes to entries between two dates
C  Option 203 goes to all entries
C  Option 204 goes to N entries
C
      GO TO (201,202,203,204)NANS
C
C  List entries WITH SPECIFIC SAMPLE ID
C
```

```
201     TYPE 1101
1101    FORMAT(' '/'$ Enter sample ID to search for ->')
        READ (5,1102,END=800,ERR=800)SMPID
1102    FORMAT(A)
        IF (SMPID.EQ.' ')GOTO 150
        NAMFLG = .TRUE.
        GO TO 203
C
C  List entries between two dates
C
 202    DATFLG=.TRUE.
C
C  Set flag for start date and get start date
C
 800    STRDFL=.TRUE.
        TYPE 1003
1003    FORMAT(' '/'$ Enter start date (most recent)  [09-AUG-84] -> ')
        READ (5,1004,END=800,ERR=800) NCHAR,INPDTE
1004    FORMAT(Q,9A)
        IF (NCHAR.EQ.0) GO TO 150
        CALL JULDAT(INPDTE,NOWTIM,SDAT,XTIM)
        GO TO 820
C
C  Reset flag for start date and get stop date
C
 810    STRDFL=.FALSE.
        TYPE 1005
1005    FORMAT(' '/'$ Enter stop date (oldest)  [02-AUG-84] -> ')
        READ (5,1004,END=810,ERR=810) NCHAR,INPDTE
        IF (NCHAR.EQ.0) GO TO 150
        CALL JULDAT(INPDTE,NOWTIM,EDAT,XTIM)

C  Check the start or stop date for validity,
C  using the numbers JULDAT saves in IYEAR, IMONTH, and IDAY
C
 820    IF (IYEAR.GE.84 .AND. IYEAR.LE.99) GO TO 830
        TYPE 1020
1020    FORMAT (//20X,'**** INCORRECT YEAR NUMBER ****'/)
        GO TO 850
 830    IF (IMONTH.GE.1 .AND. IMONTH.LE.12) GO TO 840
        TYPE 1021
1021    FORMAT (//20X,'**** INCORRECT MONTH NAME ****'/)
        GO TO 850
 840    IF (IDAY.GE.1 .AND. IDAY.LE.31) GO TO 860
        TYPE 1022
1022    FORMAT (//20X,'**** INCORRECT DAY NUMBER ****'/)
 850    IF (STRDFL) GO TO 800
        GO TO 810
 860    IF (STRDFL) GO TO 810
        IF (SDAT.GE.EDAT) GO TO 203
        TYPE 1023
1023    FORMAT (//20X,'**** START DATE IS EARLIER THAN STOP DATE ****'/)
        GO TO 800
C
C  List N entries
C
 204    TYPE 1006,MAXASY-1
1006    FORMAT(/'$ How many entries do you want listed ? (',
        1       I3,' maximum) -> ')
        READ (5,1000,END=205,ERR=205) N
```

```
 1000    FORMAT (I4)
         IF (N .EQ. 0) GO TO 150
         IF (N .GE. 1 .AND. N .LT. MAXASY) GO TO 220
C
C  Error message for improper input

  205    TYPE 1009
 1009    FORMAT(' '/
         1   '  **** ANSWER WAS NON-NUMERIC, NEGATIVE, OR TOO LARGE ****')
         GO TO 204
C
C  List all entries
C
  203    N = MAXASY - 1
C
C  Output header for listing
C
  220    CALL LHEAD
C
C  Main loop for listing entries
C
         LGNUM = 1
         IPTR=LPTR
C
         DO 300 I = 1, N
         IERR=2
         READ (DKLUN'IPTR,ERR=90)
         1        ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
         2        LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
         3        BIASOK, PRECOK, BACKOK,    BAD, CONDIT, C240FG,
         4        (WTPCT(J),      PCT(J),J=1,6)
         IF (.NOT.DATFLG) GO TO 228
         CALL JULDAT(ASDATE,ASTIME,XDAT,XTIM)
         IF (XDAT.GT.SDAT) GO TO 240
         IF (XDAT.LT.EDAT) GO TO 400
228      IF (.NOT.NAMFLG)GOTO 230
         IF (SMPID.EQ.SAMPID)GOTO 232
         GOTO 240
  230    IF ( VAXFLG .AND.  (ASTYPE(1).NE.'A') ) GO TO 240
         ENCODE (3,1001,CYCBUF) NUMCYC
 1001    FORMAT (I3)
  231    IF (CYCBUF(1).NE.' ') GO TO 232
         CYCBUF(1)=CYCBUF(2)
         CYCBUF(2)=CYCBUF(3)
         CYCBUF(3)=' '
         GO TO 231
  232    WRITE (LOUT,1002) LGNUM,
         1        ASDATE,ASTIME,
         2        SAMPID,OPERID,LGFILE,
         3        ICYCLE,CYCBUF,ASANS,C240FG,BAD,(237+IJ,WTPCT(IJ),IJ=1,2),
         4        WTPCT(5),wtpct(3),WTPCT(6),wtpct(4),ASANS
1002     FORMAT (I4,'.',A9,1X,A5,1X,A8,1X,A8,1X,A13,I4,'/',3A,
         1        1PE10.3,4X,L,2X,L,/,4X,2(' %',I3,'=',0PF7.4),
         2        ' %240=',F7.4,' %241=',F7.4,' %242=',F7.4,' %Am=',F7.4,/,
         3        4X,'Specific power =',1PE10.3)
         INTPTR(LGNUM) = IPTR
         LGNUM = LGNUM + 1
  240    IF (IPTR.EQ.FPTR) GO TO 400
         IPTR=IPTR-1
         IF (IPTR.LT.2) IPTR=MAXASY
```

```
  300   CONTINUE
C
C  If the line printer is open, close it in order to finish printing
C
  400   IF(LOUT .EQ. 6)CLOSE(UNIT = LOUT)

C  Prompt for sending entries to VAX
C
        GOTO 150
C       IF (.NOT.YESNO('Send entry to VAX')) GO TO 150
C       IF (YESNO('All files not yet sent')) GO TO 350
  380   TYPE 1014
 1014   FORMAT(' '/'$ Type number of entry (RETURN to quit) ->  ')
        READ (5,1000,END=380,ERR=380) ENTNUM
        IF (ENTNUM.EQ.0) GO TO 150
C
C  Check if number entered is in the range of entries listed
C
        IF (ENTNUM .GE. 1 .AND. ENTNUM .LT. LGNUM) GO TO 390
C
C  Error message for number less than one or
C  greater than maximum entries listed
C
        TYPE 1015,ENTNUM
 1015   FORMAT(/' **** ENTRY NUMBER',I4,' WAS NOT LISTED ****'/)
        GO TO 380
C
C  Read the entry from the file and send it to the VAX
C
  390   IERR=2
        READ (DKLUN'INTPTR(ENTNUM),ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
        2       LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,    BAD, CONDIT, C240FG,
        4       (WTPCT(I),      PCT(I),I=1,6)
        IF (ASTYPE(1).NE.'A') GO TO 392
C       CALL COMM('RA')                    !*communications - log assay
        IF (.NOT.VAXUP) GO TO 150
C       IF (LVFLAG) GO TO 380
C       LVFLAG=.TRUE.
        IERR=3
        WRITE (DKLUN'INTPTR(ENTNUM),ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
        2       LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,    BAD, CONDIT, C240FG,
        4       (WTPCT(I),      PCT(I),I=1,6)
        GO TO 380
  392   TYPE 1030,ASTYPE
 1030   FORMAT (//20X,'**** ENTRY TYPE ',2A1,
        1       ' CANNOT BE SENT TO VAX ****'/)
        GO TO 380
C
C  All entries not sent to VAX yet will be sent here
C
  350   IPTR=LPTR
C
  355   IERR=2
        READ (DKLUN'IPTR,ERR=90)
        1       ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
        2       LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
```

```
     3       BIASOK, PRECOK, BACKOK,     BAD, CONDIT, C240FG,
     4       (WTPCT(I),       PCT(I),I=1,6)
C    IF (LVFLAG) GO TO 361
     IF (ASTYPE(1).NE.'A') GO TO 361
C    CALL COMM('RA')                    !*communications - log assay
     IF (VAXUP) GO TO 362
     IF (AUTO) GO TO 500
     GO TO 150
C362 LVFLAG=.TRUE.
 362 CONTINUE
     IERR=3
     WRITE (DKLUN'IPTR,ERR=90)
     1       ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
     2       LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
     3       BIASOK, PRECOK, BACKOK,     BAD, CONDIT, C240FG,
     4       (WTPCT(I),       PCT(I),I=1,6)
 361 IF (IPTR.EQ.FPTR) GO TO 365
     IPTR=IPTR-1
     IF (IPTR.LT.2) IPTR=MAXASY
     GO TO 355
 365 IF (AUTO) GO TO 500
C
C  Check if more are to be sent
C
     IF (YESNO('Send more')) GO TO 380
     GO TO 150
C
C  ** Log file handling errors **
C
  90 CALL ERRMSG(IERR,ASLOG)
C
   Clean up and exit
C
 500 CLOSE (UNIT=DKLUN,ERR=501)
 501 RETURN
     END
```

```
C*****************************************************************
C  MCBIAS
C       Measurement control check for instrument bias
C                  Generic package
C
        05-Jun-84 lc
C
C       Rev. 07-Jun-84  lc
C       REV. 05-Sep-84  TES
C       REV. 25-Sep-84  CMS
C       Rev. 08-Jan-85  MPK     Move call to VAX comm and add
C                               call to DATTIM so that VAX
C                               gets start time
C       Rev. 18-JAN-85  MPK     Add calls to ADATE and AEVAX
C                               remove reference to VAXCOM
C       Rev. 22-Jan-85  MPK     Check VAXUP in COMM calls
C       Rev. 12-Feb-85  MPK     Redo logical flags
C       Rev. 15-Feb-85  MPK     Use YESNO to ask yes/no questions
C       Rev. 20-Feb-85  MPK     Changed BIACT=2.58 to BIACT=3.00
C       Rev. 07-Mar-85  MPK     Included CVAX common
C       Rev. 04-Apr-86  WDR     Modified for Micro VAX II
C       Rev. 02-May-90  WDR     Removed COMM and AEVAX for Pu238
C                               instrument
C
C*****************************************************************
C
        SUBROUTINE MCBIAS
C
        LOGICAL*1 SUMMRY,YESNO
        REAL*8 WORD
C
   The following common block files are generic for all instruments
C
        INCLUDE 'CMCONS.CMN'
        INCLUDE 'CMEAS.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CMC.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CVAX.CMN'
C
C
        DATA WORD /'  BIAS  '/
        MB = .TRUE.
C
C  Set constants for bias warning limit and action limit
C  HISTSD comes from the parameter file
C
        BIWARN = 1.96 * HISTSD
        BIACT  = 3.00 * HISTSD
        RTYP = 'MC Bias '
C
C  Initialize repeat count
C  Assay
C
        SUMMRY = .FALSE.
        ITIMES = 0
        ICYCLE = 1
        CALL DIALOG
```

```
      IF(ABORT) GO TO 500
C
C  Announce measurement control bias to VAX
C
      CALL ADATE
      IF (VAXUP)CALL COMM ('SB')       !*communications - start bias run
      CALL NEWNAM(0)
 5    CALL ASSAY
      IF(ABORT) GO TO 500
      VALUE = ASANS          ! Instrument specific variable here.
      SIGMA = ERRBAR
C
C  Initialize bias check pass flag
C  Get difference between measured value & reference standard value
C
      BIASOK = .TRUE.
      DIFF = VALUE - BIREF
      DIFF = ABS(DIFF)
C
C  Difference less than warning limit
C
      IF(DIFF .LT. BIWARN) GO TO 250
C
C  Difference equals or exceeds warning limit
C  but is less than action limit
C
      IF(DIFF .GE. BIACT) GO TO 200
      CALL ERRMSG(77,' BIAS WARNING ')
      GO TO 250
C
C  Difference equals or exceeds action limit

 200  BIASOK = .FALSE.
      CALL ERRMSG(78,' ')
      ITIMES = ITIMES + 1
C
 250  CALL PRINT1
      WRITE(LOUT,1200)VALUE,BIREF,DIFF
 1200 FORMAT(/' Current value = ',F10.4/' Reference value = ',F10.4/
     X        ' Difference = ',F10.4)
      IF(BIASOK .OR. ITIMES .GT. 1) GO TO 300
      IF (YESNO('Do you want to repeat bias check'))GO TO 5
C
C  Send to VAX
C  Write to measurement control log
C  IDCAL is to be defined for specific instrument
C
 300  CALL AEND
C300  CONTINUE
      VAR = BIREF
      MCRUNS = 1
      IF(.NOT. BIASOK) MCERR = .TRUE.
      MTYP(1) = 'M'
      MTYP(2) = 'B'
C     IF (VAXUP) CALL COMM('RB')       !*communications - bias results
      MCVAX = VAXUP
      CALL WTMCLG
C
500   RETURN
      END
```

```
C*****************************************************************************
C
C  MCCHKS       (MCCHK subroutine for solid isotopic instrument)
C
C       This subroutine checks that measurement control has been
C       performed within the required time limits and also checks the
C       status of the measurement control flag.  These checks are made
C       before an assay or an autocycle run.
C
C       TES  31-AUG-84
C
C       Rev 29-Nov-84 MPK       Replace list-directed with
C                               formatted I/O
C                               Simplify code by coalescing similar
C                               or identical code in different paths
C       Rev  5-Dec-84 MPK       Replace nulls in printing strings -
C                               nulls don't work because they terminate
C                               the message printout.
C       Rev 15-Jan-85 MPK       Move check for all three record types
C                               seen to where it is meaningful
C       Rev 15-JAN-85 WDR       Modified for solid isotopic system
C                               - removed background check code
C       Rev 15-Feb-85 MPK       Reorganize code so that it will not
C                               loop if an MC log entry is missing,
C                               and so that it catches situations
C                               where a given MC check has never
C                               been done.  The new strategy is to
C                               read through the entire MC log, and
C                               set flags to indicate whether a check
C                               is missing or expired, and whether it
C                               is out of limit.
C       Rev 20-Feb-85 MPK       Correct error typeout formats
C       Rev 06-Mar-85 MPK       Set flags to indicate status of
C                               meas control checks
C       Rev 04-Apr-86 WDR       Modified for Micro VAX II
C
C*****************************************************************************
C
C
      SUBROUTINE MCCHK
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CMC.CMN'
      INCLUDE 'CMCONS.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CASSAY.CMN'
C
C
      INTEGER*2 LPTR,FPTR,IPTR
      LOGICAL*1 LBKG, LPREC, LBIAS
      LOGICAL*1 LBKTIM, LPRTIM, LBITIM, LTIME
      LOGICAL*1 LBKERR, LPRERR, LBIERR
      LOGICAL*1 YESNO
C
C   Set local flags indicating whether background, precision, and bias
```

```
C   have been checked.
C   Also initialize flags for run expired, and run out of limit.
C   A run that is missing from the log will appear to be an expired run.
C
      LBKG  = .TRUE. !Set to TRUE for solid isotopic instrument
      LPREC  = .FALSE.
      LBIAS  = .FALSE.
      LBKTIM = .FALSE.!Set to FALSE for solid isotopic instrument
      LPRTIM = .TRUE.
      LBITIM = .TRUE.
      LBKERR = .FALSE.
      LPRERR = .FALSE.
      LBIERR = .FALSE.
C
C   Open measurement control log file and read first record to obtain
C   number of records in file.
C
      OPEN (UNIT=DKLUN,NAME=MCLOG,ACCESS='DIRECT',TYPE='OLD',
      1       RECORDSIZE=MCREC,ERR=105)
      GO TO 110
  105 CALL ERRMSG(1,MCLOG)              !** open error
      GO TO 650
  110 READ (DKLUN'1,ERR=115) LPTR,FPTR
      GO TO 120
  115 CALL ERRMSG(2,MCLOG)              !** read error
      GO TO 650
C
C  Check that number of records in file is within range
C
  120 IF (LPTR.LT.2 .OR. LPTR.GT.MCMAX) GO TO 125
      IF (FPTR.GE.2 .AND. FPTR.LE.MCMAX) GO TO 200
  125 CALL ERRMSG(5,MCLOG)
      GO TO 650
C
C  Convert the current date and time to "julian" days and seconds
C
  200 CALL JULDAT(NOWDAT, NOWTIM, XNDAT, XNTIM)
C
C  Read through the MC log looking for records of runs that have not
C  been checked yet.  Stop when all three run types (bias, precision,
C  and background) have been checked, or when the entire log has been
C  read.
C
      IPTR = LPTR
205   READ (DKLUN'IPTR,ERR=115)
      1       ASDATE, ASTIME, SAMPID, OPERID,  MTYP,  MCRUNS,
      2       VALUE, SIGMA, VAR, ARCHV1, ARCHV2, DGFLG,
      3       BIASOK, PRECOK, BACKOK, MCERR,  MCVAX
C
      IF (LBIAS) GO TO 210                  !bias already checked
      IF (MTYP(2).NE.'B') GO TO 210         !not a bias record
C
C  Check a bias record.  Save the error flag, and copy the time limit
C  for a bias run.  Also set the flag to remember that a bias record
C  was checked.
C
      LBIAS=.TRUE.
      LBIERR=MCEFLG
      TIMLIM=BITIME
      GO TO 230
```

```
C
  210   IF (LPREC) GO TO 220                    !precicion already checked
        IF (MTYP(2).NE.'P') GO TO 220           !not a precision record
C
C  Check a precision record.

        LPREC=.TRUE.
        LPRERR=MCEFLG
        TIMLIM=PRTIME
        GO TO 230
C
  220   IF (LBKG) GO TO 240                     !background already checked
        IF (MTYP(2).NE.'G') GO TO 240           !not a background record
C
C  Check a background record
C
        LBKG=.TRUE.
        LBKERR=MCEFLG
        TIMLIM=BKTIME
C
C  Common code to do measurement control checks.
C  Find the interval (in days) between the current time and the
C  time of the run in the log.  Compare this to the allowed time
C  interval, and set a flag for the appropriate run type to
C  indicate whether the run is current or expired.
C
  230   CALL JULDAT(ASDATE, ASTIME, XDAT, XTIM)
        DIF = (XNDAT+XNTIM/86400.) - (XDAT+XTIM/86400.)
        LTIME=DIF.GT.TIMLIM
        IF (MTYP(2).EQ.'B') LBITIM=LTIME
        IF (MTYP(2).EQ.'P') LPRTIM=LTIME
        IF (MTYP(2).EQ.'G') LBKTIM=LTIME
C
C  If all three MC types have been checked, stop looking
C
  240   IF (LBKG.AND.LPREC.AND.LBIAS) GO TO 300
        IF (IPTR.EQ.FPTR) GO TO 300
        IPTR = IPTR -1
        IF (IPTR.LT.2) IPTR=MCMAX
        GO TO 205
C
C  Check flags for runs expired or out-of-limit, and generate
C  appropriate messages
C
  300   IF (LBITIM) TYPE 1000
 1000   FORMAT (' **** MEASUREMENT CONTROL BIAS RUN',
        1       ' SHOULD BE MADE ****')
        IF (LPRTIM) TYPE 1001
 1001   FORMAT (' **** MEASUREMENT CONTROL PRECISION RUN',
        1       ' SHOULD BE MADE ****')
        IF (LBKTIM) TYPE 1002
 1002   FORMAT (' **** MEASUREMENT CONTROL BACKGROUND RUN',
        1       ' SHOULD BE MADE ****')
        IF (LBIERR) TYPE 1010
 1010   FORMAT (' **** LAST MEASUREMENT CONTROL BIAS RUN',
        1       ' WAS OUT OF LIMIT ****')
        IF (LPRERR) TYPE 1011
 1011   FORMAT (' **** LAST MEASUREMENT CONTROL PRECISION RUN',
        1       ' WAS OUT OF LIMIT ****')
        IF (LBKERR) TYPE 1012
```

```
 1012   FORMAT (' **** LAST MEASUREMENT CONTROL BACKGROUND RUN',
        1        ' WAS OUT OF LIMIT ****')
C
C  Set global flags to indicate whether meas control is OK
C
        BIASOK=.NOT.(LBITIM.OR.LBIERR)
        PRECOK=.NOT.(LPRTIM.OR.LPRERR)
        BACKOK=.NOT.(LBKTIM.OR.LBKERR)
C
C  If a run out of limit was found, get the operator's permission
C  before continuing
C
        IF ( .NOT.(LBIERR.OR.LPRERR.OR.LBKERR) ) GO TO 650
        IF (YESNO('Do you want to continue')) GO TO 650
C
C  Abort exit
C
  640   ABORT = .TRUE.
C
C  Non-abort exit
C
  650   CLOSE (UNIT=DKLUN)
        RETURN
        END
```

```
C
C*************************************************************************
C
C  MCHEAD
C       Outputs header information for the measurement control
        log list option
C
C       CMS 14-MAY-84
C
C       REV. 28-Aug-84 MJC
C       REV. 04-Apr-86 WDR      Modified for Micro VAX II
C
C*************************************************************************
C
        SUBROUTINE MCHEAD
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CLOG.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CMC.CMN'
C
        CALL DATTIM(1)
        WRITE(LOUT,1099)NOWDAT,NOWTIM
 1099   FORMAT('1',10X,'Measurement control log as of ',A,
     X    2X,A///1X,'Ent Date   Time  Operator Ty ',
     X    ' Result       Fractional Standard     No  VX Er '
     X    /44X,'Error      Value        Run Fg Fg'/)
        RETURN
        END
```

```
C*****************************************************************
C  MCPREC
C
C       Measurement control check for precision
C
C       06-Jun-84 lc
C
C       Rev. 07-Jun-84 lc
C       REV. 06-Sep-84 TES
C       REV. 25-Sep-84 CMS
C       Rev. 08-Jan-85 MPK      Move call to VAX comm and add
C                               call to DATTIM so that VAX
C                               gets start time
C       Rev. 18-Jan-85 MPK      Add calls to ADATE and AEVAX
C                               Remove reference to to VAXCOM
C       Rev. 22-Jan-85 MPK      Check VAXUP in COMM calls
C       Rev. 14-FEB-85 WDR      Limits for 5 and 15 runs were reversed
C       Rev. 19-Feb-85 MPK      Correct statistics
C                               Use YESNO to ask yes/no question
C       Rev. 20-Feb-85 MPK      Declared XVAL,XSIG,VALUE,XMEAN as REAL*8
C       Rev. 21-Feb-85 MPK      Declare YESNO as Logical*1
C       Rev. 07-Mar-85 MPK      Include CVAX common
C       Rev. 18-Apr-85 MPK      Remove REAL*8 declaration of VALUE
C                               (causing misalignment of CRESLT common)
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 02-May-90 WDR      Removed COMM and AEVAX for Pu238
C                               instrument
C       Rev. 03-May-90 WDR      Add call to AEND
C*****************************************************************
C
      SUBROUTINE MCPREC
C
      LOGICAL * 1 SUMMRY,YESNO
      REAL*8 XVAL(25), XSIG(25), XMEAN
      REAL * 8 WORD
      REAL * 4 PRWARN(2), PRACT(2)
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CMCONS.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CMC.CMN'
      INCLUDE 'CVAX.CMN'
C
      DATA WORD /'PRECISIO'/
      MP = .TRUE.
      RTYP = 'MC PRECN'
C
C  Initialize - set up values for the precision warning limits
C  and action limits - these values are dependent on the NRUNS
C  selected in the parameter file.  First make sure NRUNS is
C  equal to 5 or 15.
C
      IF (NRUNS.NE. 5 .AND. NRUNS.NE. 15)NRUNS = 5
      IF(NRUNS .EQ. 15) GO TO 150
```

```
C
C  Set limits for 5 runs
C
      PRWARN(1) = 0.12
      PRWARN(2) = 2.79
      PRACT(1)  = 0.05
      PRACT(2)  = 3.72
      GO TO 155
C
C  NRUNS equals 15
C
  150 PRWARN(1) = 0.40
      PRWARN(2) = 1.87
      PRACT(1)  = 0.29
      PRACT(2)  = 2.24
C
  155 ITIMES = 0
    5 XMEAN = 0.0
      SIGSQ = 0.0
      SUMMRY = .FALSE.
C
C  Dialog with operator
C
      CALL DIALOG
      IF(ABORT) GO TO 500
      NUMCYC = NRUNS          !For printout in Header summary
C
C
C  Assay, print single assay results
C  Accumulate results
C
      ICYCLE = 0
      DO 10 I=1,NRUNS
      ICYCLE = ICYCLE + 1
      CALL ADATE
        CALL ASSAY
        IF(ABORT) GO TO 500
      CALL NEWNAM(0)
        VALUE = ASANS
        SIGMA = ERRBAR
        CALL PRINT1
        XVAL(I) = VALUE
        XSIG(I) = SIGMA
        XMEAN = XMEAN + VALUE
        ABSSIG = SIGMA * VALUE
        SIGSQ = SIGSQ + ABSSIG*ABSSIG
   10   CONTINUE
C
C  Average results
C  Square mean sigma
C
      XMEAN = XMEAN/NRUNS
      SGMEAN = SIGSQ/NRUNS
C
C  Accumulate variance
C
      SUM2 = 0.0
      DO 15 I=1,NRUNS
        VAR = (XVAL(I)-XMEAN) * (XVAL(I) -XMEAN)
        SUM2 = SUM2 + VAR
```

```
 15     CONTINUE
C
C  Compute square of standard deviation
C  Compute chi square (standard deviation **2 / mean sigma **2)
C
       STDEV2 = SUM2/(NRUNS-1)
       CHISQ = STDEV2/SGMEAN
C
C  Set up variables to write to MC log file.
C
       VALUE = CHISQ
       SIGMA = STDEV2
       VAR = SGMEAN
       MCRUNS = NRUNS
C
C  Check for chi square limits
C
       PRECOK = .TRUE.
       IF(CHISQ .GT. PRWARN(1) .AND. CHISQ .LT. PRWARN(2))
     X          GO TO 250
C
C  Warning limits exceeded
C
       IF(CHISQ .LT. PRACT(1) .OR. CHISQ .GT. PRACT(2))
     X          GO TO 200
       CALL ERRMSG(75,'  ')
       GO TO 250
C
C  Action limits exceeded
C
 200   PRECOK = .FALSE.
       CALL ERRMSG(76,'  ')
C
C  Print results
C
 250   SUMMRY = .TRUE.
       CALL HEADER(SUMMRY)
       WRITE(LOUT,1500)
 1500  FORMAT(//3X,'Cycle',T15,'Result',T30,'Sigma',T50,'Abs. Sigma'/)
       DO 50 I=1,NRUNS
         WRITE(LOUT,1501)I,XVAL(I),XSIG(I),XSIG(I)*XVAL(I)
 1501    FORMAT(5X,I3,T13,F10.4,T28,F8.4,T48,F8.4)
   50  CONTINUE
       STDEV = SQRT(STDEV2)
       WRITE(LOUT,1502)XMEAN,SGMEAN,STDEV,CHISQ
 1502  FORMAT(/' Average result = ',T25,F10.4/' Mean Sq sigma = ',T25,
     X          F10.4/' Standard deviation = ',T25,F10.4/' Chisquare = ',
     X          T25,F10.4)
       ITIMES = ITIMES + 1
       IF(PRECOK .OR. ITIMES .GT. 1) GO TO 300
       IF (YESNO('Do you want to repeat precision check')) GO TO 5
C
C  Send results to VAX
C  write to measurement control log
C
 300   CALL AEND
       IF(.NOT. PRECOK) MCERR = .TRUE.
       MTYP(1) = 'M'
       MTYP(2) = 'P'
C      IF(VAXUP) CALL COMM ('RP')       !communications - precision results
```

```
      MCVAX = VAXUP
      CALL WTMCLG
C
  500 RETURN
      END
```

```
C*****************************************************************
C  MENU
C
C       Searches the table 'MENU.TXT' to locate a match
C       with the option entered by the operator.  If no
        match is found a flag is set and the routine
C       returns to the Executive.
C
C       Calling arguments and format:
C                 OPTIN (Option entered by operator)   A4
C                 MNUM  (Number of entry in table)     I2
C                 NOPT  (Logical flag set if no match) L
C
C
C       CMS 25-APR-84
C
C*****************************************************************
C
C
        SUBROUTINE MENU(OPTIN,MNUM,NOPT)
        LOGICAL * 1 NOPT
        CHARACTER   OPT(25)*4
        CHARACTER   OPTIN*4,OPTTBL*4
        INTEGER OPTNUM,MENPRM,OPTLEN
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CONTRL.CMN'
C
        DATA MENPRM /25/, OPTLEN /4/
        NOPT = .FALSE.
C
C  Open the file 'MENU.TXT' for reading
C
        OPEN(UNIT=DKLUN,NAME='MENU.TXT',TYPE='OLD',ERR=100)
        GOTO 20
  100   CALL ERRMSG(1,'MENU')
        ABORT = .TRUE.
        GOTO 70
C
C  Read the menu file and store data in the internal menu
C
   20   READ(DKLUN,10,END=15,ERR=200) OPTNUM,OPTTBL
   10   FORMAT(I3,A4)
        GOTO 505
  200   CALL ERRMSG(2,'MENU')
        GOTO 70
  505   IF (OPTNUM .GT. MENPRM .OR. OPTNUM .LT. 1) GOTO 20
        OPT(OPTNUM) = OPTTBL
        GOTO 20
   15   CLOSE(UNIT=DKLUN)
C
C  Look for the option match in the table
C
        DO 60 I = 1, MENPRM
          IF (OPTIN.NE.OPT(I)) GOTO 60
          MNUM = I
        GOTO 70
```

```
   60   CONTINUE
C
C  Unrecognized option - flag set and return
C
        TYPE *,'**** OPTION NOT RECOGNIZED - TYPE H TO SEE MENU ****'
        NOPT = .TRUE.
   70   RETURN
        END
```

```
      SUBROUTINE  MLR  (MM, M, N, A, Y, X, B, R, W, V)
C
      INTEGER*2  MM, M, N
      REAL       A(MM,N), Y(M), X(N), B(MM,N), R(M), W(N), V(N)
C
      INTEGER  MLRI
      COMMON  /MLRCC/ MLRI
C
C-----------------------------------------------------------------------
C
C     MLR (MINIMIZE LINEAR RESIDUALS) COMPUTES A SOLUTION
C     TO THE (POSSIBLY) OVERDETERMINED LINEAR SYSTEM
C
C                A X = Y
C
C     IN THE SENSE THAT THE LENGTH OF THE RESIDUAL VECTOR
C
C                R(X) = Y  -  A X
C
C     IS MINIMIZED, USING THE METHOD OF HOUSEHOLDER REFLECTIONS
C     WITH ITERATIVE IMPROVEMENT.
C
C-----------------------------------------------------------------------
C
C     CALLING SEQUENCE:
C
C           COMMON  /MLRCC/ MLRI
C           MLRI = <INTEGER VALUE>
C           CALL MLR (MM, M, N, A, Y, X, B, R, W, V)
C
C     WHERE:
C
C     MLRI  TELLS WHICH "ENTRY" TO USE:
C              MLRI = 1:  MLR
C              MLRI = 2:  MLRA     USE MLR AGAIN WITH SAME MATRIX A, NEW
C                                  NEW RIGHT-HAND SIDE Y.
C              MLRI = 3:  MLRE     COMPUTE DIAGONAL OF INVERSE OF NORMAL
C                                  MATRIX, A'A.
C              MLRI = 4:  MLRS     COMPUTE FULL INVERSE OF NORMAL MATRIX.
C
C           MLRI  IS SET TO  1  IN A DATA STATEMENT IN  MLR .
C
C       MM  IS THE NUMBER OF ROWS SPECIFIED IN THE DIMENSION STATEMENT
C           FOR A.
C
C        M  IS THE NUMBER OF EQUATIONS (ROWS).
C
C        N  IS THE NUMBER OF UNKNOWNS  (COLUMNS).
C
C        A  IS THE TWO-DIMENSIONAL ARRAY OF COEFFICIENTS.
C
C        Y  IS THE ONE-DIMENSIONAL ARRAY OF CONSTANTS (M WORDS).
C
C        X  IS A ONE-DIMENSIONAL ARRAY FOR THE UNKNOWNS (N WORDS).
C
C        B  IS A TEMPORARY STORAGE ARRAY OF MM*N WORDS.
C           IN CASE MLRI=4, B IS REPLACED BY THE FULL INVERSE OF THE
C           NORMAL MATRIX.  B IS ASSUMED TO BE DIMENSIONED EXACTLY AS A.
C
C        R  IS A ONE-DIMENSIONAL ARRAY FOR THE RESIDUALS (M WORDS).
```

```
C            IN CASE MLRI=3, R IS REPLACED BY THE DIAGONAL ELEMENTS
C            OF THE INVERSE OF THE NORMAL MATRIX (N WORDS).
C
C         W  IS A TEMPORARY STORAGE ARRAY OF N WORDS.
C
C         V  IS A TEMPORARY STORAGE ARRAY OF N WORDS.
C
C       INPUT ARGUMENTS:  MLRI, MM, M, N, A, Y.
C     WORKING ARGUMENTS:  B, R, W, V.
C      OUTPUT ARGUMENTS:  X, R.
C
C     STANDARD FORTRAN FUNCTIONS REFERENCED:  DBLE, SIGN, SQRT
C
C     OTHER ROUTINES CALLED:  MLRERR
C
C-----------------------------------------------------------------------
C
C     CAUTION:
C
C       1. A CALL WITH  MLRI.GT.1  ASSUMES THAT:
C          A. MLR HAS PREVIOUSLY BEEN CALLED WITH  MRLI=1 , AND
C          B. THE CONTENTS OF ARRAYS  A, B, W, V  HAVE NOT BEEN
C             DISTURBED SINCE THE PREVIOUS CALL.
C          (THERE IS NO WAY TO CHECK FOR THIS.)
C
C       2. MLR MAY NOT BE CALLED AGAIN WITH  MLRI.GT.1
C          IF THE  MLRI=4  OPTION IS USED.
C
C-----------------------------------------------------------------------
C
C     STANDARD FORTRAN VERSION WRITTEN BY  F. N. FRITSCH  (LLL).
C     DATE LAST CHANGED:  24 MAY 1978.
C
C-----------------------------------------------------------------------
C
C     NOTE..  THE USER MAY MONITOR THE ERROR FLAG  JFLAG  BY INCLUDING
C             THE FOLLOWING COMMON BLOCK DECLARATION IN THE CALLING
C             PROGRAM.  (JFLAG.EQ.0  FOR NORMAL RETURN. )
C                       (JFLAG.NE.0  FOR  ERROR RETURN. )
      INTEGER  JFLAG
      COMMON  /MLRERF/ JFLAG
C
C.....DECLARE LOCAL VARIABLES.
      INTEGER            I, IN, IP, J, JM, K
      REAL               RE, REOLD, TEMP, YE
      DOUBLE PRECISION   RD, XD
C
      DATA  MLRI /1/
C
C.....CHECK FOR INVALID INPUTS.
      JFLAG = -1
      IF (MLRI.LT.1 .OR. MLRI.GT.4)  GO TO 510
      JFLAG = 0
      IF (M.GT.MM) JFLAG = 55
      IF (N.GT.M)  JFLAG = 56
      IF (N.LT.1)  JFLAG = 57
      IF (JFLAG.NE.0)  GO TO 510
C
      GO TO  (1, 100, 300, 400), MLRI
C
```

```
C-----------------------------------------------------------------------
C
C...CASE  MLRI = 1    ( MLR:
C                     ( NORMAL USAGE -- COMPLETE SOLUTION.
    1 CONTINUE

C       INITIALIZE  B  TO  A .
        DO 20  I=1,M
           DO 10  J=1,N
              B(I,J) = A(I,J)
   10      CONTINUE
   20   CONTINUE
C
C.....TRIANGULARIZATION PHASE.
C
        DO 70  I=1,N
           TEMP = 0.
           DO 30  K=I,M
              TEMP = TEMP + B(K,I)**2
   30      CONTINUE
C          TEST FOR SINGULARITY.
           IF (TEMP .EQ. 0.)  GO TO 500
           TEMP = SQRT(TEMP)
           W(I) = SIGN (TEMP, B(I,I))
           B(I,I) = B(I,I) + W(I)
           V(I) = - B(I,I) * W(I)
           IF (I .EQ. N)  GO TO 80
           IP = I+1
           DO 60  J=IP,N
              TEMP = 0.
              DO 40  K=I,M
                 TEMP = TEMP + B(K,I)*B(K,J)
   40         CONTINUE
              TEMP = TEMP/V(I)
              DO 50  K=I,M
                 B(K,J) = B(K,J) + TEMP*B(K,I)
   50         CONTINUE
   60      CONTINUE
   70   CONTINUE
C
   80   CONTINUE
C
C-----------------------------------------------------------------------
C
C...CASE  MLRI = 2    ( MLRA:
C                     ( RE-USE  MLR  WITH SAME  A , DIFFERENT  Y .
  100   CONTINUE
C
C       INITIALIZE  X  TO ZERO.
        DO 110  I=1,N
           X(I) = 0.
  110   CONTINUE
C       INITIALIZE  R  TO  Y  AND COMPUTE LENGTH OF  Y .
        YE = 0.
        DO 120 I=1,M
           R(I) = Y(I)
           YE = YE + Y(I)**2
  120   CONTINUE
C       ALREADY DONE IF INPUT VECTOR IS ZERO.
        IF (YE .EQ. 0.)  RETURN
```

```
      REOLD = YE
C
C.....FORWARD SOLUTION PHASE.
C
  130 CONTINUE
      COMPUTE VECTOR  Z .
      DO 160 I=1,N
         TEMP = 0.
         DO 140 K=I,M
            TEMP = TEMP + B(K,I)*R(K)
  140    CONTINUE
         TEMP = TEMP/V(I)
         DO 150 K=I,M
            R(K) = R(K) + TEMP*B(K,I)
  150    CONTINUE
  160 CONTINUE
C
C.....BACK-SUBSTITUTION PHASE.
C
C     COMPUTE CORRECTION, USING  R  FOR TEMPORARY STORAGE.
      DO 190  I = 1, N
C        IN RUNS FROM N DOWN TO 1.
         IN = N - I + 1
         R(IN) = R(IN) / W(IN)
         IF (IN .EQ. 1)  GO TO 190                                    78-05-24
         DO 180  K = 2, IN
C           J RUNS FROM IN-1 DOWN TO 1.
            J = IN - K + 1
            R(J) = R(J) + R(IN)*B(J,IN)
  180    CONTINUE
  190 CONTINUE

C     ADD CORRECTION TO  X .   (DOUBLE PRECISION)
      DO 200 I=1,N
         XD = X(I)
         RD = R(I)
         XD = XD - RD
         X(I) = XD
  200 CONTINUE
C
C.....ITERATIVE IMPROVEMENT PHASE.
C
C     DOUBLE PRECISION RESIDUAL CALCULATION.
      DO 230 I=1,M
         RD = Y(I)
         DO 220 J=1,N
            RD = RD - DBLE(A(I,J))*DBLE(X(J))
  220    CONTINUE
         R(I) = RD
  230 CONTINUE
C     COMPUTE LENGTH OF NEW RESIDUAL VECTOR.
      RE = 0.
      DO 240 I=1,M
         RE = RE + R(I)**2
  240 CONTINUE
C     DONE WITH ITERATIVE IMPROVEMENT IF LENGTH OF  R  NOT DECREASED.
      IF (RE .GE. REOLD)  GO TO 270
      REOLD = RE
      GO TO 130
C
```

```
  270 CONTINUE
C     TEST FOR NUMERICAL SINGULARITY.
      IF (RE .LT. YE)  RETURN
      JFLAG = 58
      GO TO 510

C-----------------------------------------------------------------------
C
C...CASE  MLRI = 3    ( MLRE:
C                     ( COMPUTE DIAGONAL OF (A'A)-INVERSE = (T'T)-INVERSE
  300 CONTINUE
C
      DO 350 I = 1, N
C.....COMPUTE I-TH ROW OF T-INVERSE AND STORE IN  R(I), R(I+1),
C           ..., R(N).
         R(I) = -1. / W(I)
         IF (N .LE. I)  GO TO 330
         IP = I+1
         DO 320 J = IP, N
            TEMP = 0.
            JM = J-1
            DO 310 K = I, JM
               TEMP = TEMP + R(K)*B(K,J)
  310       CONTINUE
            R(J) = TEMP / W(J)
  320    CONTINUE
C.....COMPUTE I-TH DIAGONAL ELEMENT OF (T-INVERSE) X (T-INVERSE)'
C           AND STORE IN  R(I).
  330    CONTINUE
         TEMP = 0.
         DO 340 J = I, N
            TEMP = TEMP + R(J)**2
  340    CONTINUE
         R(I) = TEMP
  350 CONTINUE
C
      RETURN
C
C-----------------------------------------------------------------------
C
C...CASE  MLRI = 4    ( MLRS:
C                     ( COMPUTE FULL INVERSE OF  A'A = T'T .
  400 CONTINUE
C
C.....FORM (T-INVERSE) IN  B .
      DO 430  I = 1, N
         B(I,I) = -1. / W(I)
         IF (N .LE. I)  GO TO 440
         IP = I+1
         DO 420  J = IP, N
            TEMP = 0.
            JM = J-1
            DO 410  K = I, JM
               TEMP = TEMP + B(I,K)*B(K,J)
  410       CONTINUE
            B(I,J) = TEMP / W(J)
  420    CONTINUE
  430 CONTINUE
C
C.....COMPUTE  (T-INVERSE) X (T-INVERSE)' .
```

```
  440 CONTINUE
      DO 470  I = 1, N
         DO 460  J = I, N
            TEMP = 0.
            DO 450  K = J, N
               TEMP = TEMP + B(I,K)*B(J,K)
  450       CONTINUE
            B(I,J) = TEMP
            B(J,I) = TEMP
  460    CONTINUE
  470 CONTINUE
C
      RETURN
C
C-----------------------------------------------------------------------
C
C...ERROR RETURNS.
C
  500 CONTINUE
      JFLAG = 59
C
  510 CONTINUE
C     FOLLOWING CALL ASSUMES THAT  MLRERR  IGNORES  RE  AND  YE
C     UNLESS  JFLAG = 58 .
C
      CALL  MLRERR (RE, YE)
C
      RETURN
C
C-----------------------------------------------------------------------
C
      END
      SUBROUTINE  MLRERR (RE, YE)
      REAL  RE, YE
C
      INTEGER  JFLAG
      COMMON  /MLRERF/ JFLAG
C
      INTEGER  OUB
      COMMON  /MLRUNIT/ OUB
C
C-----------------------------------------------------------------------
C
C          M L R    E R R O R    R O U T I N E
C
C     THIS ROUTINE IS CALLED IF, AND ONLY IF, AN ERROR IS ENCOUNTERED
C     IN  MLR .
C
C-----------------------------------------------------------------------
C
C     CALLING SEQUENCE:
C
C             COMMON  /MLRERF/ JFLAG
C             JFLAG = <NONZERO INTEGER VALUE>
C             CALL  MLRERR (RE, YE)
C
      WHERE:
C
C      JFLAG  INDICATES WHICH ERROR HAS OCCURRED.  ALLOWABLE VALUES ARE:
C               JFLAG = -1:  ILLEGAL VALUE OF  MLRI .
```

```
C                 JFLAG = 55:  M > MM .
C                 JFLAG = 56:  N > M .
C                 JFLAG = 57:  N < 1 .
C                 JFLAG = 58:  SINGULAR SYSTEM DETECTED IN  MLRA .
C                              [LENGTH(R) .GE. LENGTH(Y).]
C                 JFLAG = 59:  SINGULAR MATRIX DETECTED DURING TRIANGULAR-
C                              IZATION PHASE.
C                              [AN ELEMENT OF  W  IS ZERO.]
C
C       RE,YE  ARE THE SQUARED LENGTHS OF  R  AND  Y , RESPECTIVELY,
C              AND ARE IGNORED UNLESS  JFLAG=58 .
C
C       OUB    DETERMINES THE DISPOSITION OF ERROR MESSAGES:
C                OUB < 0:  NO PRINTED MESSAGES.
C                OUB = 0:  TELETYPE OUTPUT (DEFAULT).
C                OUB > 0:  OUTPUT TO LOGICAL UNIT  OUB .
C
C-----------------------------------------------------------------------
C
C     WRITTEN BY  F. N. FRITSCH  (LLL).
C     DATE LAST CHANGED:  10 OCTOBER 1975.
C
C-----------------------------------------------------------------------
C
      INTEGER  MLRI
      COMMON  /MLRCC/ MLRI
C
C.....DECLARE LOCAL VARIABLES.
      INTEGER  LOUT, JFL
C
      DATA  OUB /0/

C.....SET UP  LOUT FOR 11/70.
      LOUT = OUB
C     TELETYPE OUTPUT IF  OUB=0 .
      IF (OUB .EQ. 0)  LOUT = 5
C
      IF (LOUT .GT. 0)    WRITE (LOUT, 2000)  JFLAG
C
C.....HANDLE SPECIAL CASE  JFLAG = -1 .
      IF (JFLAG .NE. -1)  GO TO 1000
      IF (LOUT .GT. 0)    WRITE (LOUT, 2001)  MLRI
      RETURN
C
C.....CHECK FOR ILLEGAL VALUE OF  JFLAG .
 1000 CONTINUE
      IF (JFLAG.LT.55 .OR. JFLAG.GT.59)  GO TO 1060
      IF (LOUT.LE.0)  RETURN
C
      JFL = JFLAG - 54
      GO TO  (1010, 1020, 1030, 1040, 1050),  JFL
C
C.....CASE JFL = 1  (JFLAG = 55).
 1010    CONTINUE
         WRITE (LOUT, 2005)
         WRITE (LOUT, 2010)
         RETURN
C
C.....CASE JFL = 2  (JFLAG = 56).
 1020    CONTINUE
```

```
          WRITE (LOUT, 2005)
          WRITE (LOUT, 2020)
          RETURN
C
C.....CASE JFL = 3  (JFLAG = 57).
 1030     CONTINUE
          WRITE (LOUT, 2005)
          WRITE (LOUT, 2030)
          RETURN
C
C.....CASE JFL = 4  (JFLAG = 58).
 1040     CONTINUE
          WRITE (LOUT, 2040)  RE, YE
          RETURN
C
C.....CASE JFL = 5  (JFLAG = 59).
 1050     CONTINUE
          WRITE (LOUT, 2050)
          RETURN
C.....ENDCASE (JFL)
C
C.....ILLEGAL  JFLAG  VALUE.  (LOGICALLY, THIS CASE CANNOT OCCUR.)
 1060 CONTINUE
      IF (LOUT .LT. 0)  LOUT = 59
      WRITE (LOUT, 2060)
      CALL EXIT
C
C-------------------------------------------------------------------------
C
C.....FORMATS.
C
 2000 FORMAT (/24H ***   MLR ERROR NUMBER ,I4,6H   ***)
 2001 FORMAT (6X,23HILLEGAL VALUE OF MLRI: ,I5)
 2005 FORMAT (6X,38HSPECIFICATION ERROR IN SUBROUTINE MLR:)
 2010 FORMAT (8X,36H M GREATER THAN MM - STORAGE OVERLAP )
 2020 FORMAT (8X,48H N GREATER THAN M - MORE UNKNOWNS THAN EQUATIONS )
 2030 FORMAT (8X,14H N LESS THAN 1 )
 2040 FORMAT (6X,27HMATRIX IS SINGULAR IN MLR: /
     *        8X,49H SUM OF SQUARES OF RESIDUALS R(I) GREATER THAN OR,
     *           42H EQUAL TO SUM OF SQUARES OF VARIABLES Y(I) /
     *        8X,18H SUM OF R(I)**2 = ,E15.8,
     *        5X,18H SUM OF Y(I)**2 = ,E15.8)
 2050 FORMAT (6X,27HMATRIX IS SINGULAR IN MLR: /
     *        8X,24H AN ELEMENT OF W IS ZERO )
 2060 FORMAT (6X,38HILLEGAL ERROR NUMBER -- JOB TERMINATED)
C
      END
```

```
C*****************************************************************
C
C NEWNM8        (NEWNAM subroutine for PU-238 isotopic instrument)
C
C       Create a new filename for an autocycle run
        NEWNMS is a special adaptation of NEWNAM for the solid
C       isotopic instrument.
C
C       Calling arguments:
C       OPTION = 0  create the name with .INS as the extension
C
C       If autocycle mode:
C
C       OPTION = 0  create the name with .I01 where 'I' is
C                    the shortened extension for the instrument
C       OPTION = 1  update the name to .I02, .I03, etc.
C
C       If user inputs the extension, assume that the extension
C       is correct for the first file and do not change it
C
C
C       SSJ 19-DEC-79
C
C       REV. 29-Jun-82  SSJ
C       REV. 07-Sep-84  CMS
C       REV. 21-NOV-84  WDR     Modified for solid isotopic instrument
C       REV. 14-FEB-85  WDR     Check for null file name, and
C                               don't touch it if it is.
C       REV. 04-APR-86  WDR     Modified for Micro VAX II
C       REV. 13-JUL-87  RDP     Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings

C*****************************************************************
C
        SUBROUTINE NEWNAM(OPTION)
C
        CHARACTER ACHAR*1
        INTEGER*2 OPTION
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CCAL.CMN'
C
C  If the file name is null, bug out now
C
        IF (FILNAM.EQ.' ') RETURN
C
C  Check whether option is 0 (create initial extension)
C  or 1 (increment the extension)
C
        IF(OPTION .EQ. 1) GO TO 200
C
C  Rules:
        The first blank or dot ends the filename proper
C       - only six characters may be in the filename.
C       If a colon appears in letter 3 or 4, six more letters
C       may appear in the name.
```

```
C       All special characters will be converted to a '*'
C
C  Initialize letter pointer K
C  Initialize count of letters in filename, J, not counting device
C
        K = 0
        J = 0
C
C  Look for blank, dot, or colon
C  Count up to 6 characters after the colon
C  Colon must be in letters 3 or 4
C
 10     K = K + 1
        ACHAR = FILNAM(K:K)
        IF(ACHAR .EQ. '.') GO TO 80
        IF(J .EQ. 8 .OR. ACHAR .EQ. ' ') GO TO 100
        IF(ACHAR .NE. ':') GO TO 20
C
C  Found:
C  Make sure it is the 3rd or 4th character and not repeated
C
        IF(J .NE. 0 .AND. (K .EQ. 3 .OR. K .EQ. 4)) GO TO 15
        GO TO 20
 15     J = 0
        GO TO 10
C
C  Check for alpha-numeric characters
C  Replace anything bad with *
C
 20     IF((ACHAR .GE. 'A' .AND. ACHAR .LE. 'Z')  .OR.
     X          (ACHAR .GE. '0' .AND. ACHAR .LE. '9')  )
     X          GO TO 25
        FILNAM(K:K) = '*'
 25     J = J + 1
        GO TO 10
C
C  User typed the extension
C
 80     GO TO 500
C
C  Insert dot
C  Insert extension 'INS' if not autocycle
C  If autocycle, insert extension 'IOO'
C
 100    IF(K .GT. 11) STOP 'Filename problem'
        FILNAM(K:K) = '.'
        FILNAM(K+1:K+1) = 'W'
        IF (AUTOCY) GO TO 300
        IF (CAL) GO TO 300
        FILNAM(K+2:K+2) = 'H'
        FILNAM(K+3:K+3) = 'S'
        GO TO 310
 300    FILNAM(K+2:K+2) = '0'
        FILNAM(K+3:K+3) = '1'
C
C  Insert blanks at end
C
 310    K = K + 4
 150    FILNAM(K:K) = ' '
```

```
        K = K+1
        IF(K .LE. 15) GO TO 150
        GO TO 500
C
C  Update the previous filename
   First search for the dot
C
 200    K = 0
 210    K = K + 1
        IF(K .GT. 15) STOP 'No dot in filename'
        IF (FILNAM(K:K) .EQ. '.') GO TO 220
        GOTO 210
C
C  Update the last digit
C  If it goes beyond 9 (:), update the previous digit
C
C       !Reset character after dot
C 220   FILNAM(K+1:K+1)= CHAR(ICHAR(FILNAM(K+1:K+1)) - 1)
220     K = K + 3                          !It was bumped for ADC #2 data
 225    IF (FILNAM(K:K) .EQ. '.') GOTO 500
        FILNAM(K:K) = CHAR(ICHAR(FILNAM(K:K)) + 1)
        IF (FILNAM(K:K) .NE. ':') GOTO 500
        FILNAM(K:K) = '0'
        K = K - 1
        GO TO 225
C
 500    RETURN
        END
```

```
      SUBROUTINE OUT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Sub to accumulate results of peak fitting, calculate photon
C  emission rate of source, tabulate data & results, and store them,
   if requested.
C               Last edit for VAX        13-Oct-87
C               Modified for Pu238 instrument  WDR  1-Sep-89
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPGMS.PMS'
C  The following common is required for the instrument
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CLUNS.CMN'
C
      DIMENSION YSUM(MXNP),RES(MXDP)
      EQUIVALENCE (KK,NWRDS(2))
      EQUIVALENCE (YSUM,SUM),(RES,DELY)

      DIMENSION QFT(100),SDOF(MXNP),TCTS(MXNP),ENDM(MXNP)
      DIMENSION QFTL(MXPKS)
      EQUIVALENCE (WM(100),ENDM)
      BYTE BKSMBL,EFSMBL

      CHARACTER*8 TBKGD
      CHARACTER*22 TSLPS

      CHARACTER*6 TWID
      CHARACTER*37 TINT
      CHARACTER*13 TNGY
      CHARACTER*66 TOUT
      CHARACTER*9 ALPJ,STAMP,STSLP,TLAMP,TLSLP

      EQUIVALENCE (TWID,TOUT(1:6)),(TOUT(10:22),TNGY)
      EQUIVALENCE (TOUT(29:),TINT)

C
      DATA LP/6/
C     DATA LP/-1/      !For VAX
      DATA EFSMBL/' '/

      IF(RUNFLG(4) .EQ. 0) EFSMBL = '*'  !Assuming det. efficiency = 1
C
C     IOFLG(2) is set when last grouping done (preparing summary)
C     IOFLG(3) is set to inhibit line printer output
C     INTFLG is set for single peaks that were simply integrated
C
      IF(IOFLG(2) .NE. 0)GO TO 300

      IOFLG(3) = 1     ! NO LINE PRINTER
      IF(IOFLG(3) .NE. 0)GO TO 30

      ALPJ = 'FIXED'
      STAMP = 'FIXED'
      STSLP = 'FIXED'
      TLAMP = 'FIXED'
      TLSLP = 'FIXED'
```

```
      IF(FIXFLG2 .NE. 0)STAMP = 'FIXED *'
      IF(FIXFLG4 .NE. 0)TLAMP = 'FIXED *'
      IF(NFLG(1) .NE. 0)ALPJ = 'FREE'
      IF(NFLG(2) .NE. 0)STAMP = 'FREE'
      IF(NFLG(3) .NE. 0)STSLP = 'FREE'
      IF(NFLG(4) .NE. 0)TLAMP = 'FREE'
      IF(NFLG(5) .NE. 0)TLSLP = 'FREE'

      WRITE(LOUT,1010),KTGRP,EST,EEND,IST,IEND

      TSLPS = '(as prespecified)'

      IF (BKLO .EQ. 0) THEN
         TBKGD = ' REMOTE '
         WRITE(LOUT,1011)TBKGD,RBKWID(1),NUMS,'low'
         WRITE(LOUT,1013)RBKEN(1)
         WRITE(LOUT,1012)SLP(1),TSLPS
         GO TO 15
      ENDIF

         TBKGD = 'Adjacent'
      IF (RBKEN(1) .EQ. 0) THEN
         WRITE(LOUT,1011)TBKGD,BKLO,NUMS,'low'
         WRITE (LOUT,1012)SLP(1),TSLPS
      ELSE
         TSLPS = 'determined internally'
         WRITE(LOUT,1011)TBKGD,BKLO,NUMS,'low'
         WRITE (LOUT,1012)SLP(1),TSLPS
         WRITE(LOUT,1014)RBKWID(1),RBKEN(1)
      ENDIF

      TSLPS = '(as prespecified)'
      IF (BKHI .EQ. 0) THEN
         TBKGD = ' REMOTE '
         WRITE(LOUT,1011)TBKGD,RBKWID(2),NUME,'high'
         WRITE(LOUT,1013)RBKEN(2)
         WRITE(LOUT,1012)SLP(2),TSLPS
         GO TO 20
      ENDIF

         TBKGD = 'Adjacent'
      IF (RBKEN(2) .EQ. 0) THEN
         WRITE(LOUT,1011)TBKGD,BKHI,NUME,'high'
         WRITE (LOUT,1012)SLP(2),TSLPS
      ELSE
         TSLPS = 'determined internally'
         WRITE(LOUT,1011)TBKGD,BKHI,NUME,'high'
         WRITE (LOUT,1012)SLP(2),TSLPS
         WRITE(LOUT,1014)RBKWID(2),RBKEN(2)
      ENDIF


1010  FORMAT(/' ----- Analyzed peak grouping No.',I3,' from',F9.3,
     +  ' to',F9.3,' keV'/' by reading channels',I5,' to',I5,' and',
     +  ' subtracting background determined over')
1011  FORMAT (3X,A,' area of',F8.3,' keV(',I3,' chans) on the ',A,
     +  ' E side,'*)
1012  FORMAT (4X,'extrapolated with a slope of',F9.4,'%/keV; ',A)
1013  FORMAT('+ beginning at',F8.3,' keV')
1014  FORMAT(5X,'using also a REMOTE area of',F8.3,' keV starting',
```

```
     +   ' at',F8.3,'keV')

20        IF(NGAIN .NE. 0)GO TO 25
          WRITE(LOUT,1020)STD(1),STD(2)
          GO TO 30
   )0     FORMAT(/' Reference peak energies used for gain:',F8.3,
          1   ' and',F8.3)
25        WRITE(LOUT,1025)
1025      FORMAT (/' GAIN IS FIXED')

          IF(INTFLG .GT. 1)WRITE(LOUT,1030),GAIN,ZERO
1030      FORMAT(' GAIN was redetermined as',F8.4,' and ZERO as',F8.3)
30        IF (INTFLG .EQ. 0)GO TO 40
          TQFTL = 0.
          CHAN = PKPOS(1)
          EPK = ENJ(1)
          CTS = SUM(1)
          PCERR = ERR(1)
C               Estimate FWHM for summed peaks, if needed.
          IF(FWHM .LE. 0) FWHM = SQRT (.3 + .002*EPK)
          NP = 1
          IF(IOFLG(3) .EQ. 0) WRITE(LOUT,1031)KTGRP
1031      FORMAT(/' - Using simple integration for group #',I3)

          IF(RUNFLG(4) .NE. 0) THEN
                   IF (LONGPR) WRITE (LOUT,1035)
                   IF(IOFLG(3).EQ.0) WRITE(LOUT,1035)
            ELSE
                   IF (LONGPR)WRITE (LOUT,1036)
                   IF(IOFLG(3).EQ.0) WRITE(LOUT,1036)
          ENDIF

1035      FORMAT (/' Peak  Channel    Energy      Counts    ',
     + ' Gammas/min   Pcterr')
1036      FORMAT (/' Peak  Channel    Energy      Counts    ',
     + ' Counts/min * Pcterr')

          GO TO 155

40        IF(IOFLG(3) .NE. 0) GO TO 43
          WRITE (LOUT,1042) ALPJ,STAMP,STSLP,TLAMP,TLSLP
          IF(NUSHPS .NE. 0) THEN
                   WRITE(LOUT,1040)
          ELSE
                   WRITE (LOUT, 1041)
          ENDIF
          IF ((FIXFLG2 + FIXFLG4) .NE. 0) WRITE(LOUT,1043)
1040      FORMAT (' These fitting parameters initialized as specified ',
     +    ' in the control file.')
1041      FORMAT (' Initial values of fitting parameters taken from',
     +    ' shape parameter file.')
1042      FORMAT(' Parameter type: ',9X,'Amplitude   Slope'/
     +    ' Alpha',23X,A/' Short term tail'13X,A,A/
     +    ' Long term tail',14X,A,A/)
1043      FORMAT(' Starred parameters turned negative 5 times & were'
     +    ' fixed at a small positive value')


43        SG = -2.7726/ ALPHA
          POS = NDPTS/2  !For FWHM at mid-energy.
```

```
      IF (NJ .EQ. 1)  POS = PKPOS(1)  !At peak position
      SG = SG + ASLP*POS - .462
      ALFA = -2.7726/SG
      FWHM = SQRT(SG)
      TSHP5 = EXP2/GAIN
      TSHP8 = EXP4/GAIN

      DO 44 I=1,100
44    QFT(I)=0.0
      QFT(50)=1.
      DX=0.
      L=50

      DO 46 I=51,100
      DX=DX+1.
      L=L-1
      QFT(I)=SQRT(EXP(ALFA*DX*DX))
      QFT(L)=QFT(I)
      IF(QFT(I)-.01)48,46,46
46    CONTINUE

48    ASUM = 0.0
      DO 50 J=1,NP
      SDOF(J)=0.0
      SUMR(J) = 0.0

C     CALCULATE PORTION OF PEAK AREAS OUTSIDE BOUNDARIES

      IF (EXP2 .NE. 0.0)  YSUM(J)=YSUM(J)+PKHT(J)*EXP1/EXP2*
     +  EXP(-EXP2*PKPOS(J))
      FRACT1 = 0.
      FRACT2 = 0.
      IF(GAMA(J) .LE. 0)  GOTO 49
C     GAMA(J)=GAMA(J)*.001/GAIN
      FRAC1 = ATAN(2. * PKPOS(J)/GAMA(J))
      FRACT1 = 1. - .6366 * FRAC1
      DELT = NDPTS - PKPOS(J)
      FRAC2 = ATAN(2. * DELT/GAMA(J))
      FRACT2 = 1. - .6366 * FRAC2
      YSUM(J) = YSUM(J) * (1. + FRACT1 + FRACT2)
49    ASUM = ASUM + YSUM(J)
50    CONTINUE

      TSUM=0.0

      DO 60 I = 1,NDPTS
      YT = YNET(I) + AVEBG
      IF (YT .LE. 0.0)  YT = 1.0
      RES(I) = RM(I) * SQRT (YT)
      RR = RM(I)**2
      TSUM = TSUM + RR

      DO 55 J=1,NP
      NPK=PKPOS(J)+0.5
      K=I-NPK+50
      IF(K.GT.100)GOTO 55
      IF(K .LE. 0)GOTO 55
      SDOF(J)=SDOF(J)+QFT(K)
      SUMR(J)=SUMR(J)+QFT(K)*RR
55    CONTINUE
```

```
60      CONTINUE

        L = KK
        DO 70 J=1,NP
        VM(J) = 0.        !Will use to re-sort diagonal matrix elem.
        ENDM(J) = 0.
70      TCTS(J) = 0.
        DO 80 J = 1,NP
        IF(NYFLG(J) .LE. 0)GO TO 73
        L = L + 1
        VM(J) = DM(L)
        I = J
        GO TO 75
73      I = -NYFLG(J)
75      TCTS(I) = TCTS(I) + YSUM(J)
        IF(NXFLG(J) .LE. 0)GO TO 80
        L = L + 1
        ENDM(J) = DM(L)
80      CONTINUE

        DOF = NDPTS - L
        QFIT = TSUM / DOF
        RMSD = 1. + 100.*(DOF * (QFIT - 1.0) / ASUM)
        FWHM = FWHM * GAIN

        IF (LONGPR)WRITE (LOUT,1090)KTGRP,ITER,QFIT,FWHM,ERR(3),GAIN,ZERO,
     1  EXP1,ERR(4),TSHP5,ERR(5),RMSD,EXP3,ERR(6),TSHP8,ERR(7)
1090    FORMAT(/12X,'Results from group',I3,'   (',I3,' iterations)'/
     +  ' QFIT',F8.3,';  FWHM',F8.5,'+/- ',F6.4,';  KEV/CHANNEL',
     +  F9.5,', ZERO',F9.3,/
     +  20X,' EXP1',F9.5,'+/-',F7.4,'  EXP2',F9.5,'+/-',F7.4,/' NQFIT',
     +  F6.3,8X,' EXP3',F9.5,'+/-',F7.4,'  EXP4',F9.5,'+/-',F7.4)

        IF(IOFLG(3) .NE. 0)GO TO 150
        WRITE(LOUT,1095)
1095    FORMAT('OPEAK   ENERGY WIDTH   PEAK-ENERGY        PEAK-',
     +  'INTENSITY',18X,'AREA'/'  NO.     keV      eV',
     +  7X,2('SPECS',14X),8X,'RATIO'/)
        DO 110 I=1,NP
        TOUT = '                      '
        TENJ = ENJ(I)
        IF(TENJ .LT. 0)TENJ = -TENJ      !Remove X-RAY flag.
C       TGAMA = GAMA(I)*GAIN/.001
        IF (GAMA(I) .NE. 0) WRITE (TWID,1100),GAMA(I)*GAIN/.001

        IF(NXFLG(I).LT.0) THEN
                WRITE (TNGY,1101),'FIXED VS # ',-NXFLG(I)
        ELSEIF (NXFLG(I) .EQ. 0) THEN
                TNGY = '  FIXED'
        ELSE
                TNGY = '   FREE'
        ENDIF


        IF (NYFLG(I) .GT. 0) THEN
                TINT = '    FREE'
                GO TO 105
        ELSE IF (HIGHT(I) .GE. 0) THEN
```

```
                WRITE (TINT,1103)' Fixed at ',HIGHT(I)
        ELSE

                WRITE (TINT,1104) -NYFLG(I),YSUM(I)/YSUM(-NYFLG(I))
        ENDIF

105     WRITE (LOUT,1105) I,TENJ,TOUT
110     CONTINUE

1100    FORMAT(F6.1)
1101    FORMAT(A,I2)
1103    FORMAT(A,F10.2,' Counts')
1104    FORMAT('Fixed vs # ',I2,13X,F9.5)
1105    FORMAT(I4,F10.3,A)

        WRITE(LOUT,1090)KTGRP,ITER,QFIT,FWHM,ERR(3),GAIN,ZERO,EXP1,
     1  ERR(4),TSHP5,ERR(5),RMSD,EXP3,ERR(6),TSHP8,ERR(7)


150     IF(RUNFLG(4) .NE. 0) THEN
                IF (LONGPR)WRITE (LOUT,1150)
                IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1150)
          ELSE
                IF (LONGPR)WRITE (LOUT,1151)
                IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1151)
        ENDIF

1150    FORMAT (/' Peak  Channel    Energy   Error     Counts    ',
     + ' Gammas/min   Pcterr    Qfit')
1151    FORMAT (/' Peak  Channel    Energy   Error     Counts    ',
     + ' Counts/min * Pcterr    Qfit')

155     DO 230 J = 1,NP
        IF(INTFLG .NE. 0)GO TO 175
        SG = -2.7726/ALPHA + ASLP * PKPOS(J)
        ALFA = -2.7726/SG
        PCERR = 0.0
        L = J
        IF(NYFLG(J) .LT. 0)L = -NYFLG(J)
        IF(PKHT(L) .LE. 0)GO TO 160

        TQFTL = SUMR(L)/(SDOF(L)-1.)
        IF(TQFTL .LT. 1.) TQFTL = 1.0
        PCERR=SQRT(TCTS(L)/YSUM(J))*(100.*VM(L)*SQRT(TQFTL))/PKHT(L)


160     CTS = YSUM(J)
        CHAN = PKPOS(J) + OFFSET
        ERRNGY = 0.
        IF(NXFLG(J) .GT. 0)ERRNGY = ENDM(J) * GAIN

175     EPK = REFEN + (CHAN  - REFCH) * GAIN

C       SUBTRACT KNOWN BACKGROUND PEAKS FOR THIS DETECTOR

        BKSMBL = ' '
        DO 180, JJ = 1,NBKPKS
        IF(BKPKEN(JJ) .EQ. 0)GO TO 190
        IF(ABS(EPK-BKPKEN(JJ)) .GT. FWHM)GOTO 180  !Window = FWHM keV
        BKSMBL = 'b'
```

```
      CTS1 = CTS
      CTS = CTS - BKPKCT(JJ)*TIMEL
      IF(CTS .LE. 0.)GOTO 179
      PCERR = PCERR * CTS1 / CTS
      IF (PCERR .LT. 100.) GO TO 180
      CTS = 0.0
      PCERR = 0.01 * PCERR * CTS1
180   CONTINUE

C     THIS PORTION OF CODING CALCULATES ABSOLUTE PHOTON INTENSITIES

190   GAMAS =0.
      VALUE=CTS
      IF (CTS .EQ. 0.0) VALUE = 2.0 * PCERR


C  CALGPM is always called for absorber corrections as well as efficiency
C  If efficiency corrns. not made, counts/min will be corrected.
      CALL CALGPM(EPK,VALUE,PCERR)

      IF(CTS .EQ. 0.) PCERR = 200.

      GAMAS=VALUE
      GAMAS = (GAMAS / TIMEL)
C     Note that TIMEL is LIVTIM in minutes

      IF(INTFLG .NE. 0)GO TO 206
      IF(ERRNGY .EQ. 0.)GO TO 201
      IF (LONGPR)WRITE (LOUT,1200)J,CHAN,EPK,ERRNGY,CTS,BKSMBL,GAMAS,PCERR,TQF
      IF(IOFLG(3).EQ.0)WRITE(LOUT,1200)J,CHAN,EPK,ERRNGY,CTS,BKSMBL,
     + GAMAS,PCERR,TQFTL
1200  FORMAT (I5,F9.3,F10.3,F8.3,F12.0,A1,1PE13.4,0PF9.3,F8.3)
      GO TO 210
201   IF (LONGPR)WRITE (LOUT,1201)J,CHAN,EPK,CTS,BKSMBL,GAMAS,PCERR,TQFTL
1201  FORMAT (I5,F9.3,F10.3,8X,F12.0,A1,1PE13.4,0PF9.3,F8.3)
      IF(IOFLG(3).EQ.0)WRITE(LOUT,1201)J,CHAN,EPK,CTS,BKSMBL,GAMAS,
     + PCERR,TQFTL
      GO TO 210

206   IF (LONGPR)WRITE (LOUT,1206)J,CHAN,EPK,CTS,BKSMBL,GAMAS,PCERR
1206  FORMAT (I5,F9.3,F10.3,F12.0,A1,1PE13.4,0PF9.3,F8.3)
      IF(IOFLG(3).EQ.0)WRITE(LOUT,1206)J,CHAN,EPK,CTS,BKSMBL,GAMAS,PCERR

C     For identification purposes it will be important to have
C     peak arrays ordered by energy.
210   NG = NGAMS+1
      IF(NGAMS .EQ. 0)GO TO 225
      IF(EPK .GE. ENGY(NGAMS))GO TO 225
      DO 215, JK=NGAMS,1,-1
      IF(EPK .GT. ENGY(JK))GO TO 216
215   CONTINUE
      JK = 0
C     Shuffle arrays to make a slot at JK+1
216   DO 220, I=NGAMS,JK+1,-1
      ENGY(I+1) = ENGY(I)
      PKS(I+1) = PKS(I)
      PKE(I+1) = PKE(I)
      CHNL(I+1) = CHNL(I)
      CNTS(I+1) = CNTS(I)
      GAMS(I+1) = GAMS(I)
```

```
      PCTER(I+1) = PCTER(I)
      QFTL(I+1) = QFTL(I)
      BKFLGS(I+1) = BKFLGS(I)
220   CONTINUE
      NG = JK+1

225   CHNL(NG) = CHAN
      ENGY(NG) = EPK
      CNTS(NG) = CTS
      GAMS(NG) = GAMAS
      PCTER(NG) = PCERR
      QFTL(NG) = TQFTL
      BKFLGS(NG) = BKSMBL
      NGAMS = NGAMS+1
      DELT = FWHM
      PKE(NG) = EPK + DELT              !Upper bound of energy window
      IF (NG .EQ. 1)  GOTO 228          !Don't apply to first peak
      DEN = EPK - ENGY(NG-1)            !Distance from previous peak
      IF (DEN .GT. 2.*FWHM)  GOTO 228 !Distance O.K.
      DELT = .5*DEN                     !Use narrower energy window
      PKE(NG-1) = ENGY(NG-1) + DELT   !Re-adjust upper bound of previous peak
228   PKS(NG) = EPK - DELT              !Lower bound
230   CONTINUE

      IF(RUNFLG(4) .EQ. 0) THEN
            IF (LONGPR)WRITE (LOUT,1232)    !Explanatory footnote on photon
            IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1232)
1232  FORMAT(/' * Detector efficiency was assumed to be unity.'//)
      ENDIF

232   IF(IOFLG(4) .NE. 0)GO TO 300
      IF(INTFLG .NE. 0)GO TO 300

      IF (LONGPR)WRITE (LOUT,1230)
      IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1230)
1230  FORMAT(/' CHANNEL  ENERGY   INDEX          YNET      ',
     + 'RECONSTRUCT  RESIDUALS    STD DEV')
      YP = 0.
      DBG = BG(2)-BG(1)
      CALL AVE(1,NDPTS,TAREA,AV,YNET)
      DO 240 I=1,NDPTS
      IICH= I + IST - 1
      EENG = (FLOAT(IICH) - REFCH)* GAIN + REFEN
      YP = YP + YNET(I)
      YR = YNET(I) + BG(1) + DBG * YP/TAREA
      IF (LONGPR)WRITE (LOUT,1240)IICH,EENG,I,YNET(I),YR,RES(I),RM(I)
      IF(IOFLG(3) .NE. 0)GO TO 240
      WRITE(LOUT,1240) IICH,EENG,I,YNET(I),YR,RES(I),RM(I)
1240  FORMAT(I5,F12.3,I6,2X,3F13.2,F10.2)
240   CONTINUE

300   IF(IOFLG(2) .EQ. 0)GO TO 400    !Not finished yet.
C     Prepare a summary after last grouping.
      IF(KTGRP .LE. 1)GO TO 320
      IF (LONGPR)WRITE (LOUT,1300)KTGRP
      IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1300),KTGRP
300   FORMAT(//' SUMMARY OF PEAK RESULTS ON',I3,' GROUPS')

      IF(RUNFLG(4) .NE. 0)    THEN
            IF (LONGPR)WRITE (LOUT,1305)
```

```
                IF(IOFLG(3) .EQ. 0) WRITE(LOUT,1305)
          ELSE
                IF (LONGPR)WRITE (LOUT,1306)
                IF(IOFLG(3) .EQ. 0) WRITE(LOUT,1306)
          ENDIF

1305      FORMAT (/' Peak  Channel    Energy      Counts     ',
     + ' Gammas/min   Pcterr    Qfit')
1306      FORMAT (/' Peak  Channel    Energy      Counts     ',
     + ' Counts/min * Pcterr    Qfit')

          DO 310, I= 1,NGAMS
          IF (LONGPR)WRITE (LOUT,1206)I,CHNL(I),ENGY(I),CNTS(I),BKFLGS(I)
     +   ,GAMS(I),PCTER(I),QFTL(I)
          IF(IOFLG(3) .NE. 0)GO TO 310
          WRITE(LOUT,1206),I,CHNL(I),ENGY(I),CNTS(I),BKFLGS(I),GAMS(I),PCTER(I),
     +   QFTL(I)
310       CONTINUE

          IF(RUNFLG(4) .EQ. 0)THEN
            IF (LONGPR)WRITE (LOUT,1232)
            IF(IOFLG(3) .EQ. 0)WRITE(LOUT,1232)
          ENDIF

C         Convert to fractional error, after adding 0.5% for LDMTRX
C         (Even if evaluation not requested at this time)
320       DO 325 J=1,NGAMS
325       FRCER(J) = .01 * SQRT(.25 + PCTER(J)**2)

400       RETURN
          END
```

```
C
C
      SUBROUTINE P238AB
C
C=======================================================================
C                                P238AB
C=======================================================================
C
C     Version 1.1
C     Author: Wayne Ruhter
C     Date:   14-Oct-88
C     Date:   01-Feb-90  Modified efficiency values
C     Date:   09-Mar-90  Fixed bug in Pa-233 correction to 376-keV peak
C     Date:   14-Mar-90  Added Specific heat calculations
C     Date:   16-Mar-90  Added decay corrections for precision and
C                        bias measurements
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     Abstract:
C
C     This program reads in gammas/min generated by the GRPANL program that
C     are corrected for Cd absorption and absorption by the EP60/61 stainless
C     steel container.  This routine estimates the thickness (g/cm2) of the
C     Pu sample, and small changes in the shape of the detector efficiency
C     curve from that of a nominal 20% coax detector.
C     The calculations are made using the measured intensities of the
C     following 12 peaks: 152, 208, 238, 375, 376, 413, 583, 727,
C     743, 766, 786, 860 keV.  The equations involve 8 unknowns: the apm's of
C     228Th, 238Pu, 239Pu, 241Pu, and 241Am if present, the Pu sample
C     thickness, the change in the slope of the efficiency curve, and the
C     change in the curvature of the efficiency curve.  The equations are
C     non-linear in form, requiring an iterative approach to their solution.
C     The following boundary constaints are also applied:
C     .01 < Pu (g/cm2) < 40.
C
C=======================================================================
C
C
C  The following common blocks are specific to the P238AB program
C
      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPGMS.PMS'
      INCLUDE 'P238COM.CMN'
C
C  The following common blocks are generic to all instruments
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CABUND.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CTIM.CMN'
C
      CHARACTER RSFILE*20
C
      DIMENSION ENRG(12),BR(5,12),PUMU(12),ANSR(6),XSUM(6),INDX(5),
     +  TEMP(12),TEMP2(12),DAT(5,12),XMU(12),TimeF(6),WT(12),
     +  SR(8),SR1(5),GRMS(6),GRMZ(6),PCT1(6),PCT2(6),SPHT(6),SPPOW(5),
     +  ACOEF(325,16)
```

```
C
      DOUBLE PRECISION APHA(8,8)
C
      DATA XMU/1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1.,1./

      DATA PUMU/2.4987,1.192,.866,.325,.32,.268,.152,.114,.112,
     +  .108,.105,.094/
C
      DATA ENRG/152.68,208.,238.6,375.,376.6,413.7,582.9,727.4,
     +  742.8,766.4,786.3,860.5/
C
C                  PU-238 PU-239 PU-241 AM-241 TH-228 PU-240
C
C  Half lives of the isotopes (in minutes)
C
      DATA T1HLF/4.613E07,1.2686E10,7.574E06,2.277E08,1.005E06,
     +  3.44E09/
      DATA SPHT/567.16,1.929,3.339,114.23,0.,7.098/
C
C                      238Pu      239Pu     241Pu    241Am    228Th
      DATA DAT/9.260E-06,        0.,       0.,      0.,      0.,
     +                   0.,        0.,5.33E-06,7.91E-6,     0.,
     +                   0.,1.440E-07,       0.,      0., .435,
     +                   0.,1.570E-05,       0.,      0.,   0.,
     +                   0.,        0.,      0.,1.383E-6,   0.,
     +                   0.,1.489E-05,       0.,      0.,   0.,
     +                   0.,7.680E-09,       0.,1.542E-8, .3058,
     +                   0.,1.240E-09,       0.,1.330E-8,.0665,
     +             5.170E-08,        0.,      0.,      0.,   0.,
     +             2.190E-07,1.14E-07 ,       0.,1.00E-07,.0065,
     +             2.190E-07,       0.0,      0.,      0.0,  0.0,
     +             3.280E-08,        0.,      0.,      0., .011,
     +                   0.,        0.,      0.,8.16E-10,.0453/
C
      DATA INDX/10,6,2,5,7/
C
C-----------------------------------------------------------------------
C
C
C Error Handling
      EXTERNAL ERR_HANDLER
      CALL LIB$ESTABLISH(ERR_HANDLER)
C
C                                !Initialize parameters
      ESLP = 0.0
      ECURV= 0.0
      DELSLP = 0.0
      DELCRV = 0.0
      DELPU = 0.0
      ITER = 0
      ICNT = 0
      OLDEL = 0.0
C     SHAPC(12)=-.342851
C     SHAPC(13)=-.0415522
      SHAPC(12)=-1.190979
      SHAPC(13)=.0204847
                                !Pu flag is initially on.  Pu will be determined
      NPU = 1
C                               !Efficiency slope will be turned on later.
      NSLP = 0
```

```
C                                       !Efficiency curvature will be turned on later.
      NCURV = 0
C                                       !Initially assume 4 isotopic components
      NM = 5
      NUM= 5
      NP = 12
C                                       !Fraction of sample considered to be Pu.
      PUGPSC=3.
      PUFRAC = 1.
      FRACX = 1.-PUFRAC
C
      DO 2 J = 1,12
      DO 2 K = 1,5
C                                       !Transfer branching coef. into arrays.
      BR(K,J) = DAT(K,J)
2     CONTINUE
C
C  Read in gamma intensity information
C
      OPEN (UNIT=INDEV,NAME=STGFIL,TYPE='OLD',ERR=5)
      GO TO 6
5     CLOSE(INDEV)
      WRITE (*,9999)STGFIL
      WRITE (NDEV1,1003)STGFIL
9999  FORMAT(1X,15A1)
1003  FORMAT(1X,A,' File not found')
C     GO TO 4
      ABORT = .TRUE.
      RETURN
6     CALL RDGAMS
C


      A376 = 0.
      A311 = 0.
      KK = 0
      DO 10 J = 1,NGAMS
C     IF (ABS(ENGY(J)-376.6).LE.0.5)THEN
C             A376 = GAMS(J)
C     END IF
      IF (ABS(ENGY(J)-311.9).LE.0.5)THEN
             A311 = GAMS(J)
      END IF
      DO 8 K = 1,NP
      DIFE = ENGY(J) - ENRG(K)
8     IF (ABS(DIFE).LE.0.6)GOTO 9
      GOTO 10
9     KK = KK + 1
      CNTS(KK) = GAMS(J)+1.
C     IF (KK.EQ.7)PCTER(J)=1.
      WT(KK) = 1./(PCTER(J)*CNTS(KK))
      IF (KK .NE.5)THEN
      IF (PCTER(J).GE.2.)ABORT=.TRUE.
      ENDIF
10    CONTINUE
C
C  Check peak areas, if any are zero abort analysis

      IF (ABORT)THEN
      TYPE *,'Insufficient data for analysis, Analysis Aborted!'
      GOTO 500
```

```
      END IF
C                 !Get initial estimate of 228Th, 238Pu, 239Pu, and 241Pu
      ATNUM = 237.
      DO 16 J = 1,NM
      K = INDX(J)
      ELOG = ALOG(ENRG(K))
      PUMAS = PUGPSC * PUMU(K)
      CORR = 1./PUMAS*(1.-EXP(-PUMAS))*EXP(SHAPC(12)
C                                     !Total correction factor
     +  *(ELOG-5.3375)+SHAPC(13)*(ELOG**2-28.4889))
C                                     !Initial answers in dpm.
      GPM = CNTS(K)
      ATNUM = ATNUM+1.
      IF (J.EQ.3)ATNUM =241.
      IF (J.EQ.4)ATNUM = 241.
      IF (J.EQ.5)ATNUM= 228.
      CNVRT= 1.6603E-21*ATNUM*T1HLF(J)/.693147
      ANSR(J) = GPM/ BR(J,K) /CORR
      ANSR(J) = ANSR(J) * CNVRT
C     WRITE (*,*)'ANSR',J,ANSR(J),GPM,BR(J,K),CORR,CNVRT
16    CONTINUE
      CNTS375 = CNTS(4)
C
      IF (LONGPR)WRITE (LOUT,9102)
9102  FORMAT ('     PU      DELPU     ESLP      ',
     +  'DELSLP      ECURV     DELCURV')
      IF (LONGPR)
     +  WRITE (LOUT,9104) PUGPSC,DELPU,ESLP,DELSLP,
     +  ECURV,DELCRV
C
      ELOG = ALOG(311.9)
      PUMAS = PUGPSC * .476
      CORR = 1./PUMAS*(1.-EXP(-PUMAS))*EXP(SHAPC(12)
C                                     !Total correction factor
     +  *(ELOG-5.3375)+SHAPC(13)*(ELOG**2-28.4889))
      CNTS311 = A311/CORR/.38
      ELOG = ALOG(ENRG(5))
      PUMAS = PUGPSC * PUMU(4)
      CORR = 1./PUMAS*(1.-EXP(-PUMAS))*EXP(SHAPC(12)
C                                     !Total correction factor
     +  *(ELOG-5.3375)+SHAPC(13)*(ELOG**2-28.4889))
      CNTS376=CNTS311*CORR*.00655
      CNTS(4) = CNTS375-CNTS376
      DO 15 J=1,16
      DO 15 K=1,325
15    ACOEF(K,J)=0.
      DO 30 J = 1,NP
      PUMAS = PUGPSC *(PUMU(J)*PUFRAC +XMU(J)*FRACX)
      PUEXP = EXP(-PUMAS)
C                                     !Pu absorption
      PUCOF = (1.-PUEXP)/PUMAS
      ELOG = ALOG(ENRG(J))
C                                     !det. eff
      EFF1 = EXP(SHAPC(12)*(ELOG-5.3375)+SHAPC(13)*
     1  (ELOG**2-28.4889))
C     IF (ENRG(J).EQ.152.68)EFF1=EFF1/1.05
      EN100 = ENRG(J) - 200.
C                                     !Change from nominal curve.
      EFF2 = (1. +ESLP*EN100 +ECURV*EN100**2)
C     WRITE (6,*)'EFF',ENRG(J),EFF1,EFF2,PUCOF
```

```
      TEMP(J) = PUCOF *EFF1 *EFF2
      TEMP2(J) = EFF1 *EFF2
C                                     !Composite correction
      CORR = TEMP(J) *WT(J)
C
      ASUM = 0.0
      ATNUM = 237.
      DO 20 K = 1,NM
      ATNUM = ATNUM+1.
      IF (K.EQ.3)ATNUM=241.
      IF (K.EQ.4)ATNUM=241.
      IF (K.EQ.5)ATNUM= 226.
      CNVRT= 1.6603E-21*ATNUM*T1HLF(K)/.693147
C                                     !Calculate coefficient
      ACOEF(J,K) = BR(K,J) *CORR/CNVRT
C                                     !Total dpms.
      ASUM = ASUM + BR(K,J) *ANSR(K)/CNVRT
20    CONTINUE
C
      CORR = CORR * ASUM
      NCMP = NUM
C
C                                     !Pu not to be evaluated
1     IF (NPU .EQ. 0)  GOTO 23
      NCMP = NCMP + 1
      PUG = PUGPSC
C                                     !Pu coefficient
      ACOEF(J,NCMP)=CORR/PUCOF/PUG*(PUEXP-(1.-PUEXP)/PUMAS)
C                                     !Eff. slope not determined
23    IF (NSLP .EQ. 0)  GOTO 30
      NCMP = NCMP + 1
                                      !Eff. slope coefficient
      ACOEF(J,NCMP) = CORR/EFF2 *EN100
C                                     !Eff. curvature not to be determined
      IF (NCURV .NE. 1)  GOTO 30
      NCMP = NCMP + 1
C                                     !Eff. curvature coefficient
      ACOEF(J,NCMP) = ACOEF(J,NCMP-1) *EN100
30    CONTINUE
C
C
      DO 40 J = 1,NCMP
      XSUM(J) = 0.0
      DO 40 K = 1,NCMP
40    APHA(J,K) = 0.0
C
      DO 42 J = 1,NP
      DO 41 K = 1,NCMP
C
      XSUM(K) = XSUM(K) +ACOEF(J,K) * CNTS(J) * WT(J)
C
      DO 41 L = 1,NCMP
41    APHA(K,L) = APHA(K,L) +ACOEF(J,K) *ACOEF(J,L)
42    CONTINUE
C
C                                     !Invert matrix
      CALL MATINV (APHA,8,NCMP,DET)
C
      IF (DET .EQ. 0.)THEN
      WRITE (*,*)'MATRIX IS SINGULAR IN "PU36H2" ROUTINE'
```

```
      ABORT = .TRUE.
      END IF
C
      DO 44 K = 1,NCMP
      ANSR(K) = 0.0
      DM(K) = APHA(K,K)
      DO 43 L = 1,NCMP
C                                  !Calculate answers
43    ANSR(K) = ANSR(K) +XSUM(L) *APHA(K,L)
44    CONTINUE
C
      DO 48 J = 1,NP
      RM(J) = CNTS(J) * WT(J)
      DO 46 K = 1,NCMP
C                                  !Residual
46    RM(J) = RM(J) - ACOEF(J,K) * ANSR(K)
C
48    CONTINUE
C
C  Determine goodness-of-fit
C
      SUMRM = 0.
      DOF = NP - NCMP
      DO 49 J = 1,NP
49    SUMRM = SUMRM + RM(J) * RM(J)
      QFIT = SUMRM/(DOF-1.)
      ICNVRG = 1
      NCMP = NUM
C
50    DELPU = 0.0
C                                  !Pu was not determined
      IF (NPU .EQ. 0)  GOTO 54
      NCMP = NCMP + 1
      DELPU = ANSR(NCMP)
      ADEL = ABS(DELPU)
      DELP = DELPU *(1.-ADEL/(PUGPSC+ADEL+1.))
      IF ((ABS((DELPU+OLDEL)/DELPU)) .LT. .1)  DELP = 0.5*DELP
C                                  !Increment Pu answer.
      PUGPSC = PUGPSC + DELP
      OLDEL = DELPU
C                                  !Check upper limit
      IF (PUGPSC .LT. 100.)  GOTO 52
C                                  !Maximum value
      PUGPSC = 100.
      GOTO 54
C                                  !Check lower limit
52    IF (PUGPSC .GT. .002) GOTO 53
      PUGPSC = .002
      GOTO 54
C                                  !Not converged
53    IF (ABS(DELPU) .GT. .1)  ICNVRG = 0
C
C                                  !Eff. slope not determined
54    IF (NSLP .EQ. 0)  GOTO 55
      NCMP = NCMP + 1
      DELSLP = ANSR(NCMP)
                                   !Absolute value of ESLP change
      ADEL = ABS(DELSLP)
C                                  !Increment slope
      ESLP=ESLP+DELSLP*(1.-ADEL/(ABS(ESLP)+.001+ADEL))
```

```
C
55      DELCRV = 0.0
C                                     !Eff. curvature not determined
        IF (NCURV .NE. 1)  GOTO 56
        NCMP = NCMP + 1
        DELCRV = ANSR(NCMP)
C                                     !Absolute value of ECURV change
        ADEL = ABS(DELCRV)
C                                     !Increment
        ECURV = ECURV + DELCRV*(1.-ADEL/(ABS(ECURV)+.00001 +ADEL))
        IF (ADEL .GT. .000002)  ICNVRG = 0
56      CONTINUE
C
        IF (LONGPR)WRITE (LOUT,9031) QFIT
        IF (LONGPR) WRITE (LOUT,9104) PUGPSC,DELPU,
     +  ESLP,DELSLP, ECURV,DELCRV
9104    FORMAT (2F8.3,4F10.6)
        IF (LONGPR)WRITE (LOUT,9130)
9130    FORMAT (/'   Energy  Gammas   Wt. Fctr. Residual  Tot. Eff.    ',
     +  'Det. Eff.')
        IF (LONGPR)WRITE (LOUT,9090), (ENRG(J),CNTS(J),WT(J),RM(J),TEMP(J),
     +  TEMP2(J), J=1,NP)
9090    FORMAT (0PF9.3,F11.0,F9.6,F8.1,P2E12.3)
C                                     !Has converged
        IF ((NSLP .EQ. 1) .AND. (ICNVRG .EQ. 1)) GOTO 90
C       IF (ICNVRG.EQ.1)GOTO 97
C                                     !O.K. to calc. ESLP
        IF (ABS(DELPU).LT. 0.1)NSLP =1
C
        ITER = ITER + 1
C  Inspect 583-keV residual, if still large, reduce weighting
        IF (ITER.GE.2 .AND. RM(7).GT. 5.0)WT(7) = WT(7)/1000.

        IF (ITER .LT. 20)  GOTO 18
        WRITE (LOUT,9110)
9110    FORMAT (' WARNING: DID NOT CONVERGE IN "P238AB" ROUTINE')
C

90      IF (NCURV .NE. 0) GOTO 92
C       GOTO 92
        NPU = 0
        NCURV = 1
        GOTO 18
92      IF (NPU .EQ. 0) GOTO 96
C                                     !Test Pu boundary conditions
        NPU = 0
        GOTO 18
C
96      IF (NCURV .EQ. 3)  GOTO 97
        NCURV = NCURV + 1
        GOTO 18
C
97      CONTINUE
C
        ATNUM = 237.
        SUM1  = 0.
        SUM2 =  0.
C
C  The results of the least-squares solution of the set of equations used are
C  in units of relative disintegration rates of each of the isotopic
```

```
C   components.  These results are now decay corrected (if requested), and
C   reinterpreted for reporting purposes.
C
        DELT = 0.
        IF (MP .OR. MB)DELT = 1440. *(SDAT-32852.)
      DO 70 J = 1,4
        IF (J.EQ.4)GOTO 72
        ATNUM=ATNUM+1
        IF (J.EQ.3)ATNUM=ATNUM+1
C                                   !Atomic number
72      ER(J) = 100.*SQRT(DM(J)) /ANSR(J)
C                                   !Statistical error
C       GRMS(J) = 1.6603E-21*ANSR(J)*ATNUM*T1HLF(J)/.693147
        GRMS(J) = ANSR(J)
C                                   !Milligrams of J
        DCYNST = .693147 / T1HLF(J)
C                                   !Decay constant
        GRMZ(J) = GRMS(J) * EXP (DCYNST * DELT)
C                                   !Grams at zero time
C                                   !Only add up Pu
        IF (J.GT.3)GOTO 70
        SUM1 = SUM1 + GRMS(J)
C                                   !Sum of grams
        SUM2 = SUM2 + GRMZ(J)
C                                   !Sum at zero time
70    CONTINUE
C
C   Calculate Pu-240/Pu-239 ratio, if PU240<0
C
        C240 = 0
        P240 = 0.
        P240T = 0.
        IF (PU240.LT.0)THEN
                C240FG = .TRUE.
                C240 = 1
                GRMS40 = .07411*GRMS(2) + 2.9804*GRMS(3)
                SUM1 = SUM1 + GRMS40
                DCYNST = .693147 / T1HLF(6)
C                               !Decay constant
                GRMZ40 = GRMS40 * EXP (DCYNST * DELT)
                SUM2 = SUM2 + GRMZ40
                ER240 = .1
        ELSE
C
C   Correct totals for missing Pu-240 abundance
C
                C240FG=.FALSE.
                        IF (PU240.EQ.0)THEN
                        P240T = 2.0
                        ELSE
                        P240T = PU240
                        ENDIF
                P240T = P240T *EXP(.693147 *DELT/ T1HLF(6))
                ER240 = 0.01
                P240  = P240T
        ENDIF
        G239 = GRMS(2)
C
        POW = 0.0
        SGPOW = 0.0
```

```
C
C   ... CORRECT TOTALS FOR PU-242 ABUNDANCE

        IF (PU242.EQ.0)THEN
                P242T = 0.1
        ELSE
                P242T = PU242
        ENDIF
        SUMN = 1. - 0.01 * (P240T +P242T)
        SUM1 = SUM1 / SUMN
        SUM2 = SUM2 / SUMN
        ER242 = 0.1
        P242 = P242T
C
      DO 78 J = 1,4
C                                   !Percent abundance of Jth isotope
        WTPCT(J) = GRMS(J) / SUM1 * 100.
C                                   !Percent abundance at zero time
        PCT2(J) = GRMZ(J) / SUM2 * 100.
C                                  Specific power due to Jth isotope
        SPPOW(J)= .01* SPHT(J)*PCT2(J)
        POW = POW + SPPOW(J)
C                                   !Abundance relative to 239Pu
        GRMS(J) = GRMS(J) / G239
        GRMZ(J) = GRMZ(J) / G239
        IF (MP .OR. MB)THEN
        GRMS(J) = GRMZ(J)
        WTPCT(J) = PCT2(J)
        ENDIF
78      CONTINUE
        IF (C240.EQ.1)P240 = GRMS40/SUM1 * 100.

        SP240 = .01 * P240 * SPHT(6)
        POW = POW + SP240

        SP242 = .01 * P242 * .1146
        POW = POW + SP242
C
C   The following set of calculations propogate the statisical uncertainties
C   of the original results to the quantities being reported.  The treatment
C   acknowledges the covarient dependancy of the normalized isotopic
C   abundances following the procedures reported in J. of INMM, vol. 12, No.2,
C   (1983)
C
        SSPOW = 0.0
        XPCT = (.01*P240 *ER240 *SUM2)**2 + (.01*P242 *ER242 * SUM2)**2
      DO 82 J = 1,4
C                                   !Count and zero time fractions of J component
        FRAC1 = .01 * WTPCT(J)
        FRAC2 = .01 * PCT2(J)
C                 !Start with Pu240 component.  Treat as independent variable.
        ER1(J) = (FRAC1 *P240 *ER240/WTPCT(2))**2 +
     1            (FRAC1 * P242 * ER242/WTPCT(2))**2
        PCT(J) = (FRAC2 *P240 *ER240/PCT2(2))**2 +
     1            (FRAC1 * P242 * ER242/PCT2(2))**2
        IF (J .LT. 4) XPCT = XPCT + APHA(J,J)
        NK = 3
C                                   !If component is Am, set new limit.
        IF (J .EQ. 4) NK = 4
C                                   !Do K-row by L-column integration
```

```
      DO 81 K = 1,NK
C                                  !Assume it is a diagonal element (ie. K=J)
       FK1 = 1.-FRAC1
       FK2 = 1.-FRAC2
       IF (K .NE. J .OR. J .EQ. 4) THEN
                                   !K not= J or component is Am
         FK1 = -FRAC1
         FK2 = -FRAC2
       ENDIF
       IF (K .EQ. 4) THEN
C                                  !Component is Am.
         FK1 = 1.0
         FK2 = 1.0
       ENDIF
C
       DO 80 L = 1,NK
C                                  !Assume it is a diagonal element (ie. L=J)
        FL1 = 1.-FRAC1
        FL2 = 1.-FRAC2
        IF (L .NE. J .OR. J .EQ. 4) THEN
C                                  !L not= J or component is Am
          FL1 = -FRAC1
          FL2 = -FRAC2
        ENDIF
        IF (L .EQ. 4) THEN
C                                  !Component is Am.
          FL1 = 1.0
          FL2 = 1.0
        ENDIF
C                                  !Integrate sigma squared.
        ER1(J) = ER1(J) + FK1 *FL1 *APHA(K,L)
        PCT(J) = PCT(J) + FK2 *FL2 *APHA(K,L)
C                                  !Also integrate the specific power error.
        IF (J .EQ. 4) SGPOW = SGPOW +(SPPOW(K)-POW)**2 *APHA(K,L)
80      CONTINUE
81     CONTINUE
C                                  !Calculate the uncertainties.
       ER1(J) = 100.*SQRT(QFIT*ER1(J))/(SUM1*FRAC1)
       PCT(J) = 100.*SQRT(QFIT*PCT(J))/(SUM2*FRAC2)
82    CONTINUE
C                                  !Uncertainty in specific power
       SGPOW = SGPOW + (SP240 * ER240)**2
       SGPOW = 100.*SQRT(SGPOW) / (SUM2*POW)


C
C
C  ... Open ASCII results file; first create file name from SAMPID

       DO 85 II = 1,20
       IF (SAMPID(II:II).EQ. ' ')GOTO 87
85     CONTINUE
87     IF (II.LE.0)THEN
        TYPE *,'Error in SAMPID; Results file ID = DUMMY.RST'
       RSFILE = 'DUMMY.RST'
       ELSE
       RSFILE = SAMPID
       RSFILE(II:II+4)='.RST'
       ENDIF
       OPEN (UNIT=OKLUN1,FILE=RSFILE,STATUS='NEW',ACCESS='SEQUENTIAL')
       WRITE (LOUT,9015)
```

```
9015     FORMAT(1H1,80(1H*)//)
         WRITE (LOUT,9017)
         WRITE (DKLUN1,9017)
9017     FORMAT(' PU-238 gamma-ray ')
         WRITE (LOUT,9020)SAMPID
         WRITE (DKLUN1,9020)SAMPID
9020     FORMAT(' Assay results for sample: ',A,//)
         WRITE (LOUT,9022)
9022     FORMAT(1X,80(1H*)//)
C
         WRITE (LOUT,9031) QFIT,FWHM
         WRITE (DKLUN1,9031) QFIT,FWHM
9031     FORMAT (' QFIT =',F7.2,15X,'FWHM at 766.4 keV =',F5.1,' keV',/)
         WRITE (LOUT,9032)
         WRITE (DKLUN1,9032)
9032     FORMAT (15X,'ISOTOPIC ANALYSIS AT')
         IF (MB .OR. MP)THEN
         WRITE (LOUT,9042)
         WRITE (DKLUN1,9042)
9042     FORMAT(19X,'ZERO TIME (1/1/1990)  ',
     +   /9X,' RATIOS                WT.PCT.   PCTERR    Milliwatts/(gm to
     +tal Pu)')
         ELSE
         WRITE (LOUT,9033)
         WRITE (DKLUN1,9033)
9033     FORMAT(19X,'COUNT TIME    ',
     +   /9X,' RATIOS                WT.PCT.   PCTERR    Milliwatts/(gm to
     +tal Pu)')
         ENDIF
         K = 0
         DO 105 J=1,3
         K = K +1
         IF (J.EQ.3)THEN
         K = K+1
         IF (C240.EQ.1)THEN
         WRITE (LOUT,9037)P240,SP240
         WRITE (DKLUN1,9037)P240,SP240
         ELSE
         IF (PU240.GT.0)THEN
                 WRITE (LOUT,9036)P240,SP240
                 WRITE (DKLUN1,9036)P240,SP240
         ELSE
                 WRITE (LOUT,9038)P240,SP240
                 WRITE (DKLUN1,9038)P240,SP240
         ENDIF
         ENDIF
         ENDIF
         WRITE (LOUT,9034) (237+K,GRMS(J),WTPCT(J),ER1(J),SPPOW(J))
         WRITE (DKLUN1,9034) (237+K,GRMS(J),WTPCT(J),ER1(J),SPPOW(J))
105      CONTINUE
9024     FORMAT (' Pu',I3,' =',F10.7,10X,F11.5,F7.2,7X,F8.3)
         IF (PU242 .GT. 0.)THEN
           WRITE (LOUT,9039)P242,SP242
           WRITE (DKLUN1,9039)P242,SP242
         ELSE
           WRITE (LOUT,9041)P242,SP242
           WRITE (DKLUN1,9041)P242,SP242
         ENDIF

         WRITE (LOUT,9005)GRMS(4),WTPCT(4),PCT(4),SPPOW(4)
```

```
      WRITE (DKLUN1,9035)GRMS(4),WTPCT(4),PCT(4),SPPOW(4)
9035  FORMAT (' Am241 =',F10.7,10X,F11.5,F7.2,7X,F8.3)
9036  FORMAT (' Pu240 Operator input value ',F11.5,14X,F8.3)
9037  FORMAT (' Pu240 Calculated value ',4X,F11.5,14X,F8.3)
9038  FORMAT (' Pu240 Default value    ',4X,F11.5,14X,F8.3)
   39 FORMAT (' Pu242 Operator input value ',F11.5,16X,1PE10.3)
9041  FORMAT (' Pu242 Default value    ',4X,F11.5,16X,1PE10.3)
      WRITE (LOUT,9040)POW,SGPOW
      WRITE (DKLUN1,9040)POW,SGPOW
9040  FORMAT(45X,' Total =',F8.3,' +/-',F4.2,'%')
      CLOSE(UNIT=DKLUN1)
C
C     Pass results to the system.  Isotopic values passed through CABUND.
      ASANS = POW
      ERRBAR = SGPOW/100.
      WTPCT(5) = P240
      WTPCT(6) = P242
500   RETURN
      END
```

```
C
      SUBROUTINE P238GRP

C                Last edit for VAX 18-Jan-85
C                Edited for Micro VAX II 30-Dec-86
                 Edited for Pu238 SRP project 30-Aug-89
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Originated by R.Gunnink & W.D.Ruhter
C       Modified by J.B.Nidav to use overlays, COMMON blocks pre-prepared
C  by EDIGRP, for easy loading, consistent COMMON definitions in all
C  sections and programs, and optional automatic operation.
C  Further modifications included reading in the peak groupings separately
C  from the other parameters, in order to allow multiple groups & also
C  saving the photon results for later nuclide identification in GEVAL.
C
C       A version of GRPANL modified for SRP Pu238 analysis system.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPGMS.PMS'
      INCLUDE 'GRPEVC.PMS'
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CLUNS.CMN'

      DATA NROW/0/,NCOL/0/,NCMP/0/

      INDEV = 1
      INPDEV = 4
      NDEV = 3
C
C     Read file of pre-edited control info & open the spectral file.
      CALL READ38
      NGAMS = 0        !Init counter.
      IOFLG(5) = 0
50    IOFLG(2) = 0
      CALL RDGRP       !Read in a block of group specifications
      IF(EST .EQ. 0.)IOFLG(2)=1
      IF(IOFLG(2) .NE. 0)GO TO 200     !A summary is needed now.

      CALL CALIB
      KTGRP = KTGRP + 1

      CALL LODGRP

C
C  Look at peak position and FWHM results for the 766.4-keV
C  peak for measurement control purposes.  Report problems.
C
      IF (KTGRP. EQ.9)THEN     !766.4-KEV PEAK
C     TYPE *,'FWHM IN P238GRP',FWHM
         FW766 = FWHM          !Keep this value for later
         IF (ABS(PKPOS(1) - 3063.) .GT. 2.)THEN
         WRITE (LOUT,1100)
1100     FORMAT(' 766.4-keV peak outside limit.  Check stabilizer!')
```

```
            DGFLG(4) = .TRUE.
            ENDIF
            IF (FWHM .GT. 2.4 )THEN
            WRITE (LOUT,1150)
1150      FORMAT(' !WARNING! 766.4-keV energy resolution exceeds
     +  2.4 keV.  Check count rate or detector.')
            DGFLG(3) = .TRUE.
            ENDIF
          ENDIF
          IF(EST .EQ. 0.)GO TO 50          !Abort the illegal group!
          IF(INTFLG .NE. 0)GO TO 200
C
          CALL FIT
          IF (ABORT)RETURN
          IF (KTGRP. EQ. 2)THEN
          SHAPC(1) = FWHM**2 - 208. * SHAPC(2)
C         WRITE (*,*)'FWHM',FWHM,SHAPC(1)
          ENDIF
C
          IF(MFLGS(4) .EQ. 0)GO TO 200
C         Write profiles into ASCII files for later plotting.
C         IF(IOFLG(5) .EQ. 2)CALL PLOTFL

200       CALL OUT

          IF(IOFLG(2) .EQ. 0)GO TO 50
          CLOSE(INPDEV)

          IF(IOFLG(5) .EQ. 0)GO TO 220
          IF(IOFLG(5) .EQ. 2)TYPE 1205
1205      FORMAT(/' Run program HIST to obtain plots from *.TMP files')

C         For peak fitting only, RUNFLG(4)=0, RUNFLG(5)=0, old KALPFG=0
C         For photon calculation after fitting, RUNFLG(4)=1, old KALPFG =1
C         For full evaluation at once RUNFLG(4) & RUNFLG(5)=1, old KALPFG=3
C         For evaluation based on counts only RUNFLG(4)=0,RUNFLG(5)=1
C
220       CALL SVPKS

C         Continue with nuclide evaluation
          PRPK(1) = DCNST(17)
          PRPK(2) = DCNST(18)
          NROW = 0
          NCOL = 0
          NCMP = 0
          NPRGMS = 0
          MGMS = 0
          NSPEC = 0
C  Perform Pu238 analysis on stored results file
C  Pass FWHM at 766 keV to this subroutine for reporting
          FWHM = FW766
          CALL P238AB
          RETURN
          END
```

```
C*************************************************************************
C PASS
C
C       Check the password in order to allow operator to
C       enter supervisory mode

C       CALLING ARGUMENTS:
C            OK = Logical flag indicating validity of operator password
C
C       SSJ  01-Jul-82
C       REV. 08-May-84 lc
C       REV. 19-APR-85 MPK      Remove logical*1 OK
C                               (Redundant with /CPASS/
C       REV. 04-APR-86 WDR      Modified for Micro VAX II
C       REV. 06-AUG-87 RDP      Modified to accept password w/o
C                               echo to screen. All character are
C                               converted to UPPERCASE.
C
C*************************************************************************
C
       SUBROUTINE PASS(OK)
C
       CHARACTER        WORD*10
       INTEGER JSW,BIT12,NOT12,CR,LF
       REAL*8 PWORD
C
       INTEGER*4        STATUS,VKID,LENGTH
       INTEGER*4        SMG$CREATE_VIRTUAL_KEYBOARD
       INTEGER*4        SMG$READ_STRING
       INTEGER*4        SMG$DELETE_VIRTUAL_KEYBOARD
C
       INCLUDE '($TRMDEF)'
C
C  The following common block files are generic for all instruments
C
       INCLUDE 'CPASS.CMN'
C
       EQUIVALENCE (PWORD,WORD)
C
       DATA CR,LF /"15,"12/
C
C
       OK = .FALSE.
C
C  Prompt operator for password
C
       TYPE 1000
 1000  FORMAT(/' You have selected a supervisory option',
     X      ' which requires a password.'/)
C
C  Store up to 8 characters
       STATUS = SMG$CREATE_VIRTUAL_KEYBOARD(VKID,'SYS$INPUT')
       IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
C
       STATUS = SMG$READ_STRING(VKID,
     c                          WORD,
     c                          'Enter password ->',
     c                          10,
     c                          TRM$M_TM_NOECHO+TRM$M_TM_CVTLOW)
       IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
```

```
 20        TYPE 1004
 1004    FORMAT(1X)
C
         STATUS = SMG$DELETE_VIRTUAL_KEYBOARD(VKID)
         IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))

C  Check if operator wanted to escape
C
         IF(WORD(1:1).EQ.' ') GOTO 500
C
C  Compare with all of the passwords
C
         DO 30 I=1,3
           IF(PWORD.EQ.PASSWD(I)) GOTO 40
 30        CONTINUE
C
C  Improper password
C  Inform operator
C
         TYPE 1002
 1002    FORMAT(//10X,'**** IMPROPER PASSWORD - NO ENTRY',
     X        ' TO SUPERVISORY MODE ****'/)
         GO TO 500
C
C  Password is OK - Enter supervisory mode
C
 40      OK = .TRUE.
C
 500     RETURN
         END
```

```
      SUBROUTINE PLOTFL

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     A module to prepare ASCII files for external plot routines.
C                Last edit for VAX 1-Oct-85
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPRFL.PMS'

      DATA PRFNAM/'P','R','F','L','O','O','.','T','M','P',19*' ',0/
      DATA NETFNM/'Y','N','E','T','O','O','.','T','M','P',19*' ',0/
      DATA REMFNM/'R','M','O','O','.','T','M','P',21*' ',0/


      ENCODE(2,1020,PRFNAM(5))KTGRP
      IF(PRFNAM(5) .EQ. ' ')PRFNAM(5) = 'O'
      ENCODE(2,1020,NETFNM(5))KTGRP
      IF(NETFNM(5) .EQ. ' ')NETFNM(5) = 'O'
      ENCODE(2,1020,REMFNM(3))KTGRP
      IF(REMFNM(3) .EQ. ' ')REMFNM(3) = 'O'

1020  FORMAT(I2)

      K= 0

      IF(MFLGS(4) .NE. 2) GO TO 20
C     Transfer fixed peak buffer for plots
      K = 1
      DO 10 I = 1,NDPTS
10    PRFILS(I,KPHF+1) = PRFILS(I,MXNP)

20    OPEN(NDEV,FILE=PRFNAM,STATUS='NEW')

      DO 50 L = 1,KPHF+K
      WRITE(NDEV,1025)L,EXPTID,SAMPID,EST,EEND
1025  FORMAT(' Profile',I3,' for ',2(1X,12A1),'; E range',F9.3,
     +  ' to',F9.3' keV')
      WRITE(NDEV,1030)NDPTS
1030  FORMAT(I4)
      WRITE(NDEV,1040)(XEN(I),PRFILS(I,L),I=1,NDPTS)
1040  FORMAT(F12.2,',',F12.2)
50    CONTINUE
      CLOSE(NDEV)
      TYPE 1055,PRFNAM
1055  FORMAT(' Plot file written: ',30A1)

      DO 70 I = 1,NDPTS
      IF(SVYNET(I) .LE. 0)SVYNET(I) = 0.01
70    CONTINUE

      OPEN(NDEV,FILE=NETFNM,STATUS='NEW')
      WRITE(NDEV,1070)EXPTID,SAMPID,EST,EEND
1070  FORMAT(' Net & profiles for ',2(1X,12A1),'; E range',F9.3,
     +  ' to',F9.3' keV')
      WRITE(NDEV,1030)NDPTS
      WRITE(NDEV,1080)(XEN(I),SVYNET(I),ERYNET(I),I=1,NDPTS)
1080  FORMAT(2(F12.2,','),F10.2)
      CLOSE(NDEV)
      TYPE 1055,NETFNM
```

```
      OPEN(NDEV,FILE=REMFNM,STATUS='NEW')
      WRITE(NDEV,1090)EXPTID,SAMPID,EST,EEND
1090  FORMAT(' RM for',2(1X,12A1),'; E range',F9.3,
     +  ' to',F9.3' keV')
      WRITE(NDEV,1030)NDPTS
      WRITE(NDEV,1095)(RM(I),I=1,NDPTS)
1095  FORMAT(F12.2)
      CLOSE(NDEV)
      TYPE 1055,REMFNM

      RETURN
      END
```

```
      SUBROUTINE PRECHK
C
C This routine performs a preliminary check on the acquisition
C of data to determine proper sample setup and the amount of time
C required for a specified precision

C     Modified by WDR for Pu238 system  Mar-15-90
C
      INTEGER          LID_ERR, SAM_ERR
      INTEGER*4         ISTAT, IMASK, ISTATE
      INTEGER*4        LID_OPEN, LID_CLOSE
      INTEGER*4        START_ROT
      REAL             LEPS_DT_RATIO
      LOGICAL*1        RFLG, EFLG, PROBLEM, STATUS
      CHARACTER        REPLY*1
C
      INCLUDE          'CASSAY.CMN'
      INCLUDE          'CDATE.CMN'
      INCLUDE          'CONTRL.CMN'
      INCLUDE          'CLUNS.CMN'
      INCLUDE          'CVAX.CMN'
C
      INCLUDE          'CONT90.CMN'
      INCLUDE          'CSPECT.CMN'
C
      INCLUDE          'ADAC.CMN'
C
 10   CONTINUE
C     TYPE *,'ENTERING PRECHK'
      TYPE*
      TYPE*,'When sample is ready for ASSAY ,'
      type 1000,' press RETURN ->'
      READ(5,2000) REPLY
 12   LID_ERR = 0
      SAM_ERR = 0
C Test if Lid is closed
C  Check if Door is closed
C     TYPE*,'CHECK LID'
C
      IMASK = 1
      ISTAT = BTSTW(IMASK,LID_OPEN,C_DATA,IOSB,,,)
C     TYPE *,'ISTAT',LID_OPEN
      STATUS = ERROR_CHECK(ISTAT,IOSB)
C
      IMASK = 2
      ISTAT = BTSTW(IMASK,LID_CLOSE,C_DATA,IOSB,,,)
C     TYPE *,'ISTAT',LID_CLOSE
      STATUS = ERROR_CHECK(ISTAT,IOSB)
C
C If Lid is close (and not open) then precede with check.
C
      IF ((LID_CLOSE.EQ.1).AND.(LID_OPEN.EQ.0)) GOTO 20
C
      IF (LID_CLOSE.EQ.LID_OPEN)LID_ERR=1
      IF (LID_ERR.EQ.1) THEN
      TYPE *,'ERROR -- Problem with LID closing or position switches'
      GOTO 15
      ENDIF
 15   TYPE 1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
      READ (5,2000)REPLY
```

```
        IF ((REPLY.NE.'A').AND.(REPLY.NE.'a'))GOTO 12
        ABORT = .TRUE.
        GOTO 900
C
C
  Enable Rotation/scanning  and verify.
 20     IMASK = 512
C!      TYPE*,'START ROTATING'
        ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C Wait a sec
        CALL SLEEP(2.0)
C
C Look at sample size before checking elevator
C
        CALL SIZESAMP
C       TYPE 9998,MATTYP(1)
9998    FORMAT(1X,'MATTYP ',A1)
C
C Test if sample is rotating
C       TYPE*,'CHECK ROTATING'
        CALL ROTMON(RFLG,EFLG)
        CALL ROTMON(RFLG,EFLG)
        IF (.NOT.RFLG) THEN
           TYPE*,'ERROR -- Unable to ROTATE Sample'
           TYPE*
           TYPE1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
           READ(5,2000) REPLY
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 12
           ABORT = .TRUE.
           GOTO 900
        ENDIF
C       TYPE *,'EFLG IN PRECHK',EFLG,MATTYP(1)
        IF (.NOT. EFLG .AND. MATTYP(1) .NE. 'F')THEN
        TYPE *, 'ERROR -- Sample elevator not working.  Check elevator !!'
        TYPE*
           TYPE1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
           READ(5,2000) REPLY
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 12
           ABORT = .TRUE.
           GOTO 900
        ENDIF


C
C
        TYPE*,'PLEASE WAIT 60 SECONDS WHILE ASSAY CHECK IN PROGRESS...'
C
C Clear Time and Erase existing spectrum.
C
        CALL CTIM(IADC,,IERR)
        CALL CHK90('CTIM  ',IERR)
        CALL CDAT(IADC,,IERR)
        CALL CHK90('CDAT  ',IERR)
C
C Set Clock Interval
C Start Acquisition
        PTIME = 60.0
        CALL PSET(IADC,,1,1,PTIME,0.0,0.0,0.0)
C
C Check that sample is rotating
```

```
        CALL ROTMON(RFLG,EFLG)
        IF (.NOT.RFLG) THEN
           TYPE*,'ERROR -- SAMPLE NOT ROTATING. '
           TYPE1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
           READ(5,2000) REPLY
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 20
           ABORT = .TRUE.
           GOTO 900
        ENDIF
C
C Turn on ADC for collection
 49     CALL COLL(IADC,,IERR)
        CALL CHK90('COLL  ',IERR)
C
C Wait Until Acquistion is completed
 50     IWAIT1 = INQADC(IADC,ISTAT,MEMSIZ,IGAIN,IRANGE,IOFF,INPUTS)
        CALL CHK90('INQADC',IWAIT1)
        IF (MATTYP(1) .EQ. 'E')GOTO 51
        CALL SIZESAMP
C       TYPE 9998,MATTYP(1)

C Test if count is completed
 51     IF (IWAIT1.EQ.0) GOTO 100
C
C Check that sample is rotating
 52     CALL ROTMON(RFLG,EFLG)
        IF (.NOT.RFLG) THEN
           CALL ADCOFF(IADC,,IERR)
           TYPE*,'ERROR -- SAMPLE NOT ROTATING. '
           TYPE1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
           READ(5,2000) REPLY
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 49
           ABORT = .TRUE.
           GOTO 900
        ENDIF
C       TYPE *,'EFLG IN PRECHK',EFLG,MATTYP(1)
        IF (.NOT. EFLG .AND. MATTYP(1) .NE. 'F')THEN
           CALL ADCOFF(IADC,,IERR)
        TYPE *, 'ERROR -- Sample elevator not working.  Check elevator !!'
        TYPE*
           TYPE1000,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
           READ(5,2000) REPLY
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 49
           ABORT = .TRUE.
           GOTO 900
        ENDIF
C
        GOTO 50
C Set Live Time (channel #0) and Clock Time (channel #1)
 100    CONTINUE
        CALL FROMCA(IADC,2,0,SPCTRM,IERR,)
        CALL CHK90('FROMCA',IERR)
        TIMELV1 = SPCTRM(1)
        TIMECL1 = SPCTRM(2)
        LEPS_DT_RATIO = (TIMECL1-TIMELV1)/TIMECL1
C
C For assay, get 152-kev peak region and determine county rate
C
        IF (MB .GE. MP)GOTO 900
        CALL FROMCA(IADC,630,800,SPCTRM,IERR,)
```

```
      CALL CHK90('FROMCA',IERR)
      CALL MAXVAL(2,28,JMAX,PHT,SPCTRM)
      N1 = JMAX-12
      IF (N1.LE.0)N1 = 1
      N2 = N1 +2
      M1 = JMAX + 10
      M2 = M1 + 2
      CALL NCNTS(N1,N2,M1,M2,ND,BGD,A152,CR152,AVBG,SPCTRM,SPCTRM)
      CR152 = A152/TIMECL1
      IF (CR152 .LE. 0.)THEN
       TYPE *,' !!! No significant data collected.  Check system !!! '
      TYPE *,'Press ENTER to continue (type A and a <CR> to ABORT) ->'
      READ (5,2000) REPLY
      IF ((REPLY.EQ.'A').OR.(REPLY.EQ.'a')) THEN
         TYPE*,' Abort assay - problem with data collection'
         ABORT = .TRUE.
         GOTO 900
      ENDIF
      END IF
      TIME = 400000./CR152
      Type *,' To count for ',time,' secs type Y'
      Type *,' To count for default time ',timepr,' secs type N'
      Type 1000,' and press RETURN ->'
      READ (5,2000)REPLY
      IF (REPLY .EQ. 'Y')TIMEPR=TIME
      RETURN
C
  900 CALL AEND
      RETURN
C*************************************************************************
 1000 FORMAT(/,1X,A,$)
 2000 FORMAT(A)
      END
```

```
C*********************************************************************
C PRINTL
C
C       Routine specific to gamma ray instruments
C       Print assay
C       Routine specific to the plutonium solution and solids
C       gamma-ray analysis instruments
C
C       SSJ 28-FEB-79
C       Rev. 08-Jun-84 lc
C       Rev. 06-NOV-84 WDR  Revised for plutonium solution gamma-ray
C                           analysis
C
C*********************************************************************
C
        SUBROUTINE PRINT1
C
        LOGICAL*1 SUMMRY
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CBACKG.CMN'
C
C
C
C         IF(LOUT .NE. 6 .AND. (.NOT. FROMDK .OR. ICYCLE .EQ. 1))
C     X      WRITE(LOUT,1998)
 1998   FORMAT('1')
        SUMMRY = .FALSE.
        CALL HEADER(SUMMRY)
C
C  Print final results (short printout)
C
  200   WRITE(LOUT,1044)ASANS,ERRBAR
 1044   FORMAT(/' Result = ',0PG13.4,' +/-',2PF6.1,' %  milliwatts/gm to
     xtal Pu')
C
C
C
  500   WRITE(LOUT,1998)
        CLOSE (LOUT)
  510   RETURN
        END
```

```
C*****************************************************************
C PUT90
C
C       Routine specific to gamma ray instruments
C       Get a spectrum from disk and write it to the Series 90;
        Write chunks of 256 channels at a time
C
C       Calling arguments:
C       Input:  NADC - number of ADC, I
C               IBEGIN - logical flag
C
C       SSJ 28-JUN-79
C
C       Rev. 13-Jun-84 lc
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 06-Jul-87 RDP      Modified for uVAX II w/ADAC Drivers
C
C*****************************************************************
C
        SUBROUTINE PUT90(NADC,IBEGIN)
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CDATE.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
        INCLUDE 'CSPECT.CMN'
        INCLUDE 'CONT90.CMN'
C
C  Open disk file. Allocate channel and free when through
C
 15     CONTINUE
        OPEN(UNIT=DKLUN,NAME=FILNAM,ACCESS='DIRECT',TYPE='OLD',
     X          RECORDSIZE=128,ERR=20)
        IREC = 1
        GO TO 25
C
C  Problem in opening the disk file
C  Inform the user and return
C
 20     CALL ERRMSG(1,FILNAM)
C
C  Free channel
C
        ABORT = .TRUE.
        GO TO 500
C
C  Open ADC for writing

 25     CONTINUE
        CALL OPEN80(NADC,,'OU',,IERR)
        CALL CHK90('OPEN80',IERR)
```

```
        ISTART = IBEGIN
C
C  Read block in from disk
C  Check for dummy variable signalling background appendage
C
   )    READ(DKLUN'IREC)SPCTRM
        IF(SPCTRM(1) .EQ. -999.0)GO TO 100
C
C Write block of 128 channels to Series 90
C
        NUMCHN = 128
        CALL WRIT80(SPCTRM,NUMCHN,ISTART,IERR)
        CALL CHK90('WRIT80',IERR)
        IF(IERR .NE. 0) GO TO 100
        IREC = IREC + 1
        ISTART = ISTART + 128
        GO TO 30
C
C  Close the ADC and the disk file
C
 100    CONTINUE
        CALL CLOS80(IERR)
        CALL CHK90('CLOS80',IERR)
        READ(DKLUN'IREC)DUM,BK,SGBK,TIMEBK,BKDAT,BKTIM,
     X           BKOP,BKSFIL,SAMPID,OPERID,SYSID,ASDATE,ASTIME,
     X           MATTYP(1)
 101    CLOSE(UNIT=DKLUN)
C
C  Check if background has been written at end of data file
C
        IF(DUM .NE. -999.0) NOBACK = .TRUE.
        NBLKS = IREC-1
        TYPE 1006,NBLKS,FILNAM
 1006   FORMAT(/' Spectrum retrieval completed --',
     X           I6,' blocks read from ',15A)
C
C  Set the live data flag to false
C
        LIVE = .FALSE.
C
 500    RETURN
        END
```

```
      SUBROUTINE RDCHNS(ARRAY,NST,NDP,NDEV)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                                   RDCHNS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     Version 2.1
C     Author: Wayne Ruhter
C     Date:   20-AUG-84
C
C     ************************************************************************
C
C     PURPOSE
C
C     Subroutine to read specified spectral region from Canberra
C     spectral data files.
C
C     ************************************************************************
C
C     ARGUMENTS
C
C     INPUT
C     NST  =  Starting channel of region to be read from data file
C     NDP  =  Number of channels to be read
C     NDEV =  Channel I/O number that data file is attached to
C
C     OUTPUT
C     ARRAY = Data array containing region read from data file
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER*2 NST,NDP,NDEV
      DIMENSION ARRAY(2),BUF(128)

      I =1
      NREC = (NST-1)/128
      N1 = NST  - 128 * NREC
      NREC = NREC + 1
10    READ(NDEV'NREC) BUF
15    ARRAY(I) = BUF(N1)
      N1 = N1 + 1
      I = I + 1
      IF (I.GT.NDP)GOTO 30
      IF(N1-128)15,15,20
20    N1 =1
      NREC = NREC + 1
      GOTO 10
30    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RDGAMS

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Read previous output of GRPANL runs on various groupings to
C  reconstruct the photons/min table(s).  This permits combination with
   other spectra before entering the final evaluation phase.
C               Last edit for RSX11 15-Jan-88 R.G.
C               Modified lines written in long printout 2-Feb-90
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INCLUDE 'GRPLGCM.PMS/LIST'
      INCLUDE 'GRPGMS.PMS/LIST'
      INCLUDE 'GRPEVC.PMS/LIST'
C
      INCLUDE 'CONTRL.CMN/LIST'
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      BYTE LINEBF(120),SAMPID(12)

C     MXPKS = 50
C     MXFNM = 30

      ITEMCT = 0
      LINECT = 0
      LFG = 0

30    READ(INDEV,1000,END=310,ERR=300)NCHRS,(LINEBF(K),K=1,NCHRS)
1000  FORMAT(Q,120A1)
      LINECT = LINECT+1
      IF(LONGPR.AND.LINECT.LE.7)WRITE(6,1030)(LINEBF(K),K=1,NCHRS)
1030  FORMAT(120A1)
C     Look for "GRP" in beginning line
      DO 32, J=1,3
      IF(LINEBF(J) .NE. 'S')GO TO 32
      IF(LINEBF(J+1) .NE. 'p')GO TO 32
      IF(LINEBF(J+2) .EQ. 'e')GO TO 33        !Found it
32    CONTINUE
      GO TO 34
33    CONTINUE
CTYPE 1030,(LINEBF(K),K=1,NCHRS)
34    IF(NCHRS .LT. 2)GO TO 30
      IF(LFG .GT. 0)GO TO 50

      DO 35, I = 1,NCHRS
      IF(LINEBF(I) .EQ. '=')GO TO 85
35    CONTINUE
      DO 40, I= 1,9
      IF(LINEBF(I) .EQ. '-')GO TO 45
40    CONTINUE
      GO TO 30

45    IF(LINEBF(I+1) .NE. '-')GO TO 300
      LFG = 1          !Two successive dashes; switch modes.
      IF (INTFLG .NE. 2) GOTO 30
      TYPE 1045                !Running CALDET
1045  FORMAT (' Use this spectrum to calibrate the detector?',
     +  ' (Y/N): ',$)
      ACCEPT 1046, M
```

```
1046    FORMAT (A1)
        IF (M .NE. 'Y') GOTO 320
        GO TO 30

C       Test for "END" flag beginning in cols 1-3
        DO 60, J=1,3
        IF(LINEBF(J) .NE. 'E')GO TO 60
        IF(LINEBF(J+1) .NE. 'N')GO TO 60
        IF(LINEBF(J+2) .EQ. 'D')GO TO 320       !End of spectrum
60      CONTINUE
        GO TO 250

85      IF(LINEBF(I+1) .NE. ' ')STOP ' SYNTAX ERR'
        M = I+2
C       ITEMCT = ITEMCT + 1
        KHKT = NCHRS - M + 1

        GO TO (100,120,130,140,150,160,170,180,185,190,200,210,
     +  220)ITEMCT

100     IF(KHKT .GT. MXFNM-1)KHKT = MXFNM-1
C       DO 102 I = 1,KHKT
C102    IFNAME(I) = LINEBF(M+I-1)
        GO TO 30

120     DO 122 I=1,12
        EXPTID(I) = LINEBF(M)
        M = M + 1
122     CONTINUE
        GO TO 30

        DO 132, I= 1,12
        SAMPID(I) = LINEBF(M+I-1)
132     CONTINUE
        GO TO 30

140     CONTINUE
C140    DO 142, I=1,6
C       DSNAME(I) = LINEBF(M+I-1)
C142    CONTINUE
        GO TO 30

150     DECODE(KHKT,1150,LINEBF(M),ERR=300)DSTTIM,ICTYR
1150    FORMAT(E21.13,I6)
        GO TO 30

160     DECODE(KHKT,1150,LINEBF(M))DTZERO,IZYR
        GO TO 30

170     DECODE(KHKT,1170,LINEBF(M))DELTAT
1170    FORMAT(E15.7)
        GO TO 30

180     DECODE(KHKT,1170,LINEBF(M))TIMEL
        GO TO 30

        DECODE(KHKT,1170,LINEBF(M))SMPWT
        GO TO 30

190     DECODE(KHKT,1170,LINEBF(M))WT100
```

```
        GO TO 30

200     DECODE(KHKT,1170,LINEBF(M))ANSFCT
        GO TO 30
210     DECODE(KHKT,1210,LINEBF(M))PRPK(1),PRPK(2)
 10     FORMAT(2E15.7)
        GO TO 30

220     IF(KHKT .GT. MXFNM-1)KHKT = MXFNM-1
C       DO 222 I = 1,KHKT
C222    LIBNAM(I) = LINEBF(M+I-1)
        GO TO 30

C       Read a line of peak data.

250     DECODE(6,1250,LINEBF)II
1250    FORMAT(I6)
        IF(II .EQ. 0)GO TO 30
        IG = NGAMS+1
        IF(IG .GT. MXPKS)GO TO 305
        DECODE(NCHRS,1252,LINEBF)I,CHANL,ENGY(IG),PKS(IG),PKE(IG),
     +  COUNT,BKSMBL,GAMS(IG),FRCER(IG)
1252    FORMAT(I7,5F13.6,A2,2E15.7)
C       NGAMS = IG
        GO TO 30

300     TYPE 1305,IG
1305    FORMAT(' Reached peak # ',I3)
        STOP ' ERROR READING DATA FILE'
305     STOP ' Too many peaks!'

 )      NDFIL = 1       !EOF seen.

320     RETURN
        END
```

```
      SUBROUTINE RDHDR

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Access SAMPLE file to get sample parameters, etc.
C       Last edit for RSX-11 15-Jan-88 R.G.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        INCLUDE 'GRPLGCM.PMS'
        INCLUDE 'GRPGMS.PMS'
        INCLUDE 'CASSAY.CMN'
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        LOGICAL*1 YESNO
        INTEGER LOAD(2)
        REAL YY(2)
        EQUIVALENCE (LOAD,TEMPR)
        BYTE TDSN(6)
        EQUIVALENCE(Y(140),TDSN)
C       IF(IOFLG(1) .EQ. 0)GO TO 20
        DO 15, J=1,6
15      TDSN(J) = DSNAME(J)

C
C       Get detector-system label from spectral header
C20     DO 21, I=1,6
C21     DSNAME(I) = ' '
C       N = 0
C       DO 22, K = 1,6
C       IF (HDRBYT(90+K) .EQ. ' ')GO TO 22
C       N = N+1
        DSNAME(N) = HDRBYT(90+K)
C22     CONTINUE
C       IF (N .EQ. 0)STOP  ' BAD HEADER FILE--NO DET-SYS LABEL'
C
C       DO 30, I=1,6
C       IF(DSNAME(I) .NE. TDSN(I))GO TO 35
C30     CONTINUE
C       GO TO 40
C35     TYPE 1035,DSNAME,TDSN
C1035   FORMAT(' Warning!! DET-SYS label found in header is: ',6A1/
C     +  '   which disagrees with specification:',6A1/'  Continue? ',$)
C       IF(.NOT. YESNO())STOP
20      CONTINUE
C       TYPE 1036
C1036   FORMAT (' Detector/System Label (Up to 6 chars.): ',$)
C       ACCEPT 1037, DSNAME
C1037   FORMAT (6A)

c40     CALL GETIME (INPDEV,LIVTIM,RLTIME,DEDTIM)
        RLTIME = TIMECL
        LIVTIM = TIMELV
C       Pick up stop time for printouts
        ISTPYR = IDBUF(87)
        DSTPTM = IDBUF(88)
        DSTPTM = DSTPTM + RLHDR(45)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        GAIN = RLHDR(38)
        ZERO = RLHDR(39)
```

```
      DO 110, I=1,12
      EXPTID(I) = HDRBYT(I)
      SMPID(I) =  HDRBYT(I+12)
110   CONTINUE

      RLTIME = RLTIME/60.
      TIMEL = LIVTIM/60.
      IF(RLTIME .EQ. 0)GO TO 125
      DEDTIM = 100. * (RLTIME-TIMEL)/RLTIME
      ICTYR = IDBUF(83)
      DSTTIM = IDBUF(84)
      DSTTIM = DSTTIM + RLHDR(43)

125   IZYR = IDBUF(79)
      DTZERO = IDBUF(80)
      DTZERO = RLHDR(41) + DTZERO
      IF(DTZERO .EQ. 0)GO TO 150

      DELTAT = DSTTIM - DTZERO + RLTIME/2880.
      IDEL = ICTYR - IZYR
      IF(IDEL .EQ. 0)GO TO 150
      DO 140, I=1,IDEL
      DELTAT = DELTAT + 365.
      J = IZYR + I - 1
      IF(0.25*FLOAT(J)-FLOAT(J/4) .EQ. 0)DELTAT = DELTAT+1.
140   CONTINUE

150   IF(RLHDR(27) .NE. 0)GEOM = (RLHDR(27))


      SURFC = RLHDR(10)
      SMPWT = RLHDR(11)
      RHO   = RLHDR(12)
      ANSFCT= RLHDR(13)        !Normalization factor
      WT100 = RLHDR(14)
      VOL = RLHDR(15)
      IF(SMPWT .EQ. 0)GO TO 160
      IF(RHO .NE. 0)GO TO 155
      IF(VOL .NE. 0)RHO = SMPWT/VOL
      IF(RHO .EQ. 0)RHO = 1.
155   IF (SURFC .NE. 0)DEPTH=SMPWT/(SURFC*RHO)
160   NF1 = 0
      DO 165, J=1,5
      IJ = 31+3*(J-1)
      MTLZ(J) = IDBUF(IJ)
      LOAD(1) = IDBUF(IJ+1)
      LOAD(2) = IDBUF(IJ+2)
      IF(TEMPR .EQ. 0)GO TO 165
      NF1=NF1+1
165   CMPOS(J) = TEMPR


      NF2 = 0
      DO 180, J=1,6
      IJ = 57+3*(J-1)
      NABS(J) = IDBUF(IJ)
      LOAD(1) = IDBUF(IJ+1)
      LOAD(2) = IDBUF(IJ+2)
      IF(TEMPR .NE. 0.)NF2 = NF2+1
```

```
180      ABSRB(J) = TEMPR

         DO 185, I = 1,32
185      MCHBUF(I) = HDRBYT(270+I)
         DO 190, I = 1,70
     )   TXTBUF(I) = HDRBYT(I+192)

         RETURN
         END
```

```
      SUBROUTINE READ38
C
C     READIN subroutine modified for SRP Pu238 system
C     Modified by WDR  30-Aug-89
      Modified for WHC by WDR 10-May-91
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     This sub reads files required by GRPANL.  The control
C  and spectral file names are always needed, but other input may be
C  defaulted, thus permitting batch runs of GRPANL.
C     The control file is in the form of preloaded common blocks
C  prepared by the separate program, EDIGRP. This file defines all peak
C  groupings to be processed and also stores some control parameters for
C  the analysis.
C  Run-time options may be used with the spectral file name to control
C  output & analysis features.  Any options given are processed, the spectral
C  file is opened and its header read before returning.
C     RDGRP is called later from the root to read successive copies of
C  the common block GPKS for individual processing.  This file organization
C  permits operation of the spectral analysis program GRPANL under the
C  control of indirect files containing only source file names, if desired.
C     This capability has been expanded to allow use of "generic"
C  control files for which the detector-system and many other parameters
C  are not defined until GRPANL run time.
C
C     Link with UTLIB.
C     LUNs INPDEV & INDEV are defined in the root.
C
C              By J.B.Niday            Last edit 5-Dec-86
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C  The following common is specific to Pu238 instrument

      INCLUDE 'CFILES.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CBACKG.CMN'
C
C  The following common is specific to analysis routines

      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPGMS.PMS'

      PARAMETER MXOPTS=12
      LOGICAL*1 CONFRM,YESNO
      BYTE FILE(18),EFILE(18),FFILE(18),ISFIL(40),CTLFIL(MXFNM)
      BYTE OPTS(MXOPTS),LOPT(MXOPTS)  !Permanent & working option flag arrays
      DIMENSION IWR2(2)
      EQUIVALENCE (IWR2,LIVTIM)       !Note: LIVTIM is REAL*4

      BYTE DETFNM(18),SHPFNM(18),DEFNAM(18),LCLDEF(18),XTN(12)
C     EQUIVALENCE (DETFNM,RM),(SHPFNM,RM(MXFNM+1))
C     Save names for INITLP (files actually used for updates).
      BYTE IBUF(80)
      EQUIVALENCE (WM,IBUF)   !Temp save of file input & options
      EQUIVALENCE (WM(30),LTH)
```

```
      DATA FILE/'P','U','2','3','8','.','C','T','L',9*' '/
      DATA FFILE/'P','U','2','3','8','.','F','C','C',9*' '/
      DATA EFILE/'P','U','2','3','8','.','E','P','6',9*' '/
      DATA DETFNM/'P','U','2','3','8','.','D','E','T',9*0/
      DATA SHPFNM/'P','U','2','3','8','.','S','H','P',9*0/
      DATA OPTS/'I','C','S','E','P','R','G','W','F','U','L','B'/
      DATA XTN/'.','S','H','P','.','D','E','T','.','B','K','G'/
      DATA DEFNAM/'P','U','B',':',13*' ',0/
      DATA LCLDEF/17*' ',0/
      DATA LOPT/'I','F',10*0/
C     CALL VERS(0)
      ISFIL(1) = ' '

C     TYPE 1001
1001  FORMAT(' GRPANL can be used for any or all of:'/
     + '    1) Fitting peaks in peak grouping(s)'/
     + '    2) Converting the counts to photons/min'/
     + '    3) Storing these results in a cumulative file, and/or'/
     + '      Directly evaluating the results for probable source(s).'/
     + '   For this it requests a control file defining the peak groups',
     + ' desired'/'  and some other parametric data.'/
     + '   This file is prepared by the EDIGRP program ')
C
C     Common blocks must be preloaded by EDIGRP

      DO 5, I=1,LENLG
5     NWRDS(I) = 0
      DO 7, I=1,LENGSP
7     IWR2(I) = 0
      DO 8, I=1,2*MXNP+7
      VM(I) = 0.
      ERR(I) = 0.

C10   TYPE 1009
1009  FORMAT(/' Enter specs. of pre-edited GRPANL control file. '/
     + '  For fully interactive control over this run, append a "/I"'/
     + ' and a /F if a Foreign header must be processed',
     + ' (e.g., LANL files.)'/)
10    NKH = MXFNM-1
      NOPT = 2

C  Read default control file PU238.CTL
      NKH = 0
      IF(NKH .NE. 0)GO TO 30
      DO 20 I = 1,18
20    CTLFIL(I) = FILE(I)
      CTLFIL(19) = 0

30    OPEN(UNIT=INDEV,STATUS='OLD',READONLY,FILE=CTLFIL,
     + FORM='UNFORMATTED',ERR=35)
      GO TO 40
35    TYPE 1035,(CTLFIL(I),I=1,40)
1035  FORMAT(' OPEN error on file: ',20A1)
      ABORT = .TRUE.
      GOTO 400
C     GO TO 10
      READ(INDEV,ERR=55)LNLG,LNGS,LNPK
      IF(LNPK .EQ. LENPKS)GO TO 41
      IF(LNPK .NE. 308)GO TO 45
41    IF(LNGS .NE. LENGSP)GO TO 45
```

```
        IF(LNLG .EQ. LENLG)GO TO 50
45      TYPE 1045,LNLG,LNGS,LNPK
1045    FORMAT(' INPUT FILE BLOCKS WRONG LENGTH -USE EDIGRP',3(',',I4))
        CLOSE(INDEV)
        ABORT = .TRUE.
        GO TO 300

50      READ(INDEV,ERR=55)(NWRDS(I),I=1,LNLG)
        READ(INDEV,ERR=55)(IWR2(I),I=1,LNGS)

C       Keep INDEV open for reading peak groups in RDGRP section.
        KTGRP = 0
        GO TO 60
55      TYPE 1055,INDEV,(CTLFIL(I),I=1,MXFNM-1)
        PAUSE ' READ FAILED'
1055    FORMAT(' INDEV= ',I2,', SPEC= ',29A1)
        CLOSE(INDEV)
        GO TO 10
60      CONTINUE

C       C       C       C       C       C       C       C       C
C       Open spectral file for direct access,
C       read its header, and report some contents.

90      NOPT = MXOPTS
        DO 92, I = 1, MXOPTS    !Load options here in case of retrys
92      LOPT(I) = OPTS(I)

C       TYPE 1095
1095    FORMAT(/' Now enter file specs for the spectral data: ',$)

        NKH = MXFNM-1

100     OPEN(UNIT=INPDEV,NAME=filnam,STATUS='OLD',ACCESS='DIRECT',
     +  READONLY,RECORDSIZE=128,ERR=105)
        GO TO 110

105     CLOSE(INPDEV)
        TYPE 1105, filnam
1105    FORMAT(' Cannot open spectrum: ',A15)
        ABORT = .TRUE.
        RETURN
C
C ... Read trailing block of data file to get sample and operator IDs
C     and measurement time and date information
110     irec=33
        READ(inpdev'irec)DUM,BK,SGBK,TIMEBK,BKDAT,BKTIM,BKOP,BKSFIL,
     X  SAMPID,OPERID,SYSID,ASDATE,ASTIME,MATTYP(1)

C       TYPE *,'SDAT',ASDATE,ASTIME,SAMPID,OPERID
C
C  Read default sample information file PU238.FCC
        IF (ISFIL(1) .EQ. ' ') THEN
          DO 62 I = 1,18
62        ISFIL(I) = FFILE(I)
C       TYPE 9999,MATTYP(1)
 99     FORMAT(1X,'MATTYP',A1)
        IF (MATTYP(1).EQ.'E')THEN
          DO 63 I = 1,18
63        ISFIL(I) = EFILE(I)
```

```
          ENDIF
          ISFIL(19) = 0
        ENDIF
        OPEN (UNIT=NDEV,NAME=ISFIL,TYPE='OLD',ACCESS='DIRECT',
     +  READONLY,RECORDSIZE=128,ERR=64)
        GOTO 66
C       CALL CLOSE (NDEV)

        TYPE 1058,(ISFIL(I), I=1,40)
1058    FORMAT (' Cannot open SAMPLE file ',40A1)
        ABORT = .TRUE.
        GOTO 300
66      CALL RDBLK(256,IDBUF,1,NDEV)
        CALL CLOSE (NDEV)
C       C       C       C       C       C       C       C       C       C
C       Update pre-edited control flags vs. any option flags used.
C       First check for nature of header

        IF(NOPT .EQ. 0) GO TO 120

        IF(RUNFLG(2) .EQ. 0)RUNFLG(2) = LOPT(9) !Test for '/F' again.
C
C
112     IF(LOPT(5) .GT. 0) IOFLG(3) = 0
        IF(LOPT(5) .LT. 0) IOFLG(3) = 1
        IF(LOPT(6) .GT. 0) IOFLG(4) = 0
        IF(LOPT(6) .LT. 0) IOFLG(4) = 1
C       IOFLG 3 & 4 are set to inhibit printing &/or residuals tables.

C       Pick up nature & extent of analysis:
        IF(LOPT(2) .LT. 0)RUNFLG(4) = 0
        IF(LOPT(2) .GT. 0)RUNFLG(4) = 1
        IF(LOPT(4) .LT. 0)RUNFLG(5) = 0
        IF(LOPT(4) .GT. 0)RUNFLG(5) = 1
        IF(LOPT(3) .EQ. 0)GO TO 117
        IF(LOPT(3) .LT. 0)THEN
                STGFIL(1) = ' '
                GO TO 117
        ENDIF
C
115      IF((STGFIL(1) .NE. 0).AND.(STGFIL(1) .NE. ' '))GO TO 117
         TYPE 1114
1114    FORMAT(' Please enter a file name for storage of results ',$)
         ACCEPT 1150,(STGFIL(I),I=1,MXFNM-1)

117     IF (LOPT(7) .EQ. 0) GO TO 118
        IF(LOPT(7) .GT. 0) THEN
                IOFLG(5) = 1
                GO TO 120
        ELSE
                IF(IOFLG(5) .EQ. 1) IOFLG(5) = 0
        ENDIF
118     IF((LOPT(8) .NE. 0).AND.(IOFLG(5) .NE. 1))IOFLG(5)=LOPT(8) * 2

        IF(IOFLG(5) .LT. 0)IOFLG(5) = 0

C  ------------------------------------------------------------------
C  Pick up spectral header info including GAIN for remaining initialization.

120       CALL RDHDR            !Read LLNL-N.C.div. standard header
```

```
C
        IF (GAIN.EQ.0) THEN
        TYPE 1089
1089    FORMAT(' Spectrum GAIN (keV/ch) = ',$)
        ACCEPT 1145,GAIN
        ENDIF
C
        IF((GEOM .EQ. 0) .AND. (RUNFLG(4) .NE. 0)) THEN
        TYPE 1122
1122    FORMAT(' Geometry is not given in this header but is',
     +  ' required for efficiency corrections.'/'   Please enter',
     +  ' distance (cm.) from sample midplane to detector window: ',$)
        ACCEPT 1145, GEOM
        IF(GEOM .EQ. 0)GO TO 60          !Another try at option setting

        ENDIF

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       Time to update parameters from standard files or data base,
C  unless the update option was negated (/-U e.g.).
C       First priority is to look for the system default files
C  whose names are based on the DS name found in the header.
C  If no files, look for any previously stored DFILNM &/or SFILNM
C  (provided the control file was made up for the same DS).
C  Third priority is to look for locally stored default names.
C  If still no data, of if request option switch set, accept
C  entry of file names form keyboard.
C

        KK = 4  !VAX default files in N.C. Div. stored in "PUB:"
        NCDS = 0
        DO 124, I = 1,6
        IF(DSNAME(I) .EQ. ' ')GO TO 126
        NCDS = NCDS + 1
124     CONTINUE

C       Now build public & local default names for updates.
126     DO 128, I = 1,NCDS
        LCLDEF(I) = DSNAME(I)
128     DEFNAM(I+KK) = DSNAME(I)

        JDS = KK + NCDS   !Remember length of default name (less XTN)

        DO 130, I = 1,4
        LCLDEF(I+NCDS) = XTN(I+4)
130     DEFNAM(I+JDS) = XTN(I+4)
        IF(RUNFLG(3) .LE. 0)GO TO 134   !Flag set by RDHDR for "new" D-S
          DFILNM(1) = 0
          SFILNM(1) = 0
          BKGDFL(1) = 0
          DO 132,I = 1,24
132       DCNST(I) = 0.


134     IF(LOPT(10) .LT. 0) THEN
                IF(RUNFLG(3) .LE. 0) GO TO 250
   O.K. to skip update when requested, provided no DS change was made.
                TYPE 1340
1340    FORMAT(' Detector-system change normally requires parameter',
     + ' updates.'/'    Proceed without updating? (Y/N) ',$)
```

```
                IF(YESNO()) GO TO 250
        ENDIF

145     OPEN(UNIT=NDEV,FILE=DETFNM,STATUS='OLD',READONLY,ERR=147)

        READ(NDEV,1145,END=146,ERR=147)(DCNST(I),I=1,24)
1145    FORMAT(E15.7)
146     CLOSE(NDEV)
        GO TO 165
147     CLOSE(NDEV)      !Error on OPEN/READ of DETFNM


        TYPE 1480,(DETFNM(I),I=1,MXFNM-1)
1480    FORMAT(' Cannot open/read parameter file: ',29A1)
        DO 149, I = 1,MXFNM-1
149     DETFNM(I) = ' '

        IF(RUNFLG(4) .EQ. 0) GO TO 170  !Counts only.


150     TYPE 1492
1492    FORMAT(' Please enter file specs for detector constants')
        ACCEPT 1150,(DETFNM(I),I=1,MXFNM-1)
1150    FORMAT(29A1)
        IF(DETFNM(1) .NE. ' ') GO TO 145
        IF(RUNFLG(3) .LE. 0) GO TO 165
        TYPE 1340
        IF(.NOT. YESNO()) GO TO 150
        GO TO 165

        DO 162, I=1,18
162     DETFNM(I) = LCLDEF(I)   !Try for a local file next
        DO 163, I = 18,MXFNM-1
163     DETFNM(I) = ' '
        KK = 0
        GO TO 145

C       Calculate approx. detector dead layer
165     CALL CALEFF(DCNST,60.,-2.813,EFF1)
        CALL CALEFF(DCNST,70.,-2.659,EFF2)
        DCNST(16) = (EFF2 - EFF1) * 1.65
        IF(DCNST(16) .LT. 0)DCNST(16) = 0.00000001
C       TYPE 1165,DCNST(16)
1165    FORMAT(' Ge dead layer calculated to be:',1PE10.2)

C       C               C       C       C       C       C       C       C

170     KK = 4          !Reset for public files flag.
        DO 171, I = 1,4
        LCLDEF(I+NCDS) = XTN(I)
171     DEFNAM(I+JDS) = XTN(I)  !Changes default to .SHP

C       IF (LOPT(10) .GT. 0) THEN       !/U flag entered.
C               IF(RUNFLG(1) .GT. 0) GO TO 204  !Interactive.
C               GO TO 190
        ENDIF
180     DO 182, I = 1,JDS+4
182     SHPFNM(I) = DEFNAM(I)
C       DO 185, I = JDS+5,MXFNM-1
```

```
C185    SHPFNM(I) = ' '
190     IF (SFILNM(1).NE.' ')THEN
        DO 192, I = 1,MXFNM
192     SHPFNM(I) = SFILNM(I)
        ENDIF
        KK = 2
C
200     OPEN(UNIT=NDEV,FILE=SHPFNM,STATUS='OLD',READONLY,ERR=203)
        I = 0
201     I = I + 1
        READ(NDEV,1145,END=202,ERR=203)SHAPC(I)
        IF (I .LT. 15) GO TO 201        !Protect from oversize files.
202     CLOSE(NDEV)
        GO TO 250

203     TYPE 1480,(SHPFNM(I),I=1,MXFNM-1)
        CLOSE(NDEV)
        IF(KK .EQ. 4) THEN
                IF((SFILNM(1).NE.0).AND.(SFILNM(1).NE.' '))GO TO 190
        ENDIF
        IF(KK .NE. 0) GO TO 220
        SHPFNM(1) = 0
        IF(RUNFLG(1) .LT. 0) GO TO 250

204     TYPE 1204
1204    FORMAT(' Please enter file specs for shape parameters, if any')
        ACCEPT 1150,(SHPFNM(I),I=1,MXFNM-1)
        IF(SHPFNM(1) .NE. ' ')GO TO 200
        IF(RUNFLG(3) .EQ. 0)GO TO 250   !If = 0: not changing DS
        DO 205, I = 1,15
205     SHAPC(I) = 0.
        IF(KK .EQ. 0) GO TO 250

220     DO 222, I = 1,17
222     SHPFNM(I) = LCLDEF (I)
        DO 223, I = 18,MXFNM-1
223     SHPFNM(I) = ' '
        KK = 0
        GO TO 200


C       The following files could not have been read in by EDIGRP.
250     KK = 4          !Reset for "PUB:" again.

C       IF(LOPT(12) .GT. 0)GO TO 260
C       IF(RUNFLG(3) .EQ. 0) THEN
C         IF((BKGDFL(1) .NE. 0) .AND. (BKGDFL(1) .NE. ' '))GO TO 265
C       ENDIF

C252    DO 253, I = 1,4
C       LCLDEF(I+NCDS) = XTN(I+8)
C253    DEFNAM(I+JDS) = XTN(I+8)
C       DO 254, I = 1,17
C254    BKGDFL(I) = DEFNAM(I)
C       DO 255, I = 18,MXFNM-1
C255    BKGDFL(I) = ' '
        KK = 0          !Flag: Default in public has been tried.
C       GO TO 265
C
C256    CLOSE(NDEV)
```

```
C       TYPE 1256,(BKGDFL(I),I=1,MXFNM-1)
C1256   FORMAT(/' Cannot read file of background peaks:',29A1)
C       IF(KK .GT. 0) GO TO 252          !Try again with default name.
C       IF(KK .EQ. 0) THEN
C         DO 257, I = 1,17
  257   BKGDFL(I) = LCLDEF(I)
C         KK = -1
C         GO TO 265
C       ENDIF
C       DO 259, I = 1,MXFNM-1
C259    BKGDFL(I) = ' '
C       IF(RUNFLG(1) .LE. 0) GO TO 270

C260    TYPE 1260
1260    FORMAT(' Please enter file specs for bkgd peak counts in',
     +  ' detector, if available.'/'    ',$)
C       ACCEPT 1150,(BKGDFL(I),I=1,MXFNM-1)
C       IF(BKGDFL(1) .EQ. ' ')GO TO 270

C265    OPEN(UNIT=NDEV,FILE=BKGDFL,STATUS='OLD',READONLY,ERR=256)

C       J = 1
C266    READ(NDEV,1266,ERR=256,END=268)BKPKEN(J),BKPKCT(J)
1266    FORMAT(2E15.6)
C       J = J+1
C       IF((J .LE. MXBKP) .AND. (BKPKEN(J-1) .NE. 0)) GO TO 266
C268    CLOSE(NDEV)
C       NBKPKS = J - 1


270     IF(RUNFLG(1) .LT. 0)  GO TO 277          !Not interactive
        IF(RUNFLG(3) .EQ. 2) GO TO 273           !DS changed
        IF((STGFIL(1) .EQ. ' ').OR.(STGFIL(1) .EQ. 0)) THEN
                IF(RUNFLG(5) .EQ. 0) GO TO 277  !Libe not needed
        ENDIF
        IF((LIBNAM(1).NE. ' ').AND.(LIBNAM(1).NE.0))GO TO 277
273     TYPE 1273
1273    FORMAT(' A library will be needed for any evaluation.'/
     +  15X,'Please enter file specs here :',$)
        ACCEPT 1150,(LIBNAM(J),J=1,MXFNM-1)


277     CALL DATE(DTE)
        CALL TIME(TBUF)

        APRGN = GAIN
        APRZRO= ZERO     !Preserve original values
C   Transfer temporary control parameter file name from local
C   to common for output.
        DO 278, I=1,MXFNM
278     ICFIL(I) = CTLFIL(I)

        IF(LONGPR)CALL INITLP


C       Read in "absorption" coefficients, when calculating photons.
        This destroys file names of detector consts, shape parameters
C       and absorption coefficients stored by EDISRP.

280     OPEN(UNIT=NDEV,FILE=AFILNM,STATUS='OLD',READONLY,ERR=282)
```

```
      READ(NDEV,1280,END=285,ERR=282)(NSMBL(J),EDGE(J),(COFMU(I,J),
     1  I=1,8),J=1,20)
1280  FORMAT(1X,A2,F7.3,4F12.9/10X,4F12.9)
      GO TO 285
282   TYPE 1282,(AFILNM(I),I=1,MXFNM)
 282  FORMAT(' Error reading: ',30A1)
      IF(NSMBL(19) .NE. 0)GO TO 285    !NSMBL(1) was used for AFILNM.
      CLOSE(NDEV)
      IF(RUNFLG(1) .LT. 0) GO TO 294
      TYPE 1283
1283  FORMAT(' Enter file specs for absorption coeffs ',$)
      ACCEPT 1150,(AFILNM(I),I=1,29)
      IF(AFILNM(1) .EQ. ' ')GO TO 294
      GO TO 280

285   CLOSE(NDEV)

C ***  Adjust certain variables after printing them out in INITLP
294   IF(GEOM .NE. 0)GEOM = GEOM + DCNST(15)
      IF(NF1 .EQ. 0)GO TO 300
      DO 295 K=1,NF1
295   CMPOS(K)=0.005*CMPOS(K)*SMPWT / SURFC

300   RETURN


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ENTRY RDGRP
C     This section is called repeatedly for reading successive blocks
C  of group specifications (Common block /GPKS/)

      READ(INDEV,ERR=390,END=395)(NXFLG(I),I=1,LNPK)
      IF(EST .NE. 0)GO TO 400
390   TYPE 1390,KTGRP+1
1390  FORMAT(/'  ERROR TRYING TO READ GROUP',I3)
395   NP = 0
      EST = 0
      CLOSE (INDEV)

400   RETURN
      END
```

```
C*****************************************************************
C READB
C
C       Read background from disk
C
        Rev. 25-Jun-84 lc
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE READB
C
        LOGICAL*1 FILL
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CPASS.CMN'
        INCLUDE 'CONTRL.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
        INCLUDE 'CONT90.CMN'
C
C  If FROMDK, read background from spectrum file
C
        IF(FROMDK) GO TO 10
C
  Notify operator
C
 1      TYPE 1003,BKFIL
 1003   FORMAT(/' Reading ',A)
C
C  OPEN A DIRECT ACCESS FILE WITH NAME BKFIL
C
        OPEN(UNIT=DKLUN,FILE=BKFIL,ACCESS='DIRECT',TYPE='OLD',
     X          RECORDSIZE=128,ERR=999)
C
C  Read data
C
        READ(DKLUN'1,ERR=999)BKDAT,BKTIM,BKOP,FILL,
     X       BK,SGBK,TIMEBK,PASSWD,BKSFIL
C
C  Close file and acknowledge
C
        CLOSE(UNIT=DKLUN)
        IF(BEGIN)GO TO 500
        TYPE 1000,BKFIL,BKDAT,BKTIM,BKOP
 1000   FORMAT(/,1X,A, ' was written  ',A,2X,A,'  by ',
     X     A12)
        GO TO 500
C
C  Read background file from spectrum file (last record)

 10     TYPE 1003,FILNAM
        OPEN(UNIT=DKLUN,NAME=FILNAM,ACCESS='DIRECT',TYPE='OLD',
     X          RECORDSIZE=128,ERR=997)
```

```
        NRECS = MEMSIZ/128
        NREC = NRECS + 1
C
C  Check if first entry in record is dummy
C
        READ(DKLUN'NREC,ERR=998)DUM
        IF(DUM .NE. -999.0) GO TO 998
        READ(DKLUN'NREC,ERR=998)DUM,BK,SGBK,TIMEBK,BKDAT,BKTIM,
     X                    BKOP,BKSFIL
        CLOSE(UNIT=DKLUN,ERR=11)
 11     TYPE 1000,FILNAM,BKDAT,BKTIM,BKOP
        GO TO 500
C
C  Error in reading from spectrum file
C  Do not abort, but use most recent background read
C
 997    CALL ERRMSG(1,FILNAM)
        GO TO 500
 998    CALL ERRMSG(2,FILNAM)
        CLOSE(UNIT=DKLUN)
        GO TO 500
C
C  Error in reading
C
 999    ABORT = .TRUE.
        CALL ERRMSG(1,BKFIL)
        CALL ERRMSG(102,ARG)
C
 500    RETURN
        END
```

```
C*****************************************************************
C
C READC38       (READC subroutine for PU238 isotopic instrument)
C
C       Read constants for Pu-238  sotopic plutonium gamma-ray
        analysis instrument from disk
C
C       REV. 14-JUN-84  lc
C       REV. 06-SEP-84  CMS
C       REV. 17-DEC-84  WDR     Modified for solid isotopic instrument
C       REV. 04-APR-86  WDR     Modified for Micro VAX II
C       REV. 13-JUL-87  RDP     Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C       Rev. 24-May-91 WDR      Modified for stabilizer settings
C
C*****************************************************************
C
        SUBROUTINE READC
C
        LOGICAL*1 FILL,CDUM(30)
        CHARACTER SNAM*12,SYN*3
C
        INTEGER*2 EMC(21),EDD(41),FILLR(58)
C
C  These common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CMCONS.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CCAL.CMN'
        INCLUDE 'CDIAG.CMN'
C
C  The following common block files are instrument specific for
C  solid isotopic analysis (gamma ray) instruments
C
        INCLUDE 'CONT90.CMN'
        INCLUDE 'P238COM.CMN'
C
C       Specify names of constants files to be read in
C
        DIMENSION OSHAPC(15)
        DATA SNAM /'SHAPC.      '/
        DATA OSHAPC/.04,.00175,-4.,.005,3.,0.,.003,.3,.0,.0,
     +              3.,.0,.0,.0,.0/
C
C       Equivalence dummy arrays to first word of common block
C       Array size = word count of common.
C
        EQUIVALENCE (EMC(1),BITIME),(EDD(1),DIAGMX)
C
C
C  Notify operator
C
        TYPE 1003,CONFIL,COMMNT
 1003   FORMAT(' Reading ',A,1X,A)
C
C  Open a direct access file with name filnam
```

```
C
      OPEN(UNIT=DKLUN,NAME=CONFIL,ACCESS='DIRECT',TYPE='OLD',
     X           RECORDSIZE=128,ERR=999)
C
C  Read data
   Fill = Logical*1 fill to even up word boundary
C  Cdum = Dummy fill argument to blank out comment
C  Fillr = Filler for more parameter arguments
C
      READ(DKLUN'1,ERR=999)FILDAT,FILTIM,FILOP,FILL,CDUM,
     X       FILLR,EMC,EDD,PU240,PU242,NPFLG,NUFLG,TIMEPR,
     X       IZPEAK,IZWNDW,ZARNG,IGPEAK,
     X       IGWNDW,GARNG
C
C  Close file and acknowledge
C
      CLOSE(UNIT=DKLUN)
      IF(BEGIN)GO TO 505
      TYPE 1000,CONFIL,FILDAT,FILTIM,FILOP
 1000 FORMAT(/1X,A,' was written  ',A,2X,A,'  by ',A)
C
C     Read in constants files specific to solid isotopic plutonium
C     gamma-ray analysis
C
C     First get system id to establish file extension
C
505   DO 100 J = 1,15
      IF (CONFIL(J:J) .EQ. '.')GO TO 120
100   CONTINUE
120   K = J
      DO 150 I = 1,3
      SYN(I:I)= CONFIL(K+I:K+I)
150   CONTINUE
C
C     Read in peak shape parameters
C
C     DO 200 J = 1,3
C200  SNAM(J+6:J+6)=SYN(J:J)
C     OPEN (UNIT=DKLUN,NAME=SNAM,TYPE='OLD',ACCESS='SEQUENTIAL',ERR=995)
C     READ (DKLUN,9087,ERR=995) (OSHAPC(I), I = 1,15)
C9087 FORMAT (E15.7)
C     CALL CLOSE (DKLUN)
      GO TO 500
C
C  Error in reading
C
 995  CALL ERRMSG(1,SNAM)
      CALL ERRMSG(102,ARG)
      GO TO 500
999   CALL ERRMSG(1,CONFIL)
      CALL ERRMSG(102,ARG)
C
 500  RETURN
      END
```

```
C*****************************************************************
C  READNM
C
C       Allow operator to select or change a constants
C       filename in a list of constants files

C          LCS 27-FEB-81
C
C       Rev. 16-Jun-84 lc
C       Rev. 19-Feb-85 MPK      Change ACCEPTS to READS with END
C                               and ERR exits
C                               Use YESNO to ask yes/no questions
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 13-Jul-87 RDP      Modifed for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C
C*****************************************************************
C
        SUBROUTINE READNM
C
        LOGICAL*1 ANS,YESNO
        CHARACTER CFLNAM(5)*12,CCMMNT(5)*30
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
C
C

        NEWFIL = .TRUE.
        IF(BEGIN) GO TO 4
C
C  Check if constants file read is still to be used
C
 1      TYPE 1005, CONFIL,COMMNT
 1005   FORMAT(/' The constants file read was: ',A15,1X,A30//
     X     '$Do you want to select another constants file?  (Y/N) -> ')
        READ (5,1050,END=1,ERR=1) ANS
        IF(ANS .NE. 'Y' .AND. ANS .NE. 'N')  GO TO 1
        IF(ANS .NE. 'Y') NEWFIL = .FALSE.
        IF(.NOT. NEWFIL) GO TO 900
C
C  Open file to read names of 5 current constants files
C
4       OPEN(UNIT=DKLUN,FILE=NAMSFL,ACCESS='SEQUENTIAL',
     X       TYPE='OLD',ERR=5)
        GO TO 80
C
C  Problem in open of names file
C
 5      CALL ERRMSG(1,NAMSFL)
        CALL ERRMSG(102,ARG)
        ABORT = .TRUE.
        IF (ABORT) CALL EXIT

C  Read names from file
C
 80     DO 100 I=1,5
```

```
 100      READ(DKLUN,1000) CFLNAM(I),CCMMNT(I)
 1000     FORMAT(A12,A30)
C
C  If just beginning, choose file 1
C
        IF(BEGIN) JANS = 1
        IF(BEGIN) GO TO 280
C
C  Type out the current constants file names & select
C  appropriate file
C
 150    TYPE 1010
 1010   FORMAT(//' The current constants files are:'/)
        DO 200 I=1,5
 200      TYPE 1011,I,CFLNAM(I),CCMMNT(I)
 1011     FORMAT(3X,I1,3X,A12,4X,A30)
C
 210    TYPE 1012
 1012   FORMAT(//'$Enter constants file selected (1-5)',
     X       '  or 6 for new file -> ')
        READ(5,1020,END=150,ERR=150) JANS
 1020   FORMAT(I1)
        IF(JANS.LT.1.OR.JANS.GT.6)GO TO 150
        IF(JANS .EQ. 6) GO TO 400
C
C  Confirm filename
C
  250      TYPE 1030,CFLNAM(JANS)
 1030   FORMAT(/'$Constants file = ',A12,' OK? (Y/N) -> ')
        READ(5,1050,END=250,ERR=250) ANS
        IF(ANS .EQ.'N') GO TO 210
        IF(ANS .NE.'Y') GO TO 250
C
C  Read filename & comment into common block
C
 280    CONFIL=CFLNAM(JANS)
        CONFIL(11:12)='  '
 350      COMMNT=CCMMNT(JANS)
        CLOSE(UNIT=DKLUN,ERR=351)
 351    GO TO 700
C
C  Input new constants file name & comment
C  Attempt to open new file
C
 400    TYPE 1040
 1040   FORMAT(/'$Enter new constants filename -> ')
        READ (5,1050,END=400,ERR=400) CONFIL
 1050   FORMAT(A30)
C
 401    TYPE 1041
 1041   FORMAT(/'$Enter comment (30 char) -> ')
        READ (5,1050,END=401,ERR=401) COMMNT
C
C  Check if new file is permanent & which file it replaces
C
        IF ( .NOT.YESNO('Is this file a permanent replacement')
     1  ) GO TO 600
C
 410    TYPE 1044
 1044   FORMAT(/'$Which file does the new file replace?  (1-5) -> ')
```

```
        READ (5,1020,END=410,ERR=410) JANS
        IF(JANS.LT.1.OR.JANS.GT.5) GO TO 410
C
 420      CFLNAM(JANS)=CONFIL
 421      CCMMNT(JANS)=COMMNT

C  Write new file name list to disk
C
C         REWIND DKLUN
        CLOSE (UNIT=DKLUN)
        OPEN(UNIT=DKLUN,NAME=NAMSFL,ACCESS='SEQUENTIAL',
     X       TYPE='OLD',ERR=5)
        DO 430 I=1,5
 430      WRITE(DKLUN,1100) CFLNAM(I),CCMMNT(I)
 1100     FORMAT(1X,A12,A30)
C
C  Close names file
C
 600    CLOSE(UNIT=DKLUN,ERR=700)
C
C  Try opening the new constants file
C
 700    OPEN(UNIT=DKLUN1,NAME=CONFIL,TYPE='OLD',ERR=999)
        CLOSE(UNIT=DKLUN1,ERR=701)
 701    GO TO 900
C
 999    CALL ERRMSG(1,CONFIL)
        GO TO 150
C
 900    BEGIN = .FALSE.
        RETURN

        END
```

```
C*****************************************************************************
C  RESET38      (RESET subroutine for Pu238 isotopic instrument)
C
C       Resets default settings for Assay options
C       This subroutine is specific to the Pu238 isotopic instrument
        CALLING ARGUMENTS:
C               IARG = 0    Default flags and parameters set
C               IARG = 1    Operator selects flags and parameters
C                           (supervisory option)
C
C       14-MAY-84 LC
C
C       Rev. 14-Jun-84 LC
C       Rev. 11-Sep-84 TES
C       Rev. 29-NOV-84 MPK      Replace list-directed with
C                               formatted I/O
C       Rev. 27-DEC-84 WDR      modified for the solid isotopic instrument
C       Rev. 19-FEB-85 MPK      Change ACCEPTS to READS with END and ERR
C                               exits
C       Rev. 01-Mar-85 WDR      modified to type values at the start
C                               and end only instead of after each entry
C       Rev. 07-MAR-85 MPK      Initialize the VAX comm channel when
C                               VAXCOM may have changed
C       Rev. 07-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 13-Jul-87 RDP      Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C       Rev. 14-Apr-88 WDR      Removed raw data to VAX option
C       Rev. 27-Apr-90 WDR      Removed "VAX communication"
C
C*****************************************************************************
C
        SUBROUTINE RESET(IARG)
C
        LOGICAL*1 RANS
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CCAL.CMN'
C
C  The following common block files are instrument specific for
C  (gamma ray) instruments and specific to the solid isotopic inst.
C
        INCLUDE 'CSPECT.CMN'
        INCLUDE 'CONT90.CMN'
        INCLUDE 'P238COM.CMN'
C
        IF(IARG .EQ. 1)GO TO 100
C
C  Compare name of last constants file read with the default
C  constant file name
C
          IF (CONFIL.EQ.LSTFIL) GOTO 10
          BEGIN = .TRUE.
 10     CONTINUE
C
```

```
C  If file names different, reread constants file
C  (flags reset by constants file)
C
        ABORT = .FALSE.
        IF(.NOT. BEGIN) GO TO 20
        CALL READNM
        IF(ABORT) GO TO 500
        CALL READC
        IF(ABORT) GO TO 500
C
C
C  Reread background file
C
 20     CALL READB
        IF(ABORT) GO TO 500
C
C  Reset parameters to default
C
        CALL INTFLG
C
C  Load last file array with constants file read
C
        DO 40 I=1,15
 40     LSTFIL(I:I) = CONFIL(I:I)
C
        RETURN
C
C  Supervisory option:
C  Print out default options
C
 100    TYPE 1000, LONGPR, WRITNG, TIMEPR, PU240, PU242, CONFIL
 1000   FORMAT(/' 1. Long printout = ',L1/
     X          ' 2. Store data with sample ID filename  = ',L1/
     X          ' 3. Preset assay time = ',F10.0/
     X          ' 4. Pu-240 abundance (Wt.%) = ',F8.4/
     X          ' 5. Pu-242 abundance (Wt.%) = ',F8.4/
     X          ' 6. Constants file = ',A15)
        TYPE *,'   '
C
C  Allow supervisor to change defaults
C
 150    TYPE 1500
 1500   FORMAT(/'$Enter number (1-6) to change (RETURN for no change) -> '
     X    )
        READ (5,1501,END=150,ERR=150) IOPT
 1501   FORMAT(I1)
        IF(IOPT .EQ. 0) GO TO 500
        IF(IOPT .EQ. 1) LONGPR = .NOT. LONGPR
C       IF(IOPT .EQ. 2) VAXCOM = .NOT. VAXCOM
        IF(IOPT .EQ. 2) WRITNG = .NOT. WRITNG
C       IF(IOPT .EQ. 4) RAWDTA = .NOT. RAWDTA
        IF(IOPT .EQ. 3) GO TO 200
        IF(IOPT .EQ. 4) GO TO 300
        IF(IOPT .EQ. 5) GO TO 400
        IF (IOPT. EQ. 6)CALL READNM
        IF(IOPT .EQ. 6) CALL READC
        IF (IOPT.EQ.5)CALL VAXINI
        GO TO 150
C
C  Get new preset time
```

```
C
 200     TYPE 2000
 2000    FORMAT(/'$Enter preset assay time (seconds) -> ')
         READ (5,2001,END=200,ERR=200)TIMEPR
 2001    FORMAT(F10.0)
         IF(TIMEPR .LT. 10000.0) GO TO 150
C
C  Check on excessive assay time
C
 2055    TYPE 2050,TIMEPR
 2050    FORMAT(/'$Is ',F10.0' seconds really correct? (Y/N) -> ')
         READ (5,1001,END=2055,ERR=2055)RANS
 1001    FORMAT(1SA1)
         IF(RANS .EQ. 'Y') GO TO 150
         IF(RANS. EQ. 'N') GO TO 200
         GO TO 2055
C
C  Get new value for Pu-240 abundance
C
300      TYPE 3000
3000     FORMAT(/'$Enter Pu-240 abundance (Wt.%) (0=1.97, -1=Calc.) -> ')
         READ (5,3100,END=300,ERR=300)PU240
3100     FORMAT(F8.3)
         GO TO 150
C
C  Get new value for Pu-242 abundance
C
400      TYPE 4000
4000     FORMAT(/'$Enter Pu-242 abundance (Wt.%) (0=.1) -> ')
         READ (5,3100,END=300,ERR=300)PU242
         GO TO 150

500      TYPE 1000, LONGPR, WRITNG, TIMEPR, PU240, PU242, CONFIL
         TYPE 5000
5000     FORMAT(/'$OK?  (Y/N) ->')
         READ (5,1001,END=400,ERR=400)RANS
         IF (RANS.EQ.'Y' .OR. RANS.EQ.'y')GO TO 600
         GO TO 150
C
 600     RETURN
         END
```

```
C
C*****************************************************************************
C
C  ROTMON
C
C       This subroutine monitors the rotation and the translation motion
C       by the sample elevator of the sample in the PU-238
C       isotopic instrument through an ADAC 1616CCI interface board.
C       Bit 4 in the ADAC data register will be set, if the sample stops
C       rotating.  If rotation is detected to have  stopped, the logical
C       flag "RFLG" is set FALSE.  Bit 3 in the ADAC data register is set,
C       if the sample elevator stops.  If the elevator is detected to
C       have stopped, the logical flag "EFLG" is set to FALSE.
C
C       REV  08-AUG-87  RPD              Using new ADLIBVMS calls.
C
C*****************************************************************************
C
      SUBROUTINE ROTMON(RFLG,EFLG)
C
      INTEGER*4        IMASK, STATUS, RSTATUS, ISTAT
C
C  The following common block file is generic to all instruments
C
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'ADAC.CMN'
C
      LOGICAL*1        RFLG,EFLG
C
      RFLG = .TRUE.
      EFLG = .TRUE.

C  Look at Data register and see if bit 4 is set
C
      IMASK = 16
      ISTAT = BTSTW(1MASK,RSTATUS,C_DATA,IOSB,,,)
      STATUS= ERROR_CHECK(ISTAT,IOSB)
C     WRITE (*,*)'ROTATION',RSTATUS
      IF (RSTATUS.EQ.1) GOTO 400
C
C  Rotation has stopped; set flag to suspend measurement
C
      RFLG = .FALSE.
C
C  Look at Data register and see if bit 3 is set
C
400   IMASK = 8
      ISTAT = BTSTW(IMASK,ESTATUS,C_DATA,IOSB,,,)
      STATUS= ERROR_CHECK(ISTAT,IOSB)
C     WRITE (*,*)'ELEVATOR',ESTATUS
      IF (ESTATUS.EQ.1) GOTO 500
C
C  Elevator has stopped moving; set flag
C
      EFLG = .FALSE.
500   RETURN
      END
```

```
C*****************************************************************
C  WRTLG8       (WRITLG subroutine with Pu-238 isotopic abundance
C               information included in write to assay log file.)
C
C       Write the most recent assay information into the
        assay log file
C
C       The format of a log entry is as follows:
C
C       Var nam Type    Dimens  Meaning
C       ------- ----    ------  -------
C       ASDATE  L*1     9       Assay date (eg 15-JUL-85)
C       ASTIME  L*1     8       Assay time (eg 14:13:27)
C       SAMPID  C*20            Sample ID (ASCII string)
C       OPERID  C*8             Operator ID (ASCII string)
C       MATTYP  L*1     3       Material type (ASCII string)
C       ASTYPE  L*1     2       Assay type (assay = "AS", calib = "CA")
C       LGFILE  C*15            Name of raw data file (null-term ASCII
C                               string, eg SY0:FILNAM.EXT)
C       ICYCLE  I*2     1       Current run cycle
C       NUMCYC  I*2     1       Number of cycles in run
C       ASANS   R*4     1       Assay result
C       ERRBAR  R*4     1       Fractional error in result
C       DGFLG   L*1     10      Data diagnostic flags
C       BIASOK  L*1     1       MC bias run is current and in limit
C       PRECOK  L*1     1       MC precision run is current and in limit
C       BACKOK  L*1     1       Background run is current and in limit
C       BAD     L*1     1       Bad assay flag
C       CONDIT  L*1     1       Conditional assay flag
C       C240FG  L*1     1       Pu-240 value calculated or entered
C       WTPCT   R*4     6       Isotopic weight percent abundances
        PCT     R*4     6       Percent errors on isotopic abundances
C
C       Record 1 of the log file always contains two pointers:
C         LPTR is the record number of the last (most recent) entry.
C         FPTR is the record number of the first (earliest) entry.
C
C       SSJ  22-OCT-82
C
C       REV 04-FEB-83 SSJ
C       REV 25-sep-84 CMS
C       REV 20-Feb-85 CMS       Added error bar and assay type to
C                               log entries
C       Rev 08-Mar-85 MPK       Correct spelling of DGFLG variable
C       Rev 08-Jul-85 MPK       Simplify log format
C                               Include more information in log
C                               Handle all errors in log file
C       Rev 07-Apr-86 WDR       Modified for Micro VAX II
C       Rev 13-Jul-87 RDP       Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C       Rev 20-Mar-90 WDR       Modified for Pu-238 system; changed
C                               LVFLAG to C240FG
C*****************************************************************
C
        SUBROUTINE WRITLG
C

C   The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
```

```
      INCLUDE 'CLOG.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CABUND.CMN'
C
      INTEGER*2 LPTR,FPTR
C
      CHARACTER NOFILE*15
      DATA NOFILE /'None           '/
C
      DATA MAXASY /100/        !max number of log entries (incl pointer)
      DATA LOGREC /44/         !entry length in 4-byte units
C
C  Open the existing log file
C
      OPEN (UNIT=DKLUN,NAME=ASLOG,ACCESS='DIRECT',TYPE='OLD',
     1     RECORDSIZE=LOGREC,ERR=150)
      GO TO 200
C
C  Unsuccessful open on existing file
C  Open a new file
C
  150 IERR=1
      OPEN (UNIT=DKLUN,NAME=ASLOG,ACCESS='DIRECT',TYPE='NEW',
     1     RECORDSIZE=LOGREC,MAXREC=MAXASY,ERR=390)
C
C  Initialize pointers for the new file

      LPTR=2
      FPTR=2
C
C  Write the last record of the file to force allocation
C
      IERR=3
      WRITE (DKLUN'MAXASY,ERR=390) 0
      GO TO 300
C
C  Read pointer to most recent assay in record 1
C
  200 IERR=2
      READ (DKLUN'1,ERR=390) LPTR,FPTR
      IERR=5
      IF (LPTR.LT.2 .OR. LPTR.GT.MAXASY) GO TO 390
      IF (FPTR.LT.2 .OR. FPTR.GT.MAXASY) GO TO 390
C
C  Update pointer to latest (current) run;
C  If the log has caught up with its tail, update the pointer
C  to the earliest run to drop the oldest entry
C
      LPTR=LPTR+1
      IF (LPTR.GT.MAXASY) LPTR=2
      IF (LPTR.NE.FPTR) GO TO 300
      FPTR=FPTR+1
      IF (FPTR.GT.MAXASY) FPTR=2
C
C  Generate file name or 'None' if not writing to disk
```

```
C
  300   IF (.NOT.WRITNG) GO TO 330
  310     LGFILE=FILNAM
        GO TO 350
C
  330   LGFILE=NOFILE
C
  350   CONTINUE
C
C  Update the pointer in record 1
C  Write the most recent assay information in the next record
C  Write every assay taken to the log including autocycle runs
C
        IERR=3
        WRITE (DKLUN'1,ERR=390) LPTR,FPTR
        WRITE (DKLUN'LPTR,ERR=390)
        1       ASDATE, ASTIME, SAMPID, OPERID, MATTYP, ASTYPE,
        2       LGFILE, ICYCLE, NUMCYC,  ASANS, ERRBAR,  DGFLG,
        3       BIASOK, PRECOK, BACKOK,    BAD, CONDIT, C240FG,
        4       (WTPCT(I),PCT(I)/100., I=1,6)
        GO TO 400
C
C  Close file
C
  390   CALL ERRMSG(IERR,ASLOG)
  400   CLOSE (UNIT=DKLUN,ERR=500)
C
  500   RETURN
        END
```

```
C
C*************************************************************************
C
C  WRTDAT
C
      Routine is used to call the subroutine that retrieves
C     the data from the hardware and writes it to disk
C     with the background data appended to the end of the
C     raw data file.  The file name is the sample id up to
C     6 characters with the instrument extension.
C
C
C     13-JUN-84  CMS
C
C*************************************************************************
C
      SUBROUTINE WRTDAT
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
      INCLUDE 'CONT90.CMN'
C
C  Get90 is called for the gamma instruments - a different
C  routine will need to be called for the neutron instruments
C
      CALL GET90(IADC)                          !instrument specific
C
C
      RETURN
      END
```

```
C*****************************************************************
C  WTMCLG
C
C       Write the most recent measurement control information into the
C       measurement control log file

C       The format of a log entry is as follows:
C
C       Var nam Typ Dim Bias/   Meaning
C                       Prec/
C                       backG
C       ------- --- --- -----   -------
C       ASDATE  C*9  -   B/P/G  Assay date (eg 15-JUL-85)
C       ASTIME  C*8  -   B/P/G  Assay time (eg 14:13:27)
C       SAMPID  C*1  20  B/P/G  Sample ID (ASCII string; BKG for backgr)
C       OPERID  C*1  8   B/P/G  Operator ID (ASCII string)
C       MTYP    L*1  2   B      The string "MB"
C                        P      The string "MP"
C                        G      The string "BG"
C       MCRUNS  I*2  1   B      ID of calibration
C                        P      Number of runs in precision test
C       VALUE   R*4  1   B      Assay result
C                        P      Reduced chi squared
C                        G      Background value
C       SIGMA   R*4  1   B      Fractional error in result
C                        P      Measured variance
C                        G      S.D. of background
C       VAR     R*4  1   B      Standard value (for compar with result)
C                        P      Calculated variance
C       ARCHV1  R*4  1   B/P    Archival value (sent to VAX)
C       ARCHV2  R*4  1   B/P    Archival value (sent to VAX)
C       DGFLG   L*1  10  B/P/G  Data diagnostic flags
C       BIASOK  L*1  1   B      Bias run was OK
C       PRECOK  L*1  1   P      Precision run was OK
C       BACKOK  L*1  1   G      Background run was OK
C       MCERR   L*1  1   B/P/G  Error in meas control run
C       MCVAX   L*1  1   B/P    Entry has been sent to VAX
C
C       Record 1 of the log file always contains two pointers:
C         LPTR is the record number of the last (most recent) entry
C         FPTR is the record number of the first (earliest) entry
C
C       CMS 31-MAY-84
C
C       REV 09-AUG-84
C       Rev 27-Mar-85 MPK       Include CRESLT common
C       Rev 09-Jul-85 MPK       Simplify log format
C                               Include more information in log
C                               Handle all errors in log file
C       Rev 07-Apr-86 WDR       Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE WTMCLG
C
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CASSAY.CMN'
```

```
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CMEAS.CMN'
      INCLUDE 'CMC.CMN'
C
      INTEGER*2 LPTR,FPTR
C
      DATA MCMAX /100/          !max number of log entries (incl pointer)
      DATA MCREC /32/           !entry length in 4-byte units
C
C
C  Open the existing log file
C
      OPEN (UNIT=DKLUN,NAME=MCLOG,ACCESS='DIRECT',TYPE='OLD',
     1     RECORDSIZE=MCREC,ERR=150)
      GO TO 200
C
C  Unsuccessful open on existing file
C  Open a new file
C
  150 IERR=1
      OPEN (UNIT=DKLUN,NAME=MCLOG,ACCESS='DIRECT',TYPE='NEW',
     1     RECORDSIZE=MCREC,MAXREC=MCMAX,ERR=390)
C
C  Initialize pointers for the new file
C
      LPTR=2
      FPTR=2
C
C  Write the last record of the file to force allocation
C
      IERR=3
      WRITE (DKLUN'MCMAX,ERR=390) 0
      GO TO 300
C
C  Read pointer to most recent assay in record 1
C
  200 IERR=2
      READ (DKLUN'1,ERR=390) LPTR,FPTR
      IERR=5
      IF (LPTR.LT.2 .OR. LPTR.GT.MCMAX) GO TO 390
      IF (FPTR.LT.2 .OR. FPTR.GT.MCMAX) GO TO 390
C
C  Update pointer to latest (current) run;
C  If the log has caught up with its tail, update the pointer
C  to the earliest run to drop the oldest entry
C
      LPTR=LPTR+1
      IF (LPTR.GT.MCMAX) LPTR=2
      IF (LPTR.NE.FPTR) GO TO 300
      FPTR=FPTR+1
      IF (FPTR.GT.MCMAX) FPTR=2
C
C  Update the pointer in record 1
C  Write the most recent measurement control information in the next
C  record.
C
  300 IERR=3
```

```
      WRITE (DKLUN'1,ERR=390) LPTR,FPTR
      WRITE (DKLUN'LPTR,ERR=390)
     1       ASDATE, ASTIME, SAMPID, OPERID,   MTYP, MCRUNS,
     2        VALUE,  SIGMA,    VAR, ARCHV1, ARCHV2,  DGFLG,
     3       BIASOK, PRECOK, BACKOK,  MCERR,  MCVAX
      GO TO 400
C
C  Close file
C
  390 CALL ERRMSG(IERR,MCLOG)
  400 CLOSE (UNIT=DKLUN,ERR=500)
C
  500 RETURN
      END
```

```
C
C*******************************************************************************
C
C  SIZESAMP
C
C       This subroutine checks the size of the sample in the PU-238
C       isotopic instrument through an ADAC 1616CCI interface board.
C       Bit 2 in the ADAC data register will be set, if the sample
C       is an EP60/61 container.
C
C
C       Version 1.0  8-May-91  WDR
C
C*******************************************************************************
C
        SUBROUTINE SIZESAMP
C
        INTEGER*4       IMASK, STATUS, SSTATUS, ISTAT
C
C  The following common block file is generic to all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'ADAC.CMN'
C
C
C  Look at Data register and see if bit 2 is set
C
        IMASK = 4
        ISTAT = BTSTW(IMASK,SSTATUS,C_DATA,IOSB,,,)
        STATUS= ERROR_CHECK(ISTAT,IOSB)
        WRITE (*,*)'SAMPLE SIZE',SSTATUS
        IF (SSTATUS.EQ.0) THEN
          MATTYP(1) = 'F'
          MATTYP(2) = 'C'
          MATTYP(3) = 'C'
        ELSE
          MATTYP(1) = 'E'
          MATTYP(2) = 'P'
          MATTYP(3) = '6'
        ENDIF
C
C       TYPE 9999,MATTYP(1)
9999    FORMAT(1X,'MATTYP',A1)
500     RETURN
        END
```

```
C*************************************************************************
C  SLEEP
C
C       Sleep for a given number of seconds
C
        REV 05-JAN-83 SSJ
C       REV 08-FEB-84 MPK
C       Rev 04-Feb-85 MPK       No change in code from version
C                               used in MUADC demo.  Supersedes
C                               the version of SLEEP formerly
C                               used in C90.
C       Rev 06-Jul-87 RDP       Modified for uVAX II w/ ADAC Drivers
C       Rev 14-Jul-87 RDP       Sleeping using VAX second counter
C       Rev 19-Jan-88 WDR       Modified to use system wait function
C                               this lets system hibernate and switch to a
C                               real parameter passed to LIB$WAIT
C
C*************************************************************************
C
        SUBROUTINE SLEEP(time_secs)
C
        REAL START,ELAPSE,DONE,time_secs
C
        CALL LIB$WAIT(time_secs)
        RETURN
        END
```

```
      SUBROUTINE STABSET
C*****************************************************************************
C                                                                              *
C                                STABSET                                       *
C                                =======                                       *
C                                                                              *
C*****************************************************************************
C Lawrence Livermore National Laboratory  Safeguards Technology Program      *
C                  Actinide Isotopic Analysis Suite                            *
C*****************************************************************************
C
C       Developed by :  Applications Systems Division
C                               and
C                       Nuclear Chemistry Divison
C
C                       Lawrence Livermore National Laboratory
C                       Livermore, CA. 94550
C
C       This work supported by LLNL Safeguards Technology Program
C       and performed under the auspices of the U.S. Department of
C       ENERGY under Contract W-7405-Eng-48.
C
C*****************************************************************************
C
C       Title      :     stabset
C       Revision   :     1.00
C       Author     :     W.M. Buckley
C       Date       :     4 June 1991
C       Language   :     FORTRAN 77 (system dependencies noted, when known)
C
C*****************************************************************************

C       Abstract   :     this routine sets the values for the control and
C                        status reporting for the Canberra 8232 digital
C                        spectrum stabalizer
C
C*****************************************************************************
C       Calling arguments:       NONE
C
C*****************************************************************************
C
C  Routines called:
C       ci8232_status            function call status routine
C       ci8232_gain_channel      sets the gain peak channel
C       ci8232_gain_width        sets the gain window width
C       ci8232_gain_range        sets the gain analog range
C       ci8232_zero_channel      sets the zero peak channel
C       ci8232_zero_width        sets the zero window width
C       ci8232_zero_range        sets the zero analog range
C       ci8232_report            reports 8232 setup and status
C       ci8232_gain_hold         sets the gain stabilization hold
C       ci8232_zero_hold         sets the zero stabilization hold
C       ci8232_store             stores the 8232 configuration to NVRAM
C       SYS$ASSIGN
C
C  Called by:
        exec238                  isotopic application executive routine

C*****************************************************************************
C       Revision History ( Include date,revision I.D., and details )          *
```

```
C*****************************************************************************
C
C       1.00    6/4/91  wmb     added to Pu-238 application for WHC
C                               Pu-238 system, diagnostics and reporting
C                               turned off
C
C*****************************************************************************


C       C*32    status          string corresponding to status values
C       C*32    status_reset    string corresponding to status values
C       C*12    routine         string corresponding to current lib call
C
C       I*2     dsslun          logical unit for serial port connection to
C                               CI8232
C       I*2     io_chan         I/O channel for serial port connection to
C                               CI8232
C       I*2     stat            status return variable from 8232lib calls
C       I*2     stat_reset      status return variable from 8232lib calls
C       I*2     channel         peak channel value for gain or zero stab
C       I*2     width           window width
C       I*2     range           analog range for zero or gain stab:
C                                       0       for     1/1
C                                       1       for     1/2
C                                       2       for     1/4
C                                       3       for     1/8
C       I*2     option          stabilization mode 0/1/2 for off/hold/on
C
C
C       Definition of COMMON variables:
C
        I*2     IZPEAK          Peak location for zero stabilization
C       I*2     IZWNDW          Peak window for zero stabilization
C       R*4     ZARNG           Analogr range for zero stabilization
C       I*2     IGPEAK          Peak location for gain stabilization
C       I*2     IGWNDW          Peak window for gain stabilization
C       R*4     GARNG           Analog range for gain stabilization


c       IMPLICIT NONE

        COMMON /P238COM/PU240,PU242,
     X                  IZPEAK,IZWNDW,ZARNG,IGPEAK,IGWNDW,GARNG

c       REAL*4          pu240,pu242,zarng,garng
c       INTEGER*2       izpeak,izwndw,igpeak,igwndw

        CHARACTER*32    status
        CHARACTER*32    status_reset
        CHARACTER*12    routine

        INTEGER*2       dsslun
        INTEGER*2       io_chan
        INTEGER*2       stat
        INTEGER*2       stat_reset

        INTEGER*2       channel
        INTEGER*2       width
        INTEGER*2       range
        INTEGER*2       option
```

```
      STRUCTURE /report_struct/
              INTEGER*2 gain_peak
              INTEGER*2 gain_window
              INTEGER*2 gain_range
              INTEGER*2 gain_stab
              INTEGER*2 gain_rate
              INTEGER*2 gain_corr
              INTEGER*2 zero_peak
              INTEGER*2 zero_window
              INTEGER*2 zero_range
              INTEGER*2 zero_stab
              INTEGER*2 zero_rate
              INTEGER*2 zero_corr
      END STRUCTURE

      RECORD /report_struct/ report

C*************************************************************************
C       variable initializations                                         *
C*************************************************************************

      dsslun = 8
      stat = 0
      stat_reset = 0

C  ...announce function
      TYPE 2000

C  ...open terminal devices

      CALL SYS$ASSIGN('TTA1:',io_chan,,)

      OPEN(UNIT=dsslun,FILE='TTA1:',ACCESS='SEQUENTIAL',
     1       FORM='FORMATTED',CARRIAGECONTROL='NONE',STATUS='OLD')

C  ...generate initial report

      routine = 'REPORT'
      CALL ci8232_REPORT(dsslun,io_chan,report,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF
C
C

      WRITE(6,1001)    report.gain_peak,
     1                 report.gain_window,
     2                 report.gain_range


      WRITE(6,1003)    report.zero_peak,
     1                 report.zero_window,
     2                 report.zero_range


      WRITE(6,1002)    report.gain_stab,
     1                 report.gain_rate,
```

```
     2                    report.gain_corr


        WRITE(6,1004)     report.zero_stab,
     1                    report.zero_rate,
     2                    report.zero_corr


C  ...set GAIN settings

        channel = igpeak
        width   = igwndw
        IF(garng.EQ.1.000)THEN
          range = 0
        ELSEIF(garng.EQ.0.500)THEN
          range   = 1
        ELSEIF(garng.EQ.0.250)THEN
          range   = 2
        ELSEIF(garng.EQ.0.125)THEN
          range   = 3
        ENDIF
        option  = 2

        routine = 'GAIN_PEAK'
        CALL ci8232_GAIN_CHANNEL(dsslun,io_chan,channel,stat,status)

        IF(stat.NE.0)THEN
          WRITE(6,1000)routine,stat,status
          stat = stat_reset
          status = status_reset
        ENDIF

        routine = 'GAIN_WINDOW'
        CALL ci8232_GAIN_WIDTH(dsslun,io_chan,width,stat,status)

        IF(stat.NE.0)THEN
          WRITE(6,1000)routine,stat,status
          stat = stat_reset
          status = status_reset
        ENDIF

        routine = 'GAIN_RANGE'
        CALL ci8232_GAIN_RANGE(dsslun,io_chan,range,stat,status)

        IF(stat.NE.0)THEN
          WRITE(6,1000)routine,stat,status
          stat = stat_reset
          status = status_reset
        ENDIF

        routine = 'GAIN_HOLD'
        CALL ci8232_GAIN_HOLD(dsslun,io_chan,option,stat,status)

        IF(stat.NE.0)THEN
          WRITE(6,1000)routine,stat,status
          stat = stat_reset
          status = status_reset
        ENDIF

C  ...set ZERO settings
```

```
      channel = izpeak
      width   = izwndw
      IF(zarng.EQ.1.000)THEN
        range = 0
      ELSEIF(zarng.EQ.0.500)THEN
        range   = 1
      ELSEIF(zarng.EQ.0.250)THEN
        range   = 2
      ELSEIF(zarng.EQ.0.125)THEN
        range   = 3
      ENDIF

      routine = 'ZERO_PEAK'
      CALL ci8232_ZERO_CHANNEL(dsslun,io_chan,channel,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF

      routine = 'ZERO_WINDOW'
      CALL ci8232_ZERO_WIDTH(dsslun,io_chan,width,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF

      routine = 'ZERO_RANGE'
      CALL ci8232_ZERO_RANGE(dsslun,io_chan,range,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF

      routine = 'ZERO_HOLD'
      CALL ci8232_ZERO_HOLD(dsslun,io_chan,option,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF

C  ...store values in NVRAM

      routine = 'STORE'
      CALL ci8232_STORE(dsslun,io_chan,stat,status)

      IF(stat.NE.0)THEN
        WRITE(6,1000)routine,stat,status
        stat = stat_reset
        status = status_reset
      ENDIF
```

```
C
C Assign Data Channel for CCI
C
        DEVICE = 'ZWA1:'
        ISTAT = SYS$ASSIGN(DEVICE,C_DATA,,)
        ESTATUS = ERROR_CHECK(ISTAT,1)
        IF (.NOT.ESTATUS) THEN
           TYPE 100,DEVICE
 100       FORMAT(1X,'Failure to assign channel to ',A5)
           ABORT = .TRUE.
        ENDIF
C
C Assign Data Channel for HCO
        DEVICE = 'ZWB0:'
        ISTAT = SYS$ASSIGN(DEVICE,H_DATA,,)
        ESTATUS = ERROR_CHECK(ISTAT,1)
        IF (.NOT.ESTATUS) THEN
           TYPE 200,DEVICE
 200       FORMAT(1X,'Failure to assign channel to ',A5)
           ABORT = .TRUE.
        ENDIF
5       IMASK = 4096
C     Send HIGH to reset
        ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C     Return to LOW
        ISTAT = BCLRW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
        IMASK = 2048
C     Send HIGH to reset
        ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C     Return to LOW
        ISTAT = BCLRW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C Turn Off Rotation
        IMASK = 1
        ISTAT = BSETW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C Enable LOAD/UNLOAD
        IMASK = 512
        CALL = BCLRW(IMASK,H_DATA,IOSB,,,)
        CALL ERROR_CHECK(ISTAT,IOSB)
C
C  Initialize Series 90
C
        CALL INIT90                          !instrument specific
C
        RETURN
        END
```

```
C
C*********************************************************************
C  STATUS
C       Routine checks the status of the hardware for failures
C       prior to the beginning of an assay or responds to
        the option 'S' entered from the menu list which
C       checks if hardware is busy
C
C       CALLING ARGUMENTS:
C            ITYP = Logical flag indicating the status of hardware
C              ITYP = 0:  hardware okay to run
C              ITYP = 1:  hardware busy checked
C
C       CMS 24-JUL-84
C
C       Rev. 29-Nov-84 MPK      Replace list-directed with
C                               formatted I/O
C       Rev. 07-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 22-Sep-87 RDP      Status now test to see if
C                               the detector is plugged in.
C
C*********************************************************************
C
        SUBROUTINE STATUS(ITYP)
C
        INCLUDE 'ADAC.CMN'
        INTEGER*2       VALUE
        INTEGER*4       SIZE
        CHARACTER       REPLY*1
        DATA SIZE/1/
C
   If argument passed is 0 - check if hardware is okay to run
C
        ABORT = .FALSE.
        IF (ITYP.EQ.1) GOTO 50
C Get Data Word from ADAC
 5      ISTAT = DINW(VALUE,SIZE,C_DATA,IOSB,,,)
C 5     CONTINUE
        ESTATUS= ERROR_CHECK(ISTAT,IOSB)
C       TYPE*,'Cable Value:',VALUE
        IF ((VALUE.EQ.96).OR.
     c      (VALUE.EQ.0).OR.(VALUE.EQ.'FFFF'X)) THEN
C-- Major Error, allow user to take corrective measure
           TYPE 500
 500       FORMAT('  **** HARDWARE ERROR DETECTED ****')
           TYPE*,'ERROR -- Problem with Cables or Connectors'
           TYPE 555,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
 555       FORMAT(1X,A,$)
           READ(5,3000) REPLY
 3000      FORMAT(A1)
           IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 5
           ABORT = .TRUE.
           GOTO 20
        ENDIF
C-- Check for Idle System
   6    IDLE = INQADC(1,ISTAT,MEMSIZ,IGAIN,IRANGE,IOFF,INPUTS)
        TYPE *,'IDLE',IDLE
        IF (IDLE.NE.1)GOTO 15
  14    TYPE 1500
        TYPE 555,'Press RETURN to retry (type A and a <CR> to ABORT) ->'
```

```
        READ(5,3000) REPLY
        IF ((REPLY.NE.'A').AND.(REPLY.NE.'a')) GOTO 6
        ABORT = .TRUE.
        GOTO 20
C
   5    TYPE 1000
1000          FORMAT('  **** STATUS OF HARDWARE OKAY ****')
        GOTO 20
C
C  If argument is 1 check if hardware is busy
C
  50    IDLE = INQADC(1,ISTAT,MEMSIZ,IGAIN,IRANGE,IOFF,INPUTS)
        IF (IDLE .NE. 1) GOTO 55
  54    TYPE 1500
1500    FORMAT(/' **** !! SERIES 90 BUSY !!  MANUAL USE??  ****',/)
        GO TO 20
  55    TYPE 2000
2000    FORMAT('  **** HARDWARE BUSY CHECKED ****')
  20    RETURN
        END
```

```
      SUBROUTINE SVPKS
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      Source file name:  SVPK38.FOR
C
C     Sub to save results of peak fitting in a disk file for use by
C  EVAL in a separate program.  Some of the initial parameters plus
C  results for each peak are appended to any data already stored in
C  the file. This permits optional text editing as well as combining
C  with results from other spectra of the same sample.
C               Last edit 22-Jan-85
C     Modified 4-Mar-90 for Pu-238 isotopic instrument
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INCLUDE 'GRPLGCM.PMS'
      INCLUDE 'GRPGMS.PMS'

      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CASSAY.CMN'

      BYTE DEFSTG(30),EFSMBL,TMPFIL
      DATA DEFSTG /'G','R','P','O','U','T','.','T','M','P',19*' ',0/
C     DATA LP/-1/      !For VAX
      DATA LP/6/
      DATA EFSMBL/' '/

      TMPFIL = 0
      IF(RUNFLG(4) .EQ. 0) EFSMBL = '*'

      IF(STGFIL(1) .EQ. ' ')STGFIL(1) = 0
      IF(STGFIL(1) .NE. 0)GO TO 5
      DO 2, I = 1,30
2     STGFIL(I) = DEFSTG(I)
      TMPFIL = 1
5     OPEN(UNIT=NDEV,NAME=STGFIL,STATUS='NEW',ERR=375)
C     TYPE *,'NDEV = ',NDEV

8     WRITE(NDEV,1060)NGAMS,KTGRP,(DTE(J),J=1,9),(TBUF(I),I=1,8)
1060  FORMAT(//' GRPANL results.  Photons/min for',I3,' peaks in',
     +  I3,' groups (calculated ',9A1,2x,8A1,')')

      LENSF = MXFNM-1
      DO 100, K = 1,MXFNM-1
      IF(FILNAM .EQ. ' ')GO TO 105
100   CONTINUE
      GO TO 110
105   LENSF = K-1
110   WRITE(NDEV,1110),FILNAM
1110  FORMAT(' Spectrum file : ',15A)
      WRITE(NDEV,1015)EXPTID
1015  FORMAT(' Expt = ',12A1)
      WRITE(NDEV,1020)SAMPID
1020  FORMAT(' Sample id = ',20A)
      WRITE(NDEV,1022)DSNAME
1022  FORMAT(' Detector-system = ',6A1)
```

```
      WRITE(NDEV,1024)DSTTIM,ICTYR
1024  FORMAT(' Start time = ',F11.6,',',I6)
      WRITE(NDEV,1025)DTZERO,IZYR
1025  FORMAT(' Zero time = ',F11.6,',',I6)
      WRITE(NDEV,1026)DELTAT
 26   FORMAT(' Decay time (days) = ',1PE15.7)
      WRITE(NDEV,1027)TIMEL
1027  FORMAT(' Live time (mins) = ',1PE15.7)
      WRITE(NDEV,1128)SMPWT
1128  FORMAT(' Sample weight = ',1PE15.7)
      WRITE(NDEV,1028)WT100
1028  FORMAT(' 100% Weight = ',1PE15.7)
      WRITE(NDEV,1029)ANSFCT
1029  FORMAT(' Normalization factor = '1PE15.7)
      WRITE(NDEV,1030)DCNST(17),DCNST(18)
1030  FORMAT(' Pair peak constants = ',1PE15.7,',',E15.7)
      LENLB = MXFNM-1
      DO 40, K = 1,MXFNM-1
      IF(LIBNAM(K) .EQ. 0)GO TO 45
40    CONTINUE
      GO TO 47
45    LENLB = K-1
47    WRITE(NDEV,1047),(LIBNAM(I),I=1,LENLB)
1047  FORMAT(' Library file = ',29A1)


1050  FORMAT(/' PEAK  CHANNEL   ENERGY  PKSTART   PKEND  ',
     +  '  COUNTS(Bkg?)  GAMMAS/MIN    FRCER')
1051  FORMAT(/' PEAK  CHANNEL   ENERGY  PKSTART   PKEND  ',
     +  '  COUNTS(Bkg?)  COUNTS/MIN    FRCER')

      IF(RUNFLG(4) .NE. 0)THEN
            WRITE(NDEV,1050)
      ELSE
            WRITE(NDEV,1051)
      ENDIF
      WRITE(NDEV,1055)
1055  FORMAT(' -----Begin table')
C     IF (LONGPR) THEN
C     IF(RUNFLG(4) .EQ. 0) THEN
C           TYPE 1051
C     ELSE
C           TYPE 1050
C     ENDIF
C     ENDIF
      DO 310, I= 1,NGAMS
C     IF(LONGPR)TYPE 1200,I,CHNL(I),ENGY(I),PKS(I),PKE(I),
C    +  CNTS(I),BKFLGS(I),GAMS(I),FRCER(I)
      WRITE(NDEV,1201)I,CHNL(I),ENGY(I),PKS(I),PKE(I),CNTS(I),
     +  BKFLGS(I),GAMS(I),FRCER(I)

1200  FORMAT(I4,4(',',F8.3),',',F11.0,A1,',',1PE12.4,',',E11.2)
1201  FORMAT(I4,4(',',F8.3),',',F11.0,',',A1,',',1PE12.4,',',E11.2)
310   CONTINUE
      IF(RUNFLG(4) .EQ. 0) WRITE(NDEV,1310)
1310  FORMAT(' *  Detector efficiency was assumed to be unity.')

      WRITE(NDEV,1311),FILNAM
1311  FORMAT(' END output for ',15A)
      CLOSE(NDEV)
```

```
        GO TO 400

370     TYPE 1370
1370    FORMAT(' ERROR opening default storage file')
        GO TO 420

375     TYPE 1375,(STGFIL(I),I=1,MXFNM-1)
1375    FORMAT(' ***** Error in opening disk storage file ',29A1)
        ABORT = .TRUE.
        GO TO 420

400     DO 405 I = LENSF,MXFNM-1
405     IFNAME(I) = ' '
C       IF(TMPFIL .NE. 0) TYPE 1403

C       IF (LONGPR)WRITE(LOUT,1400),FILNAM,(STGFIL(I),I=1,MXFNM-1)
1400    FORMAT(' Results for ',A,'were inserted in file ',15A)
420     RETURN
        END
```

```
C*****************************************************************
C START
C
C       Initialize Program
C
        SSJ 17-AUG-79
C
C       REV. 17-APR-84 LC
C       REV. 13-JUL-84 CMS
C       Rev. 03-Jan-85 MPK      Go to VAX for time sync when
C                               setting date and time
C       Rev. 30-Jan-85 MPK      Include CVAX common
C       Rev. 06-Mar-85 MPK      Initialize VAX communication
C       Rev. 06-Mar-85 MPK      Take out VAX Comm init and time
C                               synch.  They are being moved to
C                               EXEC.
C       Rev. 07-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 07-Aug-87 RDP      Modified for allocation of ADAC
C                               drivers.
C       Rev. 25-Aug-87 RDP      Clear SET lights
C
C*****************************************************************
C
        SUBROUTINE START
C
C
        CHARACTER       DEVICE*5
        INTEGER*4       ISTAT, IMASK
C
C  Required common blocks and declaration for ADAC calls.
C
        INCLUDE 'ADAC.CMN'
        INCLUDE '($SYSSRVNAM)'
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CVAX.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
        INCLUDE 'CSPECT.CMN'
C
C  Initialize logical flags
C
        CALL INTFLG
C
C  Set up beginning
C
        BEGIN = .TRUE.
C
C  Initialize hardware necessary for the instrument - checking
C  that it is operational prior to the measurement beginning

        ABORT = .FALSE.
C
C Allocate Channels for ADAC Control and Monitoring
```

```
C  ...generate final report

        routine = 'REPORT'
        CALL ci8232_REPORT(dsslun,io_chan,report,stat,status)

        IF(stat.NE.0)THEN
          WRITE(6,1000)routine,stat,status
          stat = stat_reset
          status = status_reset
        ENDIF

        WRITE(6,1001)    report.gain_peak,
     1                   report.gain_window,
     2                   report.gain_range


        WRITE(6,1003)    report.zero_peak,
     1                   report.zero_window,
     2                   report.zero_range


        WRITE(6,1002)    report.gain_stab,
     1                   report.gain_rate,
     2                   report.gain_corr


        WRITE(6,1004)    report.zero_stab,
     1                   report.zero_rate,
     2                   report.zero_corr


        CLOSE(UNIT=dsslun)

C*****************************************************************************
C       format declarations                                                   *
C*****************************************************************************

1000    FORMAT(' Error in ',A12,' , ID = ',I3,' , msg is ',A32)
1001    FORMAT(' GAIN Peak = ',I5,'   Window = ',I5,'   Range  = ',I3)
1002    FORMAT(' GAIN Stab = ',I5,'   Rate   = ',I5,'   % corr = ',I3)
1003    FORMAT(' ZERO Peak = ',I5,'   Window = ',I5,'   Range  = ',I3)
1004    FORMAT(' ZERO Stab = ',I5,'   Rate   = ',I5,'   % corr = ',I3)
2000    FORMAT
     1(' 8232 Digital Stabilizer being set from values in PARMTR file')

C*****************************************************************************

        RETURN
        END
```

```
C*************************************************************************
C
C  YESNO
C       Prompt for and accept a yes-or-no answer
C
        Calling argument:
C       PROMPT  The string to be used for prompting.  It is output,
C               with the characters " ?  (Y/N) -> " appended to it.
C
C       Function value (LOGICAL*1):
C       .TRUE. if the answer begins with the character Y
C       .FALSE. if the answer begins with the character N
C
C       If the answer does not begin "Y" or "N", the prompt
C       is repeated until it does.
C
C       Example:
C       If YESNO is called as follows
C               LOGICAL*1 CONT,YESNO
C               CONT=YESNO('Do you want to continue')
C       the following prompt is produced
C                Do you want to continue ?  (Y/N) ->
C
C       04-Feb-85  MPK
C       06-Aug-87  RDP          Convert BYTE string to Character String
C
C*************************************************************************
C
        LOGICAL*1 FUNCTION YESNO(PROMPT)
C*************************************************************************
C
        CHARACTER PROMPT*(*),TAG*10,QUESTION*70,ANS*1
        INTEGER   LOC
C
        DATA TAG/'? (Y/N) ->'/
10      LOC = LEN(PROMPT)+1
C
        QUESTION = PROMPT
        QUESTION(LOC:LOC+10) = TAG
C
        TYPE 1000, QUESTION
 1000   FORMAT(1X,A<LOC+11>,$)
        READ (5,1002,END=10,ERR=10) ANS
 1002   FORMAT (A)
        YESNO=.TRUE.
        IF (ANS.EQ.'Y') RETURN
        YESNO=.FALSE.
        IF (ANS.EQ.'N') RETURN
        GO TO 10
        END
```

```
C
C*****************************************************************************
C                                                                             *
C                                  8232LIB                                    *
C                                  =======                                    *
C                                                                             *
C*****************************************************************************
C Lawrence Livermore National Laboratory  Safeguards Technology Program      *
C                    Actinide Isotopic Analysis Suite                         *
C*****************************************************************************
C
C       Developed by :  Applications Systems Division
C                               and
C                       Nuclear Chemistry Divison
C
C                       Lawrence Livermore National Laboratory
C                       Livermore, CA. 94550
C
C       This work supported by LLNL Safeguards Technology Program
C       and performed under the auspices of the U.S. Department of
C       ENERGY under Contract W-7405-Eng-48.
C
C*****************************************************************************
C
C       Title      :     8232lib
C       Revision   :     1.02
C       Author     :     W.M. Buckley
C       Date       :     5 JUNE 1991
C       Language   :     FORTRAN 77 (system dependencies noted, when known)
C
C*****************************************************************************

C       Abstract   :     this collection of routines provides control and
C                        status reporting for the Canberra 8232 digital
C                        spectrum stabalizer
C
C*****************************************************************************
C
C       Routine          Description
C       ==========================
C
C       timed_read               function to provide QIO-based read
C                                of port to 8232
C       ci8232_status            function call status routine
C       ci8232_gain_channel      sets the gain peak channel
C       ci8232_gain_width        sets the gain window width
C       ci8232_gain_range        sets the gain analog range
C       ci8232_zero_channel      sets the zero peak channel
C       ci8232_zero_width        sets the zero window width
C       ci8232_zero_range        sets the zero analog range
C       ci8232_report            reports 8232 setup and status
C       ci8232_gain_hold         sets the gain stabilization hold
C       ci8232_zero_hold         sets the zero stabilization hold
C       ci8232_store             stores the 8232 configuration to NVRAM

C*****************************************************************************
C       Revision History ( Include date,revision I.D., and details )          *
C ******************************************************************************
C       1.01    6/3/91  wmb      changed delays, reads still don't work
C
```

```
C       1.02    6/5/91  wmb     added QIO routine for port reads
C
C*************************************************************************



C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE timed_read(io_chan,string,length,ier)

C*************************************************************************
C
C       Title     :     timed_read
C       Revision  :     1.00
C
C*************************************************************************
C
C       Abstract  :     this routine performs a timed QIO-based read of the
C                       port to the Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments :     io_chan I       I*2     I/O channel for serial
C                                               communication with 8232
C                       ier     O       I*2     status code(0/1 for OK/error)
C                       length  I/O     I*2     length of read
C                       string  O       C*120   character read buffer
C
C*************************************************************************

C       Routines
C       Called    :     SYS$QIOW
C
C       Libraries
C       Used      :     none
C
C*************************************************************************
C*************************************************************************

C       C*120   string          calling argument
C       I*2     io_chan         calling argument
C       I*2     ier             calling argument
C       I*2     length          calling argument

      IMPLICIT NONE

C  ...declarations required for QIO

      INCLUDE '($IODEF)'

      INTEGER*4       SYS$QIOW
      EXTERNAL        SYS$QIOW

      INTEGER*2       IOSB(4)
      INTEGER*4       timeout
      INTEGER*4       ierr

C  ...........
```

```
      INTEGER*2       io_chan
      INTEGER*2       length
      INTEGER*2       ier

      CHARACTER*120   string

C*************************************************************************
C     variable initializations                                           *
C*************************************************************************

      timeout = 30
      string = ' '

C  ...perform timed read

      ierr = SYS$QIOW(,
     1                %VAL(io_chan),
     1                %VAL(IO$_READLBLK+IO$M_NOECHO+IO$M_TIMED),
     1                IOSB,,,
     1                %REF(string),
     1                %VAL(length),
     1                %VAL(timeout),,,,)

C  ...obtain actual number of characters read

      length = IOSB(2)

C  ...trasfer return status to I*2 variable

      IF(ierr.EQ.1)THEN
        ier = 0
      ELSE
        ier = ierr
      ENDIF

C*************************************************************************

      RETURN
      END

C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_STATUS(lun,io_chan,stat,status)

C*************************************************************************
C
C     Title     :     ci8232_status
C     Revision  :     1.00
C
C*************************************************************************
C
C     Abstract  :     this routine provides status on the previous call
C                     to the Canberra 8232 stabalizer
C
C*************************************************************************
C
C     Arguments :     lun     I       I*2     logical unit for serial
C                                             communication with 8232
```

```
C                       io_chan I       I*2     I/O channel for serial
C                                               communication with 8232
C                       stat    I       I*2     status code(0/1 for OK/error)
C                       status  O       C*32    status string
C
C*************************************************************************
C
C       Routines
C       Called      :       timed_read
C
C       Libraries
C       Used        :       none
C
C*************************************************************************
C*************************************************************************
C
C       C*32    status          calling argument
C       C*32    stat_string(6)  strings corresponding to status values
C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for generating report
C
C       I*2     lun             calling argument
C       I*2     io_chan         calling argument
C       I*2     stat            calling argument
C
C       I*2     length          length of port read
C       I*2     clength         anticipated length of port read
C       C*120   string          character buffer for port read
C       I*2     ier             return status of timed read

        IMPLICIT NONE

        CHARACTER*32    status
        CHARACTER*32    stat_string(12)

        CHARACTER*1     esc
        CHARACTER*1     code

        INTEGER*2       lun
        INTEGER*2       io_chan
        INTEGER*2       stat

        INTEGER*2       length
        INTEGER*2       clength
        INTEGER*2       ier

        CHARACTER*120   string

C*************************************************************************
C       variable initializations                                        *
C*************************************************************************

        esc  = CHAR(27)
        code = 'L'

        stat_string(1) = 'Invalid number of characters'
        stat_string(2) = 'Invalid parameter value'
        stat_string(3) = 'Invalid terminator'
        stat_string(4) = 'No ESC in sequence'
```

```
        stat_string(5) = 'Invalid parameter value'
        stat_string(6) = 'Invalid number of parameters'
        stat_string(10) = '8232 not reachable, check port'
        stat_string(11) = 'error in 8232 communication'
        stat_string(12) = 'error in QIO read'

C   ...initiate report from 8232

        WRITE(lun,1000)esc,code

C   ...read status report from 8232

        clength = 5
        length = clength
        CALL timed_read(io_chan,string,length,ier)
        type *,string(2:clength)
        IF((length.LT.clength-2).OR.ier.NE.0)THEN
          type *, length,clength,ier
          stat = 12
          status = stat_string(stat)
          GOTO 600
        ENDIF
        READ(string,1001,end=500,err=550)stat

C   ...set status string, if stat .NE. 0

        IF(stat.NE.0)THEN
          status = stat_string(stat)
        ELSE
          status = ' '
        ENDIF

        GOTO 600
C   ...no communication established
500     CONTINUE
          stat = 10
          status = stat_string(stat)
        GOTO 600
550     CONTINUE
          stat = 11
          status = stat_string(stat)
600     CONTINUE

C*****************************************************************************
C       format declarations                                                  *
C*****************************************************************************

1000    FORMAT(1X,A1,A1,$)
1001    FORMAT(1X,I1,1X)

C*****************************************************************************

        RETURN
        END



C//////////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

```
      SUBROUTINE ci8232_GAIN_CHANNEL(lun,io_chan,channel,stat,status)

C*************************************************************************
C
C       Title      :     ci8232_gain_channel
C       Revision   :     1.00
C
C*************************************************************************
C
C       Abstract   :     this routine sets the gain channel on the
C                        Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments  :     lun      I        I*2      logical unit for serial
C                                                   communication with 8232
C                        io_chan  I        I*2      I/O channel for serial
C                                                   communication with 8232
C                        channel  I        I*2      gain peak channel
C                        stat     O        I*2      status code(0/1 for OK/error)
C                        status   O        C*32     status string
C
C*************************************************************************
C
C       Routines
C       Called     :     ci8232_status
C
C       Libraries
C       Used       :     none
C
C*************************************************************************
   *************************************************************************

C
C       C*1      esc               ASCII code for escape
C       C*1      code              1 character 8232 code for gain peak channel

C       C*32     status            calling argument
C
C       I*2      lun               calling argument
C       I*2      io_chan           calling argument
C       I*2      stat              calling argument
C
C       I*2      channel           calling argument

        IMPLICIT NONE

        CHARACTER*1      esc
        CHARACTER*1      code

        CHARACTER*32     status

        INTEGER*2        lun
        INTEGER*2        io_chan
        INTEGER*2        stat
        INTEGER*2        channel


C*************************************************************************
C       variable initializations                                        *
```

```
C*******************************************************************************

        esc  = CHAR(27)
        code = '@'

   ...initiate gain peak cahnnel set on 8232

        WRITE(lun,1000)esc,channel,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*******************************************************************************
C       format declarations                                                    *
C*******************************************************************************

1000    FORMAT(1X,A1,I5.5,A1,$)

C*******************************************************************************

        RETURN
        END


C///////////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        SUBROUTINE ci8232_GAIN_WIDTH(lun,io_chan,width,stat,status)

   ********************************************************************************
C
C       Title     :     ci8232_gain_width
C       Revision  :     1.00
C
C*******************************************************************************
C
C       Abstract  :     this routine sets the gain window width on the
C                       Canberra 8232 stabalizer
C
C*******************************************************************************
C
C       Arguments :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                       io_cnan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                       width    I       I*2     gain window width
C                       stat     O       I*2     status code(0/1 for OK/error)
C                       status   O       C*32    status string
C
C*******************************************************************************
C
C       Routines
C       Called    :     ci8232_status
C
        Libraries
C       Used      :     none
C
C*******************************************************************************
```

```
C*************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for gain window width

C       C*32    status          calling argument
C
C       I*2     lun             calling argument
C       I*2     io_chan         calling argument
C       I*2     stat            calling argument
C
C       I*2     width           calling argument

        IMPLICIT NONE

        CHARACTER*1     esc
        CHARACTER*1     code

        CHARACTER*32    status

        INTEGER*2       lun
        INTEGER*2       io_chan

        INTEGER*2       stat
        INTEGER*2       width

C*************************************************************************
C       variable initializations                                         *
C*************************************************************************

        esc  = CHAR(27)
        code = 'A'

C  ...initiate gain window width set on 8232

        WRITE(lun,1000)esc,width,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                              *
C*************************************************************************

1000    FORMAT(1X,A1,I3.3,A1,$)

C*************************************************************************

        RETURN
        END



C'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        SUBROUTINE ci8232_GAIN_RANGE(lun,io_chan,range,stat,status)
```

```
C*****************************************************************
C
C       Title      :       ci8232_gain_range
C       Revision   :      1.00
C
C*****************************************************************
C
C       Abstract   :      this routine sets the gain analog range on the
C                         Canberra 8232 stabalizer
C
C*****************************************************************
C
C       Arguments  :      lun      I        I*2      logical unit for serial
C                                                    communication with 8232
C                         io_chan  I        I*2      I/O channel for serial
C                                                    communication with 8232
C                         range    I        I*2      gain analog range
C                         stat     O        I*2      status code(0/1 for OK/error)
C                         status   O        C*32     status string
C
C*****************************************************************
C
C       Routines
C       Called     :      ci8232_status
C
C       Libraries
C       Used       :      none
C
C*****************************************************************
C*****************************************************************

C
C       C*1       esc               ASCII code for escape
C       C*1       code              1 character 8232 code for gain analog range

C       C*32      status            calling argument
C
C       I*2       lun               calling argument
C       I*2       io_chan           calling argument
C       I*2       stat              calling argument
C
C       I*2       range             calling argument

        IMPLICIT NONE

        CHARACTER*1       esc
        CHARACTER*1       code

        CHARACTER*32      status

        INTEGER*2         lun
        INTEGER*2         io_chan

        INTEGER*2         stat
        INTEGER*2         range

C*****************************************************************
C       variable initializations                                 *
C*****************************************************************
```

```
      esc  = CHAR(27)
      code = 'B'

   ...initiate gain analog range set on 8232

      WRITE(lun,1000)esc,range,code

C  ...read status report from 8232

      CALL ci8232_status(lun,io_chan,stat,status)

C*************************************************************************
C     format declarations                                               *
C*************************************************************************

1000  FORMAT(1X,A1,I1,A1,$)

C*************************************************************************

      RETURN
      END



C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_ZERO_CHANNEL(lun,io_chan,channel,stat,status)

C*************************************************************************
C
C     Title      :    ci8232_zero_channel
C     Revision   :    1.00
C
C*************************************************************************
C
C     Abstract   :    this routine sets the zero channel on the
C                     Canberra 8232 stabalizer
C
C*************************************************************************
C
C     Arguments  :    lun      I      I*2     logical unit for serial
C                                             communication with 8232
C                     io_chan  I      I*2     I/O channel for serial
C                                             communication with 8232
C                     channel  I      I*2     zero peak channel
C                     stat     O      I*2     status code(0/1 for OK/error)
C                     status   O      C*32    status string
C
C*************************************************************************
C
C     Routines
      Called     :    ci8232_status
C
C     Libraries
C     Used       :    none
```

```
C
C*************************************************************************
C*************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for zero peak channel

C       C*32    status          calling argument
C
C       I*2     lun             calling argument
C       I*2     io_chan         calling argument
C       I*2     stat            calling argument
C
C       I*2     channel         calling argument

        IMPLICIT NONE

        CHARACTER*1     esc
        CHARACTER*1     code

        CHARACTER*32    status

        INTEGER*2       lun
        INTEGER*2       io_chan

        INTEGER*2       stat
        INTEGER*2       channel

C*************************************************************************
C       variable initializations                                        *
C*************************************************************************

        esc  = CHAR(27)
        code = 'C'


C  ...initiate zero peak cahnnel set on 8232

        WRITE(lun,1000)esc,channel,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                             *
C*************************************************************************

1000    FORMAT(1X,A1,I5.5,A1,$)

C*************************************************************************

        RETURN
        END


C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

```
      SUBROUTINE ci8232_ZERO_WIDTH(lun,io_chan,width,stat,status)

C*****************************************************************************
C
C       Title     :     ci8232_zero_width
C       Revision  :     1.00
C
C*****************************************************************************
C
C       Abstract  :     this routine sets the zero window width on the
C                       Canberra 8232 stabalizer
C
C*****************************************************************************
C
C       Arguments :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                       io_chan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                       width    I       I*2     zero window width
C                       stat     O       I*2     status code(0/1 for OK/error)
C                       status   O       C*32    status string
C
C*****************************************************************************
C
C       Routines
C       Called    :     ci8232_status
C
C       Libraries
C       Used      :     none
C
C*****************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for zero window width

C       C*32    status          calling argument
C
C       I*2     lun             calling argument
C       I*2     io_chan         calling argument
C       I*2     stat            calling argument
C
C       I*2     width           calling argument

      IMPLICIT NONE

      CHARACTER*1       esc
      CHARACTER*1       cooe

      CHARACTER*32      status

      INTEGER*2         lun
      INTEGER*2         io_chan

      INTEGER*2         stat
      INTEGER*2         width

C*****************************************************************************
```

```
C       variable initializations                                              *
C*************************************************************************

        esc  = CHAR(27)
        code = 'D'


C  ...initiate zero window width set on 8232

        WRITE(lun,1000)esc,width,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                                   *
C*************************************************************************

1000    FORMAT(1X,A1,I3.3,A1,$)

C*************************************************************************

        RETURN
        END



C/////////////////////////////////////////////////////////////////////////
C \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

        SUBROUTINE ci8232_ZERO_RANGE(lun,io_chan,range,stat,status)

C*************************************************************************
C
C       Title     :     ci8232_zero_range
C       Revision  :     1.00
C
C*************************************************************************
C
C       Abstract  :     this routine sets the zero analog range on the
C                       Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                       io_chan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                       range    I       I*2     zero analog range
C                       stat     O       I*2     status code(0/1 for OK/error)
C                       status   O       C*32    status string
C
C*************************************************************************
C
        Routines
C       Called    :     ci8232_status
C
C       Libraries
```

```
C       Used       :      none
C
C*******************************************************************************
C*******************************************************************************


C       C*1      esc              ASCII code for escape
C       C*1      code             1 character 8232 code for zero analog range

C       C*32     status           calling argument
C
C       I*2      lun              calling argument
C       I*2      io_chan          calling argument
C       I*2      stat             calling argument
C
C       I*2      range            calling argument

        IMPLICIT NONE

        CHARACTER*1      esc
        CHARACTER*1      code

        CHARACTER*32     status

        INTEGER*2        lun
        INTEGER*2        io_chan

        INTEGER*2        stat
        INTEGER*2        range

C*******************************************************************************
        variable initializations                                              *
C*******************************************************************************

        esc  = CHAR(27)
        code = 'E'


C  ...initiate zero analog range set on 8232

        WRITE(lun,1000)esc,range,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*******************************************************************************
C       format declarations                                                   *
C*******************************************************************************

 1000   FORMAT(1X,A1,I1,A1,$)

C*******************************************************************************

        RETURN
        END
```

```
C////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_GAIN_HOLD(lun,io_chan,option,stat,status)

C*************************************************************************
C
C       Title     :     ci8232_gain_hold
C       Revision  :     1.00
C
C*************************************************************************
C
C       Abstract  :     this routine sets the gain stabilization on the
C                       Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                       io_chan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                       option   I       I*2     0/1/2 for off/hold/on
C                       stat     O       I*2     status code(0/1 for OK/error)
C                       status   O       C*32    status string
C
C*************************************************************************
C
C       Routines
C       Called    :     ci8232_status
C
      Libraries
C       Used      :     none
C
C*************************************************************************
C*************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for gain stabilization

C       C*32    status          calling argument
C
C       I*2     lun             calling argument
C       I*2     io_chan         calling argument
C       I*2     stat            calling argument
C
C       I*2     option          calling argument

      IMPLICIT NONE

      CHARACTER*1       esc
      CHARACTER*1       code

      CHARACTER*32      status

      INTEGER*2         lun
      INTEGER*2         io_chan

      INTEGER*2         stat
```

```
      INTEGER*2       option

C*************************************************************************
C       variable initializations                                         *
C*************************************************************************

      esc  = CHAR(27)
      code = 'P'


C  ...initiate gain stabilization set on 8232

      WRITE(lun,1000)esc,option,code

C  ...read status report from 8232

      CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                              *
C*************************************************************************

1000  FORMAT(1X,A1,I1,A1,$)

C*************************************************************************

      RETURN
      END




C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_ZERO_HOLD(lun,io_chan,option,stat,status)

C*************************************************************************
C
C       Title     :     ci8232_zero_hold
C       Revision  :     1.00
C
C*************************************************************************
C
C       Abstract  :     this routine sets the zero stabilization on the
C                       Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                       io_chan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                       option   I       I*2     0/1/2 for off/hold/on
C                       stat     O       I*2     status code(0/1 for OK/error)
C                       status   O       C*32    status string
C
C*************************************************************************
C
```

```
C       Routines
C       Called    :     ci8232_status
C
C       Libraries
C       Used      :     none

C*************************************************************************
C*************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for zero stabilization

C       C*32    status          calling argument
C
C       I*2     io_chan         calling arguemnt
C       I*2     lun             calling arguemnt
C       I*2     stat            calling argument
C
C       I*2     option          calling argument

        IMPLICIT NONE

        CHARACTER*1     esc
        CHARACTER*1     code

        CHARACTER*32    status

        INTEGER*2       lun
        INTEGER*2       io_chan

        INTEGER*2       stat
        INTEGER*2       option

C*************************************************************************
C       variable initializations                                        *
C*************************************************************************

        esc  = CHAR(27)
        code = 'G'


C  ...initiate zero stabilization set on 8232

        WRITE(lun,1000)esc,option,code

C  ...read status report from 8232

        CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                             *
C*************************************************************************

1000    FORMAT(1X,A1,I1,A1,$)

C*************************************************************************

        RETURN
```

```
      END


C'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_STORE(lun,io_chan,stat,status)

C*********************************************************************
C
C     Title      :     ci8232_store
C     Revision   :     1.00
C
C*********************************************************************
C
C     Abstract   :     this routine stores the current settings to NVRAM
C                      on the Canberra 8232 stabalizer
C
C*********************************************************************
C
C     Arguments  :     lun      I       I*2     logical unit for serial
C                                                communication with 8232
C                      io_chan  I       I*2     I/O channel for serial
C                                                communication with 8232
C                      stat     O       I*2     status code(0/1 for OK/error)
C                      status   O       C*32    status string
C
C*********************************************************************
C
      Routines
C     Called     :     ci8232_status
C
C     Libraries
C     Used       :     none
C
C*********************************************************************
C*********************************************************************

C
C     C*1      esc              ASCII code for escape
C     C*1      code             1 character 8232 code for store to NVRAM

C     C*32     status           calling argument
C
C     I*2      lun              calling argument
C     I*2      io_chan          calling argument
C     I*2      stat             calling argument
C

      IMPLICIT NONE

      CHARACTER*1       esc
      CHARACTER*1       code

      CHARACTER*32      status

      INTEGER*2         lun
      INTEGER*2         io_chan
```

```
      INTEGER*2       stat

C*************************************************************************
C       variable initializations                                        *
C*************************************************************************

      esc  = CHAR(27)
      code = 'M'


C  ...initiate store to NVRAM on 8232

      WRITE(lun,1000)esc,code

C  ...read status report from 8232

      CALL ci8232_status(lun,io_chan,stat,status)


C*************************************************************************
C       format declarations                                             *
C*************************************************************************

1000  FORMAT(1X,A1,A1,$)

C*************************************************************************

      RETURN
      END









C/////////////////////////////////////////////////////////////////////////
C\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

      SUBROUTINE ci8232_REPORT(lun,io_chan,report,stat,status)

C*************************************************************************
C
C       Title     :     ci8232_report
C       Revision  :     1.00
C
C*************************************************************************
C
C       Abstract  :     this routine retireves setup and status information
C                       on the Canberra 8232 stabalizer
C
C*************************************************************************
C
C       Arguments :     lun      I        I*2      logical unit for serial
C                                                  communication with 8232
C                       io_chan  I        I*2      I/O channel for serial
C                                                  communication with 8232
C                       report   O        struct   status and setup report
```

```
C                       stat    O       I*2     status code(0/1 for OK/error)
C                       status  O       C*32    status string
C
C*************************************************************************
C
C       Routines
C       Called      :       timed_read, ci8232_status
C
C       Libraries
C       Used        :       none
C
C*************************************************************************
C*************************************************************************

C
C       C*1     esc             ASCII code for escape
C       C*1     code            1 character 8232 code for store to NVRAM

C       C*32    status          calling argument
C
C       I*2     lun             calling arguemnt
C       I*2     io_chan         calling arguemnt
C       I*2     stat            calling argument
C
C       struct  record          calling argument
C
C       I*2     length          length of port read
C       I*2     clength         anticipated length of port read
C       C*120   string          character buffer for port read
C       I*2     ier             return status of timed read

        IMPLICIT NONE

        CHARACTER*1     esc
        CHARACTER*1     code

        CHARACTER*32    status

        INTEGER*2       lun
        INTEGER*2       io_chan
        INTEGER*2       stat

        STRUCTURE /report_struct/
                INTEGER*2 gain_peak
                INTEGER*2 gain_window
                INTEGER*2 gain_range
                INTEGER*2 gain_stab
                INTEGER*2 gain_rate
                INTEGER*2 gain_corr
                INTEGER*2 zero_peak
                INTEGER*2 zero_window
                INTEGER*2 zero_range
                INTEGER*2 zero_stab
                INTEGER*2 zero_rate
                INTEGER*2 zero_corr
        END STRUCTURE

        RECORD /report_struct/ report

        INTEGER*2       length
```

```
        INTEGER*2       clength
        INTEGER*2       ier

        CHARACTER*120   string

C*************************************************************************
C       variable initializations                                        *
C*************************************************************************

        esc  = CHAR(27)

C  ...initiate gain setup report on 8232

        code = 'H'
        WRITE(lun,1000)esc,code

        clength = 15
        length = clength
        CALL timed_read(io_chan,string,length,ier)
        type *,string(2:clength)
        IF((length.LT.clength-2).OR.ier.NE.0)THEN
          type *,length,clength,ier
          stat = 12
          status = 'error in QIO read'
          GOTO 500
        ENDIF
        READ(string,1001,end=400,err=450)       report.gain_peak,
     1                                         report.gain_window,
     2                                         report.gain_range, code

C  ...read status report from 8232
        CALL ci8232_status(lun,io_chan,stat,status)

C  ...check for error from status report
        IF(stat.NE.0)THEN
          GOTO 500
        ENDIF

C  ...check for error in response from 8232
        IF(code.NE.'H')THEN
          stat   = 7
          status = 'Error in 8232 gain setup report'
          GOTO 500
        ENDIF


C  ...initiate zero setup report on 8232

        code = 'J'
        WRITE(lun,1000)esc,code

        clength = 15
        length = clength
        CALL timed_read(io_chan,string,length,ier)
        type *,string(2:clength)
        IF((length.LT.clength-2).OR.ier.NE.0)THEN
          type *,length,clength,ier
          stat = 12
          status = 'error in QIO read'
          GOTO 500
```

```
      ENDIF
      READ(string,1001,end=400,err=450)        report.zero_peak,
     1                                         report.zero_window,
     2                                         report.zero_range, code

...read status report from 8232
      CALL ci8232_status(lun,io_chan,stat,status)

C  ...check for error from status report
      IF(stat.NE.0)THEN
        GOTO 500
      ENDIF

C  ...check for error in response from 8232
      IF(code.NE.'J')THEN
        stat   = 7
        status = 'Error in 8232 zero setup report'
        GOTO 500
      ENDIF

C  ...initiate gain status report on 8232

      code = 'I'
      WRITE(lun,1000)esc,code

      clength = 15
      length = clength
      CALL timed_read(io_chan,string,length,ier)
      type *,string(2:clength)
      IF((length.LT.clength-2).OR.ier.NE.0)THEN
        type *,length,clength,ier
        stat = 12
        status = 'error in QIO read'
        GOTO 500
      ENDIF
      READ(string,1002,end=400,err=450)        report.gain_stab,
     1                                         report.gain_rate,
     2                                         report.gain_corr, code

C  ...read status report from 8232
      CALL ci8232_status(lun,io_chan,stat,status)

C  ...check for error from status report
      IF(stat.NE.0)THEN
        GOTO 500
      ENDIF

C  ...check for error in response from 8232

      IF(code.NE.'I')THEN
        stat   = 7
        status = 'Error in 8232 gain status report'
        GOTO 500
      ENDIF



L  ...initiate zero status report on 8232

      code = 'K'
```

```
        WRITE(lun,1000)esc,code

        clength = 15
        length = clength
        CALL timed_read(io_chan,string,length,ier)
        type *,string(1:clength)
        IF((length.LT.clength-2).OR.ier.NE.0)THEN
          type *,length,clength,ier
          stat = 12
          status = 'error in QIO read'
          GOTO 500
        ENDIF
        READ(string,1002,end=400,err=450)       report.zero_stab,
     1                                         report.zero_rate,
     2                                         report.zero_corr, code

C   ...read status report from 8232
        CALL ci8232_status(lun,io_chan,stat,status)

C   ...check for error from status report
        IF(stat.NE.0)THEN
          GOTO 500
        ENDIF

C   ...check for error in response from 8232
        IF(code.NE.'K')THEN
          stat   = 7
          status = 'Error in 8232 zero status report'
          GOTO 500
        ENDIF

        GOTO 500
400     CONTINUE
          stat   = 10
          status = '8232 not reachable, check port'
          GOTO 500
450     CONTINUE
          stat   = 11
          status = 'Error in 8232 communication'
C   ...label for error return
500     CONTINUE

C*************************************************************************
C       format declarations                                              *
C*************************************************************************

1000    FORMAT(1X,A1,A1,$)
1001    FORMAT(1X,I5,1X,I3,1X,I1,A1)
1002    FORMAT(1X,I1,1X,I5,1X,I3,A1)

C*************************************************************************

        RETURN
        END
```

```
      SUBROUTINE AVE (NS,NE,SUM,AV,Y)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                               AVE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     Version: 1
C     Author: Ray Gunnink
C     Date:   circa 1974
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C     PURPOSE
C     To determine the average value of a set of data counts taken fro
C     channels NS to NE (inclusive) in array Y.
C
C     ARGUMENTS
C
C     INPUT
C     NS      Starting channel of integration.
C     NE      Ending channel of integration.
C     Y       Data array
C
C     RETURN
C     SUM     Integrated count
C     AVE     Average count per channel
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DIMENSION Y(2)

      SUM = 0.0
      SM = NE - NS + 1
      DO 2 I = NS,NE
    2 SUM = SUM + Y(I)
      AV = SUM / SM
      RETURN
      END
```

```
      SUBROUTINE BKGD (NS,NE,AV1,AV2,S2,ND,Y1,Y2)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                                BKGD
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Version: 2
C       Author: Ray Gunnink
C       Date:   12-Oct-84
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       PURPOSE
C       To calculate the background continuum under a peak region,
C       to subtract it from the channell counts, and return the net
C       channel and total counts.
C
C       Like NCTS, this subroutine calculates the background distributio
C       for a specified peak region, performs a channel-by-channel
C       subtraction, and returns the net area and net count array.
C       However, the input arguments provide the before and aft average
C       background levels to be used.  The routine uses one pass for a
C       positive loping background, and two for a negative slope. The fi
C       pass determines a straight line connecting the before and aft
C       background levels. The second pass additionally computes a
C       smoothed-step background which is subtracted from the gross chan
C       counts to obtain the net count array and the integrated net coun
C
C       This rountine is similar to the NCTS and BKGRD routines.  The
C       BKGRD is more rigorous and also takes into account the entrance
C       exit slopes of the background.
C
C       *       *       *       *       *       *       *       *
C
C       ARGUMENTS.
C
C       INPUT
C       NS = Beginning channel of peak grouping
C       NE = Ending channel of peak grouping
C       AV1= Average background level in front of the peak grouping.
C       AV2= Average background level aft of the peak grouping.
C       Y1 = Array containing gross counts of the peak grouping.
C
C       RETURN
C       ND = Number of channels in peak grouping
C       S2 = Net counts in the peak region.
C       Y2 = Array of net counts generated by the routine.
C
C       OTHER VARIABLES
C       DBG = Change in background level.
C       CHANS = Number of channels in the peak grouping.
C       SLOPE = Average change in background per channel.
C       AVSIG = A factor related to the statistcal variance of the data
C               in the region.  It is used to provide smoothing to
C               statistically poor data.
```

```
0058    C       SUMY =  Variable used to accumulate the net count using
0059    C               the straight line for background determination.
0060    C       SUM = The final value of SUMY following the 1st pass.
0061    C       YN = Net count of a channel (temporary argument).
0062    C       YNT= Positive value only of YN (temporary argument).
0063    C       SLBG = Straight line background value at a given chennel.
0064    C
0065    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0066    C
0067
0068            REAL*4 S2,AV1,AV2,Y1(2),Y2(2)
0069            INTEGER*2 NS,NE,ND
0070    C
0071
0072            DBG = AV2 - AV1             !Change in background level.
0073            ND = 0
0074            S2 = 0.0
0075            CHANS = NE - NS + 1         !Total channels in group.
0076            SLOPE = DBG / CHANS         !Average change in background per cha
0077            AVSIG = 5./CHANS*SQRT(AV2*CHANS)   !Smoothing factor
0078            NN = 2
0079            IF (SLOPE .GT. 0.0)  NN = 1  !One pass for positive sloping bkgr
0080
0081            DO 30 J = 1,NN
0082            SUMY = 0.0
0083            SLBG = AV1                  !Initial straight line bkgrd value set a
0084
0085            DO 20 I = NS,NE
0086            SLBG = SLBG + SLOPE         !New straight line bkgrd value.
0087            YN = Y1(I) - SLBG           !Calc. net count.
0088            YNT = YN
0089            IF (YN .LT. 0.0)  YNT = 0.0     !Only integrate positive values.
0090            SUMY = SUMY + YNT + AVSIG       !Running sum of net counts.
0091            IF (SLOPE .GT. 0.0)  GOTO 10    !Straight line only on + slope
0092            IF (J .EQ. 1)  GOTO 20          !Straight line only on 1st pass
0093            YN = Y1(I) -AV1 -DBG*SUMY/SUM   !Calc. smoothed-step bkgrd value
0094
0095    10      ND = ND + 1
0096            Y2(ND) = YN                 !Store net channel count.
0097            S2 = S2 + YN                !Integrate net channel counts.
0098    20      CONTINUE
0099
0100            SUM = SUMY      !Store total net counts using straight line bkgr
0101            IF (SUMY .EQ. 0.)RETURN
0102    30      CONTINUE
0103
0104            RETURN
0105            END
```

```

      SUBROUTINE BKGRD (NS,NE,BGLO,BGHI,SLPLO,SLPHI,NPTS,SMY,Y,YNET)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                               BKGRD
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Version: 1b
C       Author: Ray Gunnink
C       Date:   12-Dec-83
C       Last revision: 20-Sep-84 (R.G.)
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C       PURPOSE
C
C       This subroutine calculates the background distribution for a
C       given peak region, subtracts it, and returns the net count array
C       Like NCTS and BKGD, this routine generates an accurate step-shap
C       background.  However, in these routines, BKGRD takes into
C       consideration the sloping backgrounds surrounding the region of
C       interest.  If the entrance and exit slopes differ, the routine
C       assumes that the change from one slope to the other is monotonic
C
C       *       *       *       *       *       *       *       *
C
C       ARGUMENTS
C
C       INPUT
C       NS = starting channel of the region
C       NE = ending channel of the region.
C       BGLO = background count in channel next to low E side of peak re
C       BGHI = background level next to high energy side of region.
C       SLPLO = background slope on low side (in counts/channel)
C       SLPHI = background slope on high side (in counts/channel)
C       Y(I) = array containing original spectral data.
C
C       RETURN
C       NPTS = number of channels in YNET array.
C       SMY = sum of net counts returned to calling program
C       YNET(I) = array into which net counts are placed (may be same as
C
C       OTHER VARIABLES
C       DBG = Change in background level.
C       SIG & AVSIG = Factors related to the statistcal variance of the
C                     in the region.  They are used to provide a measure
C                     smoothing to statistically poor data.
C       SLPBG = Contribution to the bkgrd due to the slope of the bkgrd.
C       SLBG = Straight line background value at a given channel.
C       YN = Net count of a channel.
C       YNT = Positive value only of YN.
C       SUMY1 = Integrated net area as obtained using a straight line bk
C       SUMY2 = Integrated net area as obtained using a step-function bk
C       YSUM1 = Variable used to accumulate stright-line net area.
C       YSUM2 = Variable used to accumulate step-function net area.
C       YS1,YS2,YS3,YS4 = Accumulator variables used to account for hist
```

```
0058    C                          nature of the data. ie use halves of 2 adjacen
0059    C                          channels.
  60    C
0061    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0062
0063            REAL*4 BGLO,BGHI,SLPLO,SLPHI,SMY,Y(2),YNET(2)
0064            INTEGER*2 NS,NE,NPTS
0065
0066            NPTS = NE-NS+1              !Number of channels in grouping.
0067            XPTS = NPTS
0068            DBG = (BGHI-BGLO) -0.5 *(SLPLO+SLPHI)*(XPTS+1.) !Level change in
0069            SLOPE = DBG / (XPTS + 1.)  !Average bkgrd slope in counts/channe
0070            SLBG = BGLO                !Starting (low-side) bkgrd level.
0071            SUMY1 = 0.0
0072            YSUM2 = 0.0
0073
0074            SIG = 5.*SQRT(BGLO)     !Variable related to statistical level of
0075            AVSIG = SIG/XPTS
0076
0077            L = 0
0078            SLPBG = 0.0
0079
0080            DO 10 I = NS,NE          !Get 1st estimate of net area of peak(s)
0081            FRAC = L / XPTS          !Fractional distance through grouping.
0082            SLPBG = SLPBG+SLPLO*(1.-FRAC)+SLPHI*FRAC  !Amt due to sloping bk
0083            SLBG = SLBG + SLOPE      !Straight line background level.
0084            L = L + 1
0085            YN = Y(I) -SLBG -SLPBG   !Net count in channel using straight lin
  86            IF(YN .LT. 0.) YN = 0.   !Only accumulate positive values.
0087    10      SUMY1 = SUMY1 + YN       !Approximate area using straight line.
0088            SUMY1 = SUMY1 + SIG      !Add value related to statistical varian
0089            NN = 2
0090            IF (SLOPE .GE. 0.0) NN = 1  !Only 1 pass if slope is positive.
0091            DO 30 J = 1,NN   !1st uses straight line.  2nd also uses step.
0092            L = 0
0093            SLBG = BGLO      !Start at initial bkgrd level.
0094            YSUM1 = 0.0
0095            SLPBG = 0.0
0096            YS1 = 0.5 * BGLO  !Data is histogram.  Use half of 2 adjacent ch
0097            YS3 = YS1
0098            SMY = 0.0
0099
0100              DO 20 I = NS,NE
0101              SLBG = SLBG + SLOPE            !Straight line background.
0102              FRAC = L / XPTS                !Fractional distance through gro
0103              SLPBG = SLPBG +SLPLO *(1.-FRAC)+SLPHI *FRAC !Background slope.
0104              YN = Y(I) - SLBG - SLPBG       !Net count using straight line.
0105              YNT = YN
0106              IF (YN .LT. 0.) YNT = 0.       !Only integrate positive values.
0107              L = L + 1
0108
0109              IF (DBG .GE. -10.)THEN
0110               IF ((J.EQ.1) .AND.(NN.EQ.2) )  GOTO 15       !Use straight li
  11               GOTO 14
0112              ENDIF
0113              YSUM1 = YSUM1 + YNT + AVSIG   !Partial area up to channel "I".
0114              YS2 = 0.5 * (BGLO + DBG * YSUM1/SUMY1)  !Other half.
```

```
0115                YN = Y(I) -YS1 -YS2 -SLPBG     !Net count using straight line b
0116                IF (J .EQ. 1)  GOTO 15         !Skip following on first iterati
   17
0118                YSUM2 = YSUM2+YN+AVSIG  !Partial area to channel "I" using ste
0119                YS4 = 0.5 * (BGLO + DBG * YSUM2/SUMY2)  !Other half.
0120                YN = Y(I) -YS3 -YS4 -SLPBG     !Net count resulting from refine
0121                YS3 = YS4
0122     14         YNET(L) = YN           !Transfer to YNET array on last iteratio
0123     15         SMY = SMY + YN         !Net counts (or area).
0124                YS1 = YS2
0125     20         CONTINUE
0126
0127              SUMY2 = SMY + SIG        !New value of SUMY2 for 2nd pass.
0128     30       CONTINUE
0129
0130              RETURN
0131              END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	460	PIC CON REL LCL SHR EXE R
2	$LOCAL	84	PIC CON REL LCL NOSHR NOEXE R
	Total Space Allocated	544	

ENTRY POINTS

Address	Type	Name
0-00000000		BKGRD

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
**	R*4	AVSIG	AP-00000010@	R*4	BGHI	AP-0000000C@	
**	R*4	FRAC	**	I*4	I	2-0000001C	
AP-00000008@	I*2	NE	2-00000018	I*4	NN	AP-0000001C@	
2-00000014	R*4	SIG	**	R*4	SLBG	2-00000008	
AP-00000018@	R*4	SLPHI	AP-00000014@	R*4	SLPLO	AP-00000020@	
2-00000028	R*4	SUMY2	2-00000000	R*4	XPTS	**	
**	R*4	YS1	2-00000024	R*4	YS2	2-00000020	
**	R*4	YSUM1	2-00000010	R*4	YSUM2		

```
0001     C        FUNCTION ER
0002
0003              FUNCTION CALERR(XY,ALPHA,NTERMS,CHISQR)
0004              DOUBLE PRECISION ALPHA(3,3)
0005
0006
0007              YR=0.
0008              DO 36 I=1,NTERMS
0009              F1 = 1.
0010              IF(I.EQ.1)GOTO 30
0011              F1=XY**(I-1)
0012     30       DO 36 J=1,NTERMS
0013              F2 = 1.
0014              IF(J.EQ.1)GOTO 34
0015              F2=XY**(J-1)
0016     34       YR=YR+ALPHA(I,J)*F1*F2
0017     36       CONTINUE
0018              CALERR=SQRT(CHISQR*ABS(YR))
0019              RETURN
0020              END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	166	PIC CON REL LCL SHR EXE R
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE R
	Total Space Allocated	190	

ENTRY POINTS

Address	Type	Name
0-00000000	R*4	CALERR

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
AP-00000010@	R*4	CHISQR	**	R*4	F1	**	
**	I*4	J	AP-0000000C@	I*4	NTERMS	AP-00000004@	

ARRAYS

Address	Type	Name	Bytes	Dimensions
0-00000008@	R*8	ALPHA	72	(3, 3)

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Generalized GFIT
C       25-MAR-80; REV 23-JUL-82 by W.D.Ruhter
C       Edited 28-Feb-83 by J.B.Niday (removed index locn to caller)
C       and 24-May-83(prevent crash on neg. Y values)
C       Edited 25-Jun-87 by R. Gunnink (Added out-of-bound check)
C
C       Fit Gaussian to upper half of a peak after stripping out the
C   tailing contribution. Return peak posn, height, and width. Return
C   negative error flag if no valid peak in region.
C       L       Approx index of peak posn in data array Y
C       M       No. of points beyond L to include
C       ALFA    Peak width parameter calculated (an approx received)
C       EXP1    Short term tail amplitude
C       EXP2    Short term tail slope
C       PKHT    Peak height to be returned.
C       ER      Uncertainty in peak position; unless its value is -.
C               Then ER is a flag (-1= Parabola nearly degenerate.
C                                  -2= Peak position out of bounds.)
C       Y       Data array received
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
        SUBROUTINE GFIT(L,M,ALFA,EXP1,EXP2,PKPOS,PKHT,ER,Y)
        DIMENSION Y(2)

C       Initializations
        PKPOS = L         !In case of early return
        ER = 0
        SY = 0.0
        SYY=0.
        SXY=0.
        SXXY=0.
        C=M*(M+1)/3.
        Y1 = Y(L+1)
        Y2 = Y(L-1)
        TPOS = .5 * (Y2-Y1) / (Y1 - 2. * Y(L) + Y2)
        YO = Y(L)

C       Strip out the short term tail contribution to peak
        I=-M
30      J=L+I
        XL=I
        DLX = XL - TPOS
        Y1 = Y(J)
        IF (DLX .GE. 0) GO TO 34
        Y1 =Y1-YO * EXP1* EXP(EXP2 * DLX) * (1. -EXP(.4*ALFA*DLX*DLX))
34      IF(Y1 .GT. 0)GO TO 37

        TYPE 1034,(Y(K),K=L-M,L+M)
1034    FORMAT(' A negative data value was incountered in GFIT.'/
     +  ' Its value will be set to 0.1. Data Values ='/1P(5E12.4))

        Y1=Y(J)
        IF(Y1 .LE. 0.)Y1 = 0.1
```

```
0058     C        Take natural log of net 'Y' data. Result should be a parabola.
0059     C  Fit to a parabola to get peak height, position, and width.
   60
0061     37       Y1=ALOG(Y1)
0062              SY=SY+Y1
0063              SYY=SYY+Y1**2
0064              SXY=SXY+XL*Y1
0065              SXXY=SXXY+(XL**2-C)*Y1
0066              I=I+1
0067              IF(I.LE.M)GOTO 30
0068
0069     C        COMPUTE THE DIAGONAL ELEMENTS OF X'X
0070              XPX11=2*M+1.
0071              XPX22=XPX11*C
0072              XPX33=XPX22*(4*M*(M+1)-3)/15.
0073     C        COMPUTE LEAST SQUARES ESTIMATES OF COEFFICIENTS
0074              A0=SY/XPX11
0075              A1=SXY/XPX22
0076              ALFA=SXXY/XPX33
0077     C        IS THE PARABOLA NEARLY DEGENERATE
0078              ER = -1.                    !Default value
0079              IF(ABS(A0) .LT. 1.E-06) GOTO 50
0080
0081     C        COMPUTE THE COORDINATES OF THE VERTEX
0082              PKPOS=-0.5*A1/ALFA
0083              PKHT=A0+0.5*A1*PKPOS-ALFA*C
0084              PKPOS = PKPOS + L
0085              IF ((PKPOS .LT. (L-M)) .OR. (PKPOS .GT. (L+M))) GOTO 40 !Check b
   86
0087              ER=SYY-A0*SY-A1*SXY-ALFA*SXXY
0088              IF(M.GT.1)ER=ER/(2*(M-1))
0089              PKHT=EXP(PKHT)
0090              GOTO 60
0091
0091     40       ER = -2.                                    !Set flag
0092     50       PKPOS = L                                   !Return starting positio
0093              PKHT = Y(L)                                 !Return default height
0094              TYPE 9000, ER
0095              PRINT 9000, ER
0096     9000     FORMAT (' ***Error in GFIT***  Error flag = ',F3.0)
0097
0098     60       RETURN
0099              END
```

```
0001     C        GPM
0002     C
0003     C        Last edit Jun-12-1986 (J.B.N.)
0004     C        Test effect of double precision.
0005
0006              SUBROUTINE GPM (EPK,ELOG,GEOM,SURFC,DEPTH,SMU,VALUE,GAMAS,DCNST)
0007
0008              IMPLICIT DOUBLE PRECISION(A-H,O-Z)
0009              REAL*4 DCNST(2),EPK,ELOG,GEOM,SURFC,DEPTH,SMU,VALUE,GAMAS
0010
0011              EXOVR = DCNST(12)
0012              DETHT = DCNST(13)
0013              DEP = 0.5 * DEPTH
0014              RSQ = DCNST(14) * DCNST(14)
0015              XMU = -2.316 + 4.2 * EXP(-.478 * ELOG - 1.434)
0016              XMU = EXP (XMU)
0017              XM = 1.0 / XMU
0018              SEE = EXP (-DETHT * XMU)
0019              XM = XM * (1. - XMU * DETHT * SEE / (1. - SEE))
0020              GM = GEOM + XM
0021              GG = GM * GM
0022
0023     C        ROSQ = -GG + 0.5
0024     C        IF(RSQ .GT. 0.000001) THEN
0025                ROSQ = -GG + .5 * RSQ / (1. - GM/SQRT (GG+ RSQ))
0026     C        ENDIF
0027
0028              ERSQ = ROSQ * (1.1 -  XM / SQRT (DETHT))
0029              GGSQ = GG + ERSQ
0030              GLOST = 1.
0031              GCORR = 1.
0032              SQGCR = 1.
0033              SATN = 1.
0034              IF (SURFC) 21,31,21
0035     21       RR = SURFC / 3.1416
0036              GM2 = GM - DEP * (1. - EXP (-SMU))
0037              GG2 = GM2 * GM2
0038              RR = RR * (1.4 - .4 / GM2)
0039              GCORR = RR / (GG2 * LOG ((GG2 + RR)/GG2))
0040              SQGCR = SQRT (GCORR)
0041              SMU = SMU * SQGCR
0042              UL = SMU * SMU
0043              TOLOA = 2. * DEP / GM
0044              ADD = 1.
0045              DFACT = 1.
0046              ASQ = GGSQ * GCORR
0047              TOTAL = (ASQ /(ASQ -DEP*DEP)) + .1 * SMU * (TOLOA **3)
0048              DO 25 I = 1,10
0049              XK = 2 * I
0050              DFACT = DFACT * XK * (XK + 1.)
0051              ADD = UL * ADD
0052              AD = ADD / DFACT
0053              IF (TOTAL - AD * 200.) 24,24,27
0054     24       TOTAL = TOTAL + AD
0055     25       CONTINUE
0056     27       TOTAL = TOTAL + AD
0057              ADD = SMU
```

```
0058            DFACT = 1.
0059            DO 28 I = 1,5
0060            XK = 2 * I - 1
0061            AD = TOLOA * ADD / ((XK + 2.) * DFACT)
0062            IF (TOTAL - AD *200.) 29,29,30
0063    29      TOTAL = TOTAL + AD
0064            ADD = ADD * UL
0065            DFACT = DFACT * (XK + 1.) * (XK + 2.)
0066    28      CONTINUE
0067    30      TOTAL = TOTAL + AD
0068            SATN = EXP (-SMU)
0069            GLOST = TOTAL * SATN
0070
0071    31      GM = GM - DEP * (1. - SATN)
0072            SGSQ = SQRT (GM * GM + ROSQ)
0073            SMU = DCNST(16) * (SGSQ - GM) / (SGSQ + GM)
0074            N = 1
0075            M = 6
0076            IF (EPK - EXOVR) 32,32,34
0077    32      N = 7
0078            M = 11
0079    34      EFLOG = DCNST(N)
0080            L = N+1
0081            DO 36 I = L,M
0082    36      EFLOG = EFLOG + DCNST(I) * ELOG ** (I-N)
0083            EFF = EXP (EFLOG)
0084            GAMAS = VALUE * GGSQ * GCORR / (EFF * GLOST)
0085            RD = .7 * DCNST(14)
0086            VALUE = (1. + RD / GG) * SQGCR
0087            RETURN
0088            END
```

```
0001            SUBROUTINE MATINV(ARRAY,MX,NORDER,DET)
0002
0003    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004    C       MATINV modified to carry array dimension
0005    C       Last edit 16-Jun-82
0006    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0007
0008            DIMENSION IK(50),JK(50)
0009            DOUBLE PRECISION ARRAY(MX,MX),AMAX,SAVE
0010
0011    10      DET = 1.
0012    11      DO 100 K =1,NORDER
0013    C
0014    C       FIND LARGEST ELEMENT ARRAY(I,J) IN REST OF MATRIX
0015    C
0016            AMAX = 0.
0017    21      DO 30 I =K,NORDER
0018            DO 30 J =K,NORDER
0019    23      IF(ABS(AMAX)-ABS(ARRAY(I,J)))24,24,30
0020    24      AMAX = ARRAY(I,J)
0021            IK(K) = I
0022            JK(K) = J
0023    30      CONTINUE
0024    C
0025    C       INTERCHANGE ROWS AND COLUMNS TO PUT AMAX IN ARRAY(K,K)
0026    C
0027    31      IF(AMAX)41,32,41
0028    32      DET = 0.
0029            GOTO 140
0030    41      I = IK(K)
0031            IF(I-K)21,51,43
0032    43      DO 50 J =1,NORDER
0033            SAVE = ARRAY(K,J)
0034            ARRAY(K,J) = ARRAY(I,J)
0035    50      ARRAY(I,J) = -SAVE
0036    51      J = JK(K)
0037            IF(J-K) 21,61,53
0038    53      DO 60 I = 1,NORDER
0039            SAVE = ARRAY(I,K)
0040            ARRAY(I,K) = ARRAY(I,J)
0041    60      ARRAY(I,J) = -SAVE
0042    C
0043    C       ACCUMULATE ELEMENTS OF INVERSE MATRIX
0044    C
0045    61      DO 70 I =1,NORDER
0046            IF(I-K)63,70,63
0047    63      ARRAY(I,K) = -ARRAY(I,K)/AMAX
0048    70      CONTINUE
0049    71      DO 80 I =1,NORDER
0050            DO 80 J =1,NORDER
0051            IF(I-K)74,80,74
0052    74      IF(J-K)75,80,75
0053    75      ARRAY(I,J)= ARRAY(I,J) + ARRAY(I,K) * ARRAY(K,J)
0054    80      CONTINUE
0055    81      DO 90 J =1,NORDER
0056            IF(J-K)83,90,83
0057    83      ARRAY(K,J) = ARRAY(K,J)/AMAX
```

```
0058     90       CONTINUE
0059              ARRAY(K,K) = 1. / AMAX
0060     100      CONTINUE
0061     C
0062     C        RESTORE ORDERING OF MATRIX
0063     C
0064     101      DO 130 L =1,NORDER
0065              K = NORDER - L + 1
0066              J = IK(K)
0067              IF(J-K)111,111,105
0068     105      DO 110 I =1,NORDER
0069              SAVE = ARRAY(I,K)
0070              ARRAY(I,K) = -ARRAY(I,J)
0071     110      ARRAY(I,J) = SAVE
0072     111      I = JK(K)
0073              IF(I-K)130,130,113
0074     113      DO 120 J =1,NORDER
0075              SAVE = ARRAY(K,J)
0076              ARRAY(K,J) = -ARRAY(I,J)
0077     120      ARRAY(I,J) = SAVE
0078     130      CONTINUE
0079     140      RETURN
0080              END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	693	PIC CON REL LCL SHR EXE R
2	$LOCAL	512	PIC CON REL LCL NOSHR NOEXE R
	Total Space Allocated	1205	

ENTRY POINTS

Address	Type	Name
0-00000000		MATINV

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
2-00000190	R*8	AMAX	AP-00000010@	R*4	DET	**	
2-00000198	I*4	K	2-000001BC	I*4	L	AP-00000008@	
**	R*8	SAVE					

```
0001            SUBROUTINE MAXVAL (NS,NE,JMAX,YMAX,Y)
0002
0003    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004    C
0005    C                                       MAXVAL
0006    C
0007    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0008    C
0009    C       Version: 1
0010    C       Author: Ray Gunnink
0011    C       Date:   circa 1974
0012    C
0013    C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
0014    C
0015    C       PURPOSE
0016    C
0017    C       To determine the position and value of the maximum value (count)
0018    C       in a set of data points from NS to NE inclusive.
0019    C
0020    C       *       *       *       *       *       *       *       *
0021    C
0022    C       ARGUMENTS
0023    C
0024    C       INPUT
0025    C       NS      Starting channel for search
0026    C       NE      Ending channel for search
0027    C       Y       Data array
0028    C
0029    C       RETURN
0030    C       JMAX    Position of maximum value
0031    C       YMAX    Maximum value
0032    C
0033    CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0034    C
0035    C
0036
0037            REAL*4 YMAX,Y(2)
0038            INTEGER*2 NS,NE,JMAX
0039
0040            YMAX = 0.0
0041
0042            DO 2 I = NS,NE
0043            IF (Y(I) .LE. YMAX) GOTO 2
0044            YMAX = Y(I)
0045            JMAX = I
0046    2       CONTINUE
0047
0048            RETURN
0049            END
```

```
0001
0002     C      MINVAL
0003     C      THIS ROUTINE RETURNS THE POSITION AND VALUE OF THE MINIMUM VALUE
0004     C      IN A GROUP OF DATA POINTS FROM NS TO NE INCLUSIVE.
0005               SUBROUTINE MINVAL (NS,NE,JMIN,YMIN,Y)
0006               DIMENSION Y(2)
0007               YMIN=1.0E38
0008               DO 4 I=NS,NE
0009               IF (Y(I)-YMIN)2,4,4
0010     2         JMIN=I
0011               YMIN=Y(I)
0012     4         CONTINUE
0013               RETURN
0014               END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	72	PIC CON REL LCL SHR EXE R
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE R
	Total Space Allocated	96	

ENTRY POINTS

Address	Type	Name
0-00000000		MINVAL

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
**	I*4	I	AP-0000000C@	I*4	JMIN	AP-00000008@	
AP-00000010@	R*4	YMIN					

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000014@	R*4	Y	8	(2)

LABELS

Address	Label	Address	Label
**	2	0-00000043	4

```
0001     C      NCNTS     VERSION 0        19-JAN-78  AT  12:30
0002
0003     C      THIS ROUTINE IS SIMILAR TO "NCTS" ROUTINE EXCEPT THAT IT RETURNS
0004     C      AN AVERAGE BACKGROUND AND THE ERROR ASSOCIATED WITH THE TOTAL NET
0005     C      COUNT.
0006
0007               SUBROUTINE NCNTS (N1,N2,M1,M2,ND,S1,S2,ER,AVEBG,Y1,Y2)
0008               DIMENSION Y1(2),Y2(2)
0009
0010               CALL AVE (N1,N2,S1,AV1,Y1)
0011               CALL AVE (M1,M2,S2,AV2,Y1)
0012               S1 = S1 + S2
0013               CALL BKGD (N2+1,M1-1,AV1,AV2,S2,ND,Y1,Y2)
0014               CHANS = ND
0015               XPTS = M2 + N2 - M1 - N1 + 1
0016               AVEBG = 0.5 * (AV1 + AV2)
0017               IF (S2 .LT. 0.0) S2 = 0.0
0018               ER = SQRT (S2 + AVEBG * CHANS * (1. + CHANS / XPTS))
0019               RETURN
0020               END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
1	$CODE	213	PIC CON REL LCL SHR EXE R
2	$LOCAL	132	PIC CON REL LCL NOSHR NOEXE R
	Total Space Allocated	345	

ENTRY POINTS

Address	Type	Name
0-00000000		NCNTS

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
2-00000000	R*4	AV1	2-00000004	R*4	AV2	AP-00000024@	
AP-00000020@	R*4	ER	AP-0000000C@	I*4	M1	AP-00000010@	
AP-00000008@	I*4	N2	AP-00000014@	I*4	ND	AP-00000018@	
**	R*4	XPTS					

13-May-1991 09:3

```
      SUBROUTINE RDBLK(ISIZE,IBUFF,IRC,ICHAN)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       READ ONE BLOCK OF DIRECT ACCESS FILE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        INTEGER*2 IBUFF(1)
        READ(ICHAN'IRC,ERR=95)(IBUFF(I),I=1,ISIZE)
        RETURN
95      TYPE 100,IRC,ISIZE,ICHAN
100     FORMAT (' ERROR READING RECORD:',3I5)
        IF(ISIZE .LE. 256)STOP ' ATTEMPTED .GT. 256'
        STOP ' ATTEMPT TO PASS EOF'
        END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes						
0	$CODE	154	PIC	CON	REL	LCL	SHR	EXE	R
1	$PDATA	68	PIC	CON	REL	LCL	SHR	NOEXE	R
2	$LOCAL	48	PIC	CON	REL	LCL	NOSHR	NOEXE	R
	Total Space Allocated	270							

ENTRY POINTS

Address	Type	Name
0-00000000		RDBLK

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
**	I*4	I	AP-00000010@	I*4	ICHAN	AP-0000000C@	

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000008@	I*2	IBUFF	2	(1)

LABELS

Address	Label	Address	Label
0-00000044	95	1-00000027	100'

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       VERSION 0 19-JAN-78
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SPRAM

C       THIS SUBROUTINE CALCULATES SOME OF THE PARAMETERS USED TO
C       CHARACTERIZE THE SHAPE OF A PEAK

        SUBROUTINE SPRAM (EN,GAIN,SHAPC,SG,ALFA,F12M,EXP1)

        DIMENSION SHAPC(2)

        GSQ = GAIN * GAIN
        SG = (SHAPC(1) + EN * SHAPC(2)) / GSQ + 0.462
        ALFA = -2.7726 / SG
        F12M = SQRT (SG)
        EXP1 = EXP (EN * SHAPC(4) + SHAPC(3))
        RETURN
        END
```

PROGRAM SECTIONS

Name	Bytes	Attributes						
0 $CODE	101	PIC	CON	REL	LCL	SHR	EXE	R
2 $LOCAL	20	PIC	CON	REL	LCL	NOSHR	NOEXE	R
Total Space Allocated	121							

ENTRY POINTS

Address	Type	Name
0-00000000		SPRAM

VARIABLES

Address	Type	Name	Address	Type	Name	Address	T
AP-00000014@	R*4	ALFA	AP-00000004@	R*4	EN	AP-0000001C@	
AP-00000008@	R*4	GAIN	**	R*4	GSQ	AP-00000010@	

```
      SUBROUTINE STRIP1 (NS,NE,PKPOS,PKHT,ALFA,TLAMP,TLSLP,WL,Y)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                                   STRIP1
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       Version: 1
C       Author: Ray Gunnink
C       Date:   circa 1978
C
C       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C       PURPOSE
C       To strip a peak or portion of a peak from a designated set of
C       spectrum channels.  In contrast to routine STRIP, this routine c
C       also calculate X ray line shapes for stripping.
C
C       *       *       *       *       *       *       *       *
C
C       ARGUMENTS
C
C       INPUT
C       NS      Starting position for stripping
C       NE      Ending position for stripping
C       PKPOS   Position of peak vertex
C       PKHT    Height of peak to be stripped
C       ALFA    Width parameter of peak to be stripped
C       TLAMP   Amplitude of tail of peak
C       TLSLP   Slope of tail
C       WL      Full width half maximum of Lorentzian line (channels)
C               WL = 0.0 for gamma rays.
C       Y       Data array
C
C       RETURN
C       Y       Data array with designated peak structure stripped out.
C
C       Routines called: BWF
C
C       *       *       *       *       *       *       *       *
C
C       Reference:  "An Algorithm for Fitting Lorentzian-Broadened, K-Se
C                    X-Ray peaks of the Heavy Elements", Nucl. Inst. & M
C                    143, p145, (1977)
C
C*      *       *       *       *       *       *       *       *
C

      REAL*4 PKPOS,PKHT,ALFA,TLAMP,TLSLP,WL,Y(2)
      INTEGER*2 NS,NE

      IF (WL .EQ. 0.)  GOTO 2

      CX = SQRT (-ALFA)
      GAMA = WL * CX
      C1 = 1. - .5642 * GAMA
```

```
0058            C2 = .15915 * GAMA
0059            C3 = .25 * GAMA * GAMA
0060            C4 = .6366 * GAMA
0061            RT = C1 + C3
0062
0063    2       DO 10 I = NS,NE
0064             XI = I
0065             DX = XI - PKPOS                    !Distance from peak vertex
0066             DDX = DX * DX
0067             ADX = ALFA * DDX
0068             FXX = EXP (ADX)                    !Normalized Gaussian value
0069             FX = FXX
0070             IF (DX .GE. 0.) GOTO 4             !Tailing only on front side
0071             EXPN2 = EXP (TLSLP * DX)           !Normalized tailing amplitude.
0072             FXX = FXX +TLAMP *EXPN2 *(1.- EXP(.4*ADX))  !Add tailing contri
0073             DX = -DX                           !Make + for X ray calculations
0074    4        IF (WL .EQ. 0.) GOTO 9             !Positive value indicates a X ra
0075             CC = CX * DX
0076
0077             CALL BWF (CC)  !To calculate last term in following Voigt funct
0078
0079             FXX = FXX -FX +(C1 -C2 *ADX +C3 *(1.+2.*ADX)) *FX +C4 *CC
0080             FXX = FXX / RT                     !Adjust peak height for X rays.
0081    9        Y(I) = Y(I) - PKHT * FXX           !Strip calc value from data arra
0082    10      CONTINUE
0083
0084            RETURN
0085            END
```

```
!
!  Command file to link CHNGSI program subroutines
!
$ SD [-.OBJECT]
$ LINK/executable=[-.ASSAY]PU238_CHNG-
        pu238_chng,-
        chngrl,-
        dattim,-
        errmsg,-
        getop,-
        juldal,-
        nocomm,-
        pass,-
        readb,-
        readc38,-
        readnm,-
        writb,-
        writ38,-
        yesno
$ set noverify
$sd [-.assay]
```

```
C*****************************************************************
C PU238_CHNG
C Change constants - [Pu-238 isotopic instrument]
C
C
      06-MAR-80 0935 SSJ
C     Rev. 14-Jun-84 lc
C     Rev. 11-Sep-84  TES
C     Rev  27-sep-84 cms
C     Rev. 21-DEC-84 WDR Modified for solid isotopic instrument
C     Rev. 03-JAN-85 MPK Restored DATTIM option argument
C     Rev. 04-Apr-86 WDR Modified for Micro VAX II
C     Rev. 14-Mar-90 WDR Modified for Pu-238 instrument
C     Rev. 24-May-91 WDR Modified for stabilizer control info.
C
C*****************************************************************
C
C
      PROGRAM CHNG
      LOGICAL*1 RANS,REREAD,REWRIT
      LOGICAL * 1 VAXANS
      INTEGER*2 CHAR
C
C
C  The following common block files are generic for all instruments
C
      INCLUDE 'CDATE.CMN'
      INCLUDE 'CLUNS.CMN'
      INCLUDE 'CFILES.CMN'
      INCLUDE 'CONFIL.CMN'
      INCLUDE 'CBACKG.CMN'
      INCLUDE 'CMCONS.CMN'
      INCLUDE 'CONTRL.CMN'
      INCLUDE 'CASSAY.CMN'
      INCLUDE 'CCAL.CMN'
      INCLUDE 'CDIAG.CMN'
      INCLUDE 'CPASS.CMN'
C
C  The following common block files are instrument specific for
C  Pu-238 isotopic assay (gamma ray) instrument
C
      INCLUDE 'P238COM.CMN'
C
      DATA LOUT/5/, DKLUN/10/, DKLUN1/11/
      DATA ERRFIL/'ERRMSG.INS  '/
      DATA BKFIL/'BACKGR.WH8  '/
      DATA NAMSFL/'CONSTA.WH8  '/
C
C  Reset reread and rewrite flags
C  Set flag for beginning constants read
C
      REREAD = .FALSE.
      REWRIT = .FALSE.
      BEGIN = .TRUE.
      CHANG = .TRUE.
      FROMDK= .FALSE.

C  Read background file
C  Get password
C
```

```
        CALL READB
        CALL PASS(OK)
        IF(.NOT. OK)GO TO 2000
C
C  Get operator id
   Allow escape here
C
        ABORT = .FALSE.
        CALL GETOP
        IF(ABORT) GO TO 2000
C
C  Get name of constants file to modify
C  Read constants file
C
 1      CALL READNM
        CALL READC
C
C  Type menu
C
 5      TYPE 900
 900    FORMAT(/' CODE-OPTION')
 6      TYPE 910
 910    FORMAT(' ---- ------')
 7      TYPE 1000
 1000   FORMAT( ' HE - HElp'/
     X          ' MC - Measurement Control'/
     X          ' PT - Preset Time'/
     X          ' PW - PassWords'/
     X          ' DV - Default Values'/
     X          ' RD - Read constants from Disk'/
     X          ' SZ - StabiliZer settings'/
     X          ' LI - LIst all constants'/
     X          ' OU - Change listing device '/
     X          ' EX - EXit from changes'//
     X          ' Note:  Enter -9999. or -9999 to input 0'//)
C
 10     CALL DATTIM(1)
        TYPE 1001,NOWDAT,NOWTIM
 1001   FORMAT(/1X,A9,1X,A8/'$Enter the CODE of the OPTION you want-> ')
        ACCEPT 1002,CHAR
 1002   FORMAT(A2)
C
C  Decide what is to be changed
C
        IF(CHAR .EQ. 'HE') GO TO 5
        IF(CHAR .EQ. 'MC') GO TO 450
        IF(CHAR .EQ. 'PT') GO TO 380
        IF(CHAR .EQ. 'DV') GO TO 500
        IF(CHAR .EQ. 'RD') GO TO 1
        IF(CHAR .EQ. 'SZ') GO TO 700
        IF(CHAR .EQ. 'LI') GO TO 820
        IF(CHAR .EQ. 'PW') GO TO 600
        IF(CHAR .EQ. 'EX') GO TO 800
        IF(CHAR .EQ. 'OU') GO TO 110
        GO TO 10
C
   Change output device for listings
C
 110    TYPE 1112
 1112   FORMAT(/'$Type 6 for listing to printer, 7 for terminal -> ')
```

```
        ACCEPT 1113,LOUT
 1113   FORMAT(I)
        IF(LOUT .NE. 6 .AND. LOUT .NE. 7) GO TO 110
        GO TO 5
C
   Preset time
C
 380    CALL CHNGRL('Asay Tim',0,TIMEPR)
        GO TO 5
C
C  Change measurement control parameters
C
 450    TYPE 1401
 1401   FORMAT(/'$Measurement control - Bias or Precision? (B/P) -> ')
        ACCEPT 1807,RANS
        IF(RANS .EQ. 'P') GO TO 480
        IF(RANS .NE. 'B') GO TO 450
C
C  Measurement control - Bias and Background
C
        CALL CHNGRL('Biastime',0,BITIME)
        WRITE (*,*)' Bias ref is the specific power (mW/g) on 1/1/90'
        CALL CHNGRL('Bias ref',0,BIREF)
        CALL CHNGRL('Hist std',0,HISTSD)
        GO TO 5
C
C  Measurement control - Precision
C
 480    CALL CHNGRL('Prcstime',0,PRTIME)
 481    CALL CHNGIN('No. runs',0,NRUNS)
        IF(NRUNS .EQ. 5 .OR. NRUNS .EQ. 15)GO TO 5
        TYPE 1450
 1450   FORMAT(' *** PRECISION RUNS MUST BE 5 OR 15 ***')
        GO TO 481
C
C  Change default values for Pu-240 abundance
C
500     TYPE 1500
1500    FORMAT(' Enter Pu-240 abundance (Wt %) (0=1.97,-1=Calc.)')
        CALL CHNGRL('Pu-240  ',0,PU240)
        TYPE 1600
1600    FORMAT(' Enter Pu-242 abundance (Wt %) (0=0.1)')
        CALL CHNGRL('Pu-242  ',0,PU242)
        NUFLG = 0                !No Uranium default value
C       TYPE 1505
1505    FORMAT(' Samples contain Uranium? (Y/N) -> ',$)
C       ACCEPT 1507,RANS
1507    FORMAT(A1)
        IF(RANS.EQ.'Y') NUFLG = 7
        NPFLG = 0                !May be used later
        GO TO 5
C
C  Change passwords
C
 600    DO 650 I=1,3
 650      CALL CHNGAL('Password',1,PASSWD(1))
        GO TO 5
C
C  Get stabilizer settings for control of gain and zero
C
```

```
700      CALL CHNGIN('Zero pk ',0,IZPEAK)
         CALL CHNGIN('Zero wnd',0,IZWNDW)
         CALL CHNGRL('Zero rng',0,ZARNG)
         CALL CHNGIN('Gain pk ',0,IGPEAK)
         CALL CHNGIN('Gain wnd',0,IGWNDW)
         CALL CHNGRL('Gain rng',0,GARNG)
         GO TO 5
C
C  If changes are permanent, set rewrite flag
C
 800     TYPE 1806
 1806    FORMAT(/'$Do you want to write these changes to disk now?',
     X           '  (Y/N)  ->  ')
         ACCEPT 1807,RANS
 1807    FORMAT(A1)
         IF(RANS .EQ. 'Y') REWRIT = .TRUE.
         IF(RANS .NE. 'Y' .AND. RANS .NE. 'N') GO TO 800
         GO TO 2000
C
C  List all values
C
 820     WRITE(LOUT,1801)NOWDAT,NOWTIM,CONFIL,COMMNT,FILDAT,FILTIM,
     X       FILOP
 1801    FORMAT('1',A9,2X,A8//1X,A,2X,A//
     X           10X,'Written ',A,2X,A,'  by ',A//)
C
C  Measurement control
C
         WRITE(LOUT,1814)BITIME,PRTIME,BIREF,HISTSD,NRUNS
 1814    FORMAT(//' Measurement Control'//
     X           23X,'Bias',14X,'Precision'//
     X           ' Interval = ',2(8X,G13.6)/
     X           ' Reference = ',7X,G13.6/
     X           ' Standard Dev. = ',3X,G13.6/
     X           ' Cycles = ',36X,I4)
C
C  Default values
C
         WRITE(LOUT,1817)PU240,PU242
1817     FORMAT(//,' Default analysis values'//
     1   ' Pu-240 abundance (wt. %) = ',F7.2,/,
     2   ' Pu-242 abundance (wt. %) = ',F7.2)
C
C  Stabilizer settings
C
         WRITE(LOUT,1818)
 1818    FORMAT(/' Stabilizer settings for zero and gain',/)
         WRITE(LOUT,1820)
 1820    FORMAT(1X,T20,'Zero settings   Gain settings')
         WRITE(LOUT,1821)IZPEAK,IGPEAK
 1821    FORMAT(1X,'Peak positions',4X,I6,11X,I6)
         WRITE(LOUT,1822)IZWNDW,IGWNDW
 1822    FORMAT(1X,'Peak windows ',4X,I6,11X,I6)
         WRITE(LOUT,1823)ZARNG,GARNG
 1823    FORMAT(1X,'Analog ranges ',4X,G13.6,5X,G13.6)
         WRITE(LOUT,1888)
  1888   FORMAT('1')
         IF(LOUT .EQ. 6)CALL CLOSE(LOUT)
         GO TO 5
C
```

```
C  Rewrite constants
C
 2000    IF(REWRIT) CALL WRITB
         IF(REWRIT) CALL WRITC
C
         STOP
         END
```

```
C*******************************************************************************
C
C NOCOMM
C
C       This module is used instead of COMM when no communication
        with the VAX is desired.
C
C           29-Mar-85 MPK
C
C*******************************************************************************
C
        SUBROUTINE COMM
        INCLUDE 'CVAX.CMN'
        VAXUP=.FALSE.
        RETURN
        END
```

```
C*****************************************************************
C
C  GETOP
C       Get operator ID
C
        Output argument
C
C       MTL (LOGICAL*1)             .TRUE. if material type and
C                                   isotopics are needed
C
C       31-May-84 lc
C       Rev 14-Feb-85 MPK           Put in "real" VAX communications
C                                   Supply end and err exits for
C                                   terminal input
C       Rev 08-Oct-85 MPK           Adapted to neutron coinc counters
C                                   GETOP and GETSMP combined
C       Rev 21-Oct-85 MPK           Pad "canned" sample ID properly
C       Rev 23-OCT-85 WDR           Revised for solution assay system
C       Rev 04-Apr-86 WDR           Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE GETOP
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CONTRL.CMN'
C
        LOGICAL*1 YESNO,MTL
C
  Get operator ID; set abort flag if it is null
C
   10   TYPE 1000
 1000   FORMAT (' '/'$Enter operator ID (up to 8 chars)',
        1       ' or RETURN to escape -> ')
        READ (5,1002,END=10,ERR=10) OPERID
 1002   FORMAT(A80)
        ABORT = OPERID.EQ.' '
        IF (ABORT) RETURN
        RETURN
        END
```

```
C*****************************************************************
C CHNGRL
C
C       CHNGRL - Change a REAL value
C       CHNGIN - Change an INTEGER value
C       CHNGAL - Change an alphanumeric value
C
C       REV. 28-Jun-82  1500 SSJ
C
C*****************************************************************
C
        SUBROUTINE CHNGRL(NAME,NUMBER,VARBLE)
C
C  Change an individual real value
C
        REAL*8 NAME
C
        IF(NUMBER .LE. 0) TYPE 1000,NAME,VARBLE
 1000   FORMAT('$',A8, '      = ',G13.6,'  -> ')
        IF(NUMBER .GT. 0) TYPE 1002,NAME,NUMBER,VARBLE
 1002   FORMAT('$',A8,'(',I2,') = ',G13.6,'  -> ')
C
        ACCEPT 1001,X
 1001   FORMAT(F)
        IF(X .NE. 0.) VARBLE = X
        IF(X .EQ. -9999.) VARBLE = 0.
        RETURN
        END
        SUBROUTINE CHNGIN(NAME,NUMBER,IVARBL)
C
C  Change an individual integer value

        REAL*8 NAME
C
        IF(NUMBER .LE. 0) TYPE 1000,NAME,IVARBL
 1000   FORMAT('$',A8, '      = ',I6,7X,'  -> ')
        IF(NUMBER .GT. 0) TYPE 1002,NAME,NUMBER,IVARBL
 1002   FORMAT('$',A8,'(',I2,') = ',I6,7X,'  -> ')
C
        ACCEPT 1001,IX
 1001   FORMAT(I)
        IF(IX .NE. 0) IVARBL = IX
        IF(IX .EQ. -9999) IVARBL = 0
        RETURN
        END
        SUBROUTINE CHNGAL(NAME,NUMBER,ALPHVR)
C
C  Change an individual alphanumeric value
C
        REAL*8 NAME,ALPHVR,X,BLANK
C
        DATA BLANK/8H        /
C
        IF(NUMBER .LE. 0) TYPE 1000,NAME,ALPHVR
 1000   FORMAT('$',A8, '      = ',A8,5X,'  -> ')
        IF(NUMBER .GT. 0) TYPE 1002,NAME,NUMBER,ALPHVR
 1002   FORMAT('$',A8,'(',I2,') = ',A8,5X,'  -> ')
C
        ACCEPT 1001,X
 1001   FORMAT(A8)
```

```
IF(X .NE. 8H         ) ALPHVR = X
IF(X .EQ. 8H#########) ALPHVR = BLANK
RETURN
END
```

```
C*********************************************************************
C  ERRMSG
C
C       Read appropriate error message from the
C       error message file and output to user terminal
        Calling arguments:
C               INUM = Message calling argument
C               ARG = ASCII argument passed to be printed (15A1)
C       File format:
C               INUM (Message call number) = I4
C               IARG (ASCII argument passed) = A1
C               MESSAG (ASCII error message) = 72A1
C
C       21-Apr-84 LC
C       REV. 12-OCT-84 MJC:     CLOSED DKLUN1 IF ARG =' '
C       Rev. 19-Apr-85  MPK     Remove redundant initialization
C                               of DKLUN and DKLUN1
C       Rev. 29-Apr-85  MPK     Put MESSAG and ICHAR in CMSG common
C       Rev. 02-Apr-86  WDR     Modified for Micro VAX II
C       Rev. 13-Jul-87  RDP     Modified for uVax II
C                               Convert BYTE strings to CHARACTER strings
C
C*********************************************************************
C
C
        SUBROUTINE ERRMSG(INUM,ARG)
C
C
        CHARACTER ARG*15,IARG*1
C
C  The following common block files are generic for all instruments

        INCLUDE 'CFILES.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CMSG.CMN'
C
C
C  Open error message file
C  Locate proper error message
C
        OPEN (UNIT=DKLUN1,NAME=ERRFIL,TYPE='OLD',ERR=25)
 10     READ(DKLUN1,1000,ERR=20,END=20)NUM,IARG,ICHAR,
     X           (MESSAG(I),I=1,ICHAR)
 1000   FORMAT(I4,A1,Q,72A1)
        IF (NUM .EQ. -999)GO TO 20
 11     IF (NUM .EQ. INUM) GO TO 15
        GO TO 10
C
C  Print error message
C
 15     TYPE 1015, NUM, (MESSAG(I),I=1,ICHAR)
 1015   FORMAT(/5X,'**** ERROR # ',I3/5X,72A1,' ****')
        IF(IARG .NE. ' ')GO TO 17
        CLOSE(UNIT=DKLUN1)
        GO TO 30
 17     TYPE 1016, ARG
 1016   FORMAT(7X,A15)
        CLOSE(UNIT=DKLUN1)
        GO TO 30
C
```

```
C  Error number not found in file
C
 20     TYPE 1020, INUM, ERRFIL
 1020   FORMAT(/5X,' Error # ',I4,' not located in ',A15)
        CLOSE(UNIT=DKLUN1)
        GO TO 30
C
C  Error opening file
C
 25     TYPE 1025, ERRFIL
 1025   FORMAT(/5X,'**** UNABLE TO OPEN ',A15,' ****')
C
C
  30    RETURN
        END
```

```
C*********************************************************************
C
C  YESNO
C       Prompt for and accept a yes-or-no answer
C
C       Calling argument:
C       PROMPT  The string to be used for prompting.  It is output,
C               with the characters " ?  (Y/N) -> " appended to it.
C
C       Function value (LOGICAL*1):
C       .TRUE. if the answer begins with the character Y
C       .FALSE. if the answer begins with the character N
C
C       If the answer does not begin "Y" or "N", the prompt
C       is repeated until it does.
C
C       Example:
C       If YESNO is called as follows
C               LOGICAL*1 CONT,YESNO
C               CONT=YESNO('Do you want to continue')
C       the following prompt is produced
C                Do you want to continue ?  (Y/N) ->
C
C       04-Feb-85  MPK
C       06-Aug-87  RDP          Convert BYTE string to Character String
C
C*********************************************************************
C
      LOGICAL*1 FUNCTION YESNO(PROMPT)
C*********************************************************************
C
      CHARACTER PROMPT*(*),TAG*10,QUESTION*70,ANS*1
      INTEGER   LOC
C
      DATA TAG/'? (Y/N) ->'/
10    LOC = LEN(PROMPT)+1
C
      QUESTION = PROMPT
      QUESTION(LOC:LOC+10) = TAG
C
      TYPE 1000, QUESTION
1000  FORMAT(1X,A<LOC+11>,$)
      READ (5,1002,END=10,ERR=10) ANS
1002  FORMAT (A)
      YESNO=.TRUE.
      IF (ANS.EQ.'Y') RETURN
      YESNO=.FALSE.
      IF (ANS.EQ.'N') RETURN
      GO TO 10
      END
```

```
C*****************************************************************
C WRITB
C
C       Write background file to disk
C
        Rev. 15-Jun-84 lc
C       Rev. 11-Sep-84 TES
C       Rev. 07-Apr-86 WDR      Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE WRITB
C
        LOGICAL*1 FILL
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CPASS.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CMC.CMN'
C
C
C  NOTIFY OPERATOR
C
        TYPE 1003,BKFIL
 1003   FORMAT(/' Writing ',A)

C  OPEN A DIRECT ACCESS FILE WITH NAME FILNAM
C
        OPEN(UNIT=DKLUN,NAME=BKFIL,TYPE='NEW',ACCESS='DIRECT',
     X          RECORDSIZE=128,MAXREC=1,ERR=999)
C
C  UPDATE FILE NAME, DATE, TIME AND OPERATOR
C
        IF(.NOT. BACKG) GO TO 150
       BKSFIL = FILNAM
          BKDAT = NOWDAT
          BKTIM = NOWTIM
       BKOP = OPERID
C
C  WRITE DATA
C
 150    WRITE(DKLUN'1,ERR=999)BKDAT,BKTIM,BKOP,FILL,
     X        BK,SGBK,TIMEBK,PASSWD,BKSFIL
C
C  CLOSE FILE AND ACKNOWLEDGE
C
        CLOSE(UNIT=DKLUN)
        TYPE 1000,BKFIL,BKDAT,BKTIM,BKOP
 1000   FORMAT(/1X,A, ' Written ',A,2X,A,'  by ',A12)
C
        GO TO 500
C
C  ERROR IN WRITING
C
```

```
 999     TYPE 1001
 1001    FORMAT(//20X,'**** WRITB DISK ERROR'/
     X           25X,'FILE WAS NOT WRITTEN CORRECTLY ****'/)
C
 500     RETURN
         END
```

```
C*************************************************************************
C PASS
C
C       Check the password in order to allow operator to
C       enter supervisory mode

C       CALLING ARGUMENTS:
C            OK = Logical flag indicating validity of operator password
C
C       SSJ  01-Jul-82
C       REV. 08-May-84 lc
C       REV. 19-APR-85 MPK      Remove logical*1 OK
C                               (Redundant with /CPASS/
C       REV. 04-APR-86 WDR      Modified for Micro VAX II
C       REV. 06-AUG-87 RDP      Modified to accept password w/o
C                               echo to screen. All character are
C                               converted to UPPERCASE.
C
C*************************************************************************
C
        SUBROUTINE PASS(OK)
C
        CHARACTER       WORD*10
        INTEGER JSW,BIT12,NOT12,CR,LF
        REAL*8 PWORD
C
        INTEGER*4       STATUS,VKID,LENGTH
        INTEGER*4       SMG$CREATE_VIRTUAL_KEYBOARD
        INTEGER*4       SMG$READ_STRING
        INTEGER*4       SMG$DELETE_VIRTUAL_KEYBOARD
C
        INCLUDE '($TRMDEF)'
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CPASS.CMN'
C
        EQUIVALENCE (PWORD,WORD)
C
        DATA CR,LF /"15,"12/
C
C
        OK = .FALSE.
C
C  Prompt operator for password
C
        TYPE 1000
 1000   FORMAT(/' You have selected a supervisory option',
     X      ' which requires a password.'/)
C
C  Store up to 8 characters
        STATUS = SMG$CREATE_VIRTUAL_KEYBOARD(VKID,'SYS$INPUT')
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
C
        STATUS = SMG$READ_STRING(VKID,
     c                           WORD,
     c                           'Enter password ->',
     c                           10,
     c                           TRM$M_TM_NOECHO+TRM$M_TM_CVTLOW)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
```

```
 20        TYPE 1004
 1004    FORMAT(1X)
C
         STATUS = SMG$DELETE_VIRTUAL_KEYBOARD(VKID)
         IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))

C  Check if operator wanted to escape
C
         IF(WORD(1:1).EQ.' ') GOTO 500
C
C  Compare with all of the passwords
C
         DO 30 I=1,3
           IF(PWORD.EQ.PASSWD(I)) GOTO 40
 30        CONTINUE
C
C  Improper password
C  Inform operator
C
         TYPE 1002
 1002    FORMAT(//10X,'**** IMPROPER PASSWORD - NO ENTRY',
     X        ' TO SUPERVISORY MODE ****'/)
         GO TO 500
C
C  Password is OK - Enter supervisory mode
C
 40      OK = .TRUE.
C
 500     RETURN
         END
```

```
C*****************************************************************
C READB
C
C       Read background from disk
C
        Rev. 25-Jun-84 lc
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C
C*****************************************************************
C
        SUBROUTINE READB
C
        LOGICAL*1 FILL
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CBACKG.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CPASS.CMN'
        INCLUDE 'CONTRL.CMN'
C
C  The following common block files are instrument specific for
C  gamma-ray instruments
C
        INCLUDE 'CONT90.CMN'
C
C  If FROMDK, read background from spectrum file
C
        IF(FROMDK) GO TO 10
C
  Notify operator
C
 1      TYPE 1003,BKFIL
 1003   FORMAT(/' Reading ',A)
C
C  OPEN A DIRECT ACCESS FILE WITH NAME BKFIL
C
        OPEN(UNIT=DKLUN,FILE=BKFIL,ACCESS='DIRECT',TYPE='OLD',
     X           RECORDSIZE=128,ERR=999)
C
C  Read data
C
        READ(DKLUN'1,ERR=999)BKDAT,BKTIM,BKOP,FILL,
     X       BK,SGBK,TIMEBK,PASSWD,BKSFIL
C
C  Close file and acknowledge
C
        CLOSE(UNIT=DKLUN)
        IF(BEGIN)GO TO 500
        TYPE 1000,BKFIL,BKDAT,BKTIM,BKOP
 1000   FORMAT(/,1X,A, ' was written  ',A,2X,A,'  by ',
     X     A12)
        GO TO 500
C
C  Read background file from spectrum file (last record)

 10     TYPE 1003,FILNAM
        OPEN(UNIT=DKLUN,NAME=FILNAM,ACCESS='DIRECT',TYPE='OLD',
     X           RECORDSIZE=128,ERR=997)
```

```
        NRECS = MEMSIZ/128
        NREC = NRECS + 1
C
C  Check if first entry in record is dummy
C
        READ(DKLUN'NREC,ERR=998)DUM
        IF(DUM .NE. -999.0) GO TO 998
        READ(DKLUN'NREC,ERR=998)DUM,BK,SGBK,TIMEBK,BKDAT,BKTIM,
     X                     BKOP,BKSFIL
        CLOSE(UNIT=DKLUN,ERR=11)
 11     TYPE 1000,FILNAM,BKDAT,BKTIM,BKOP
        GO TO 500
C
C  Error in reading from spectrum file
C  Do not abort, but use most recent background read
C
 997    CALL ERRMSG(1,FILNAM)
        GO TO 500
 998    CALL ERRMSG(2,FILNAM)
        CLOSE(UNIT=DKLUN)
        GO TO 500
C
C  Error in reading
C
 999    ABORT = .TRUE.
        CALL ERRMSG(1,BKFIL)
        CALL ERRMSG(102,ARG)
C
 500    RETURN
        END
```

```
C*****************************************************************
C WRIT38
C
C       Routine specific to the Pu-238 isotopic instrument
C       Write constants to disk
C       Routine needs to be modified for each instrument
C
C       REV. 14-Jun-84 LC
C       REV. 6-Sep-84 CMS
C       Last edit:  11-Sep-84  TES
C       REV. 27-DEC-84 WDR      Modified for the solid isotopic
C                               instrument
C       REV. 07-APR-86 WDR      Modified for Micro VAX II
C                      WDR      Included Pu-238 common
C       Rev. 24-May-91 WDR      Modified for stabilizer settings
C
C*****************************************************************
C
        SUBROUTINE WRITC
C
        LOGICAL*1 FILL
C
C
        INTEGER*2 EMC(21), EDD(41),FILLR(58)
C
C  These common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CMCONS.CMN'
        INCLUDE 'CCAL.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CDIAG.CMN'
C
C  The following common block files are instrument specific for
C  Pu-238 isotopic instrument
C
        INCLUDE 'P238COM.CMN'
C
        EQUIVALENCE (EMC(1),BITIME),(EDD(1),DIAGMX)
C
C
C  Notify operator
C
        TYPE 1003,CONFIL
 1003   FORMAT(/' Writing ',A)
C
C  Open a direct access file with name filnam
C
        OPEN(UNIT=DKLUN,NAME=CONFIL,TYPE='NEW',ACCESS='DIRECT',
     X           RECORDSIZE=128,MAXREC=2,ERR=999)
C
C  Update file date, time and operator

        FILDAT = NOWDAT
        FILTIM = NOWTIM
        FILOP  = OPERID
```

```
C
C  Write data
C
      WRITE(DKLUN'1,ERR=999)FILDAT,FILTIM,FILOP,FILL,COMMNT,
     X        FILLR,EMC,EDD,PU240,PU242,NPFLG,NUFLG,TIMEPR,
     X        IZPEAK,IZWNDW,ZARNG,IGPEAK,IGWNDW,GARNG
C
C  Close file and acknowledge
C
      CLOSE(UNIT=DKLUN)
      TYPE 1000,CONFIL,FILDAT,FILTIM,FILOP
 1000 FORMAT(/1X,A, ' Written ',A,2X,A,'  by ',A)
C
      GO TO 500
C
C  Error in writing
C
  999 TYPE 1001
 1001 FORMAT(//20X,'**** WRITC DISK ERROR'/
     X          25X,'FILE WAS NOT WRITTEN CORRECTLY ****'/)
C
  500 RETURN
      END
```

```
C*****************************************************************
C
C READC38       (READC subroutine for PU238 isotopic instrument)
C
C       Read constants for Pu-238  sotopic plutonium gamma-ray
        analysis instrument from disk
C
C       REV. 14-JUN-84  lc
C       REV. 06-SEP-84  CMS
C       REV. 17-DEC-84  WDR     Modified for solid isotopic instrument
C       REV. 04-APR-86  WDR     Modified for Micro VAX II
C       REV. 13-JUL-87  RDP     Modified for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C       Rev. 24-May-91 WDR      Modified for stabilizer settings
C
C*****************************************************************
C
        SUBROUTINE READC
C
        LOGICAL*1 FILL,CDUM(30)
        CHARACTER SNAM*12,SYN*3
C
        INTEGER*2 EMC(21),EDD(41),FILLR(58)
C
C  These common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CLUNS.CMN'
        INCLUDE 'CMCONS.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CASSAY.CMN'
        INCLUDE 'CCAL.CMN'
        INCLUDE 'CDIAG.CMN'
C
C  The following common block files are instrument specific for
C  solid isotopic analysis (gamma ray) instruments
C
        INCLUDE 'CONT90.CMN'
        INCLUDE 'P238COM.CMN'
C
C       Specify names of constants files to be read in
C
        DIMENSION OSHAPC(15)
        DATA SNAM /'SHAPC.      '/
        DATA OSHAPC/.04,.00175,-4.,.005,3.,0.,.003,.3,.0,.0,
     +              3.,.0,.0,.0,.0/
C
C       Equivalence dummy arrays to first word of common block
C       Array size = word count of common.
C
        EQUIVALENCE (EMC(1),BITIME),(EDD(1),DIAGMX)
C
C
C  Notify operator
C
        TYPE 1003,CONFIL,COMMNT
 1003   FORMAT(/' Reading ',A,1X,A)
C
C  Open a direct access file with name filnam
```

```
C
      OPEN(UNIT=DKLUN,NAME=CONFIL,ACCESS='DIRECT',TYPE='OLD',
     X           RECORDSIZE=128,ERR=999)
C
C  Read data
   Fill = Logical*1 fill to even up word boundary
C  Cdum = Dummy fill argument to blank out comment
C  Fillr = Filler for more parameter arguments
C
      READ(DKLUN'1,ERR=999)FILDAT,FILTIM,FILOP,FILL,CDUM,
     X       FILLR,EMC,EDD,PU240,PU242,NPFLG,NUFLG,TIMEPR,
     X       IZPEAK,IZWNDW,ZARNG,IGPEAK,
     X       IGWNDW,GARNG
C
C  Close file and acknowledge
C
      CLOSE(UNIT=DKLUN)
      IF(BEGIN)GO TO 505
      TYPE 1000,CONFIL,FILDAT,FILTIM,FILOP
 1000 FORMAT(/1X,A, ' was written  ',A,2X,A,'  by ',A)
C
C     Read in constants files specific to solid isotopic plutonium
C     gamma-ray analysis
C
C     First get system id to establish file extension
C
505   DO 100 J = 1,15
      IF (CONFIL(J:J) .EQ. '.')GO TO 120
100   CONTINUE
120   K = J
      DO 150 I = 1,3
      SYN(I:I)= CONFIL(K+I:K+I)
150   CONTINUE
C
C     Read in peak shape parameters
C
C     DO 200 J = 1,3
C200  SNAM(J+6:J+6)=SYN(J:J)
C     OPEN (UNIT=DKLUN,NAME=SNAM,TYPE='OLD',ACCESS='SEQUENTIAL',ERR=995)
C     READ (DKLUN,9087,ERR=995) (OSHAPC(I), I = 1,15)
C9087 FORMAT (E15.7)
C     CALL CLOSE (DKLUN)
      GO TO 500
C
C  Error in reading
C
 995  CALL ERRMSG(1,SNAM)
      CALL ERRMSG(102,ARG)
      GO TO 500
999   CALL ERRMSG(1,CONFIL)
      CALL ERRMSG(102,ARG)
C
 500  RETURN
      END
```

```
C*********************************************************************
C  JULDAL       (Livermore version of JULDAT)
C
C       Using ASCII strings for date, time, compute Julian day
C       From Sue Johnson's Julday

C       Calling arguments:
C       IDAT - Input ascii string for date  30-apr-84
C       ITIM - Input ascii string for time  13:54:17
C       XDAT - Output Julian day
C       XTIM - Output time of day in seconds
C
C       12-APR-83 LC
C       REV. 02-MAY-84  LC      for generic package
C       REV. 31-AUG-84  CMS
C       REV. 13-DEC-84  WDR     made day 0 1/1/1900
C                               modified days calculation
C                               Note: Day 0 1/1/1900
C
C       Rev. 11-Oct-84  mjc:    IDAT is accepted as 8 characters (1-May-84)
C                               or 9 characters (10-May-84)
C       Rev. 30-Oct-84  mpk     Fix error in 11-Oct correction -
C                               tighten code
C       Rev. 31-Oct-84  mpk     Tighten code some more, and handle
C                               time format more flexibly.  This is
C                               done by copying date and time into
C                               an auxiliary array, and editing the
C                               array to convert all special characters
C                               into commas.  Then the DECODE can use
C                               the comma-delimited integer field
C                               feature.
C                               Also, correct Julian date algorithm
C                               to use integer arithmetic.
C                               In the correct algorithm, day 0 comes
C                               out to 30-OCT-79, not 28-OCT as
C                               stated previously.
C       Rev. 01-Nov-84  mpk     Leave spaces in date/time string
C                               rather than editing to commas.
C       Rev. 02-Nov-84  mpk     Latest installment in space saga:
C                               Ignore leading spaces and replace
C                               trailing spaces with commas; this
C                               makes it work right for time inputs
C                               like 11:23 (seconds omitted).
C       Rev. 10-DEC-85  WDR     Modified to calculate number of days
C                               from 1/1/1900 rather than as described
C                               above.
C       Rev.  3-Apr-86  WDR     Modified for Micro VAX II
C
C*********************************************************************
C
      SUBROUTINE JULDAT(IDAT,ITIM,XDAT,XTIM)
C
      CHARACTER XMONTH*3
      CHARACTER*1 IDAT(9),ITIM(8), DATTIM(23)
      LOGICAL*1 TERMFG
      DIMENSION NDAYS(12)

      DATA NDAYS /31,28,31,30,31,30,31,31,30,31,30,31/
C
C  The following common block files are generic for all instruments
```

```
C
      INCLUDE 'CMONTH.CMN'
C
C
C  Decode ASCII strings

      DO 5 I=1,9
 5    DATTIM(I)=IDAT(I)
      DO 6 I=1,8
 6    DATTIM(I+9)=ITIM(I)
D     TYPE 1998,DATTIM
C
C  Replace special chars ('-' or ':') and trailing spaces with commas
C
      TERMFG = .TRUE.
      DO 7 I=1,17
      IF (DATTIM(I).NE.' ') GO TO 71
      IF (TERMFG) GO TO 7
      GO TO 72
71    IF (DATTIM(I).GE.'0' .AND. DATTIM(I).LE.'9') GO TO 73
      IF (DATTIM(I).GE.'A' .AND. DATTIM(I).LE.'Z') GO TO 73
72    DATTIM(I)=','
      TERMFG = .TRUE.
      GO TO 7
73    TERMFG = .FALSE.
 7    CONTINUE
D     TYPE 1998,DATTIM
D1998 FORMAT (' EDITED DATE/TIME: ',17A1)
      IDAY = 0
      XMONTH = '   '
      IYEAR = 0
      IHOUR = 0
      IMIN = 0
      ISEC = 0
      DECODE (9,1000,DATTIM,ERR=75) IDAY,XMONTH,IYEAR
 1000 FORMAT (I3,A3,1X,I4)
75    DECODE (8,1001,DATTIM(10),ERR = 76) IHOUR,IMIN,ISEC
 1001 FORMAT (2I5,I2)
76    CONTINUE
C
C  Compare month to common
C
      IMONTH = 0
      DO 9 I=1,12
 9    IF(XMONTH .EQ. AMONTH(I))IMONTH = I
D     TYPE 1999,IMONTH,IDAY,IYEAR,IHOUR,IMIN,ISEC
D1999 FORMAT(' DATE:',3I3,'  TIME:',3I3)
C
C  Convert time to seconds
C
      XTIM = ISEC + 60. * (IMIN + 60. * IHOUR)
C
C  Convert date to days
C
C     IF (IYEAR .GT. 1900) IYEAR = IYEAR - 1900
      NMONTH = IMONTH - 1

 10   XDAT = AINT (365.25 * IYEAR)
C     1    + AINT (30.6001 * NMONTH)
      2    + IDAY
```

```
C       3    + XTIM/86400.       !uncomment to include fractional day
C
C  Add the number of days up to this month
C
        IF (NMONTH.EQ.0)GO TO 16
        DO 15 J =1,NMONTH
15      XDAT = XDAT + NDAYS(J)
16      IF(NMONTH .LT. 3)GO TO 500
        YR = IYEAR
C
C  Include an extra day for leap year
C
        IF(AMOD(YR,4.).EQ. 0.)XDAT = XDAT + 1.
C
500     RETURN
        END
```

```
C*****************************************************************
C  READNM
C
C       Allow operator to select or change a constants
C       filename in a list of constants files

C          LCS 27-FEB-81
C
C       Rev. 16-Jun-84 lc
C       Rev. 19-Feb-85 MPK      Change ACCEPTS to READS with END
C                               and ERR exits
C                               Use YESNO to ask yes/no questions
C       Rev. 04-Apr-86 WDR      Modified for Micro VAX II
C       Rev. 13-Jul-87 RDP      Modifed for uVAX II
C                               Convert BYTE strings to CHARACTER strings
C
C*****************************************************************
C
        SUBROUTINE READNM
C
        LOGICAL*1 ANS,YESNO
        CHARACTER CFLNAM(5)*12,CCMMNT(5)*30
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CFILES.CMN'
        INCLUDE 'CONFIL.CMN'
        INCLUDE 'CONTRL.CMN'
        INCLUDE 'CLUNS.CMN'
C
C

        NEWFIL = .TRUE.
        IF(BEGIN) GO TO 4
C
C  Check if constants file read is still to be used
C
 1      TYPE 1005, CONFIL,COMMNT
 1005   FORMAT(/' The constants file read was: ',A15,1X,A30//
     X     '$Do you want to select another constants file?  (Y/N) -> ')
        READ (5,1050,END=1,ERR=1) ANS
        IF(ANS .NE. 'Y' .AND. ANS .NE. 'N')  GO TO 1
        IF(ANS .NE. 'Y') NEWFIL = .FALSE.
        IF(.NOT. NEWFIL) GO TO 900
C
C  Open file to read names of 5 current constants files
C
 4      OPEN(UNIT=DKLUN,FILE=NAMSFL,ACCESS='SEQUENTIAL',
     X       TYPE='OLD',ERR=5)
        GO TO 80
C
C  Problem in open of names file
C
 5      CALL ERRMSG(1,NAMSFL)
        CALL ERRMSG(102,ARG)
        ABORT = .TRUE.
        IF (ABORT) CALL EXIT

C  Read names from file
C
 80     DO 100 I=1,5
```

```
 100      READ(DKLUN,1000) CFLNAM(I),CCMMNT(I)
 1000     FORMAT(A12,A30)
C
C  If just beginning, choose file 1
C
        IF(BEGIN) JANS = 1
        IF(BEGIN) GO TO 280
C
C  Type out the current constants file names & select
C  appropriate file
C
 150    TYPE 1010
 1010   FORMAT(//' The current constants files are:'/)
        DO 200 I=1,5
 200      TYPE 1011,I,CFLNAM(I),CCMMNT(I)
 1011     FORMAT(3X,I1,3X,A12,4X,A30)
C
 210    TYPE 1012
 1012   FORMAT(//'$Enter constants file selected (1-5)',
     X       '  or 6 for new file -> ')
        READ(5,1020,END=150,ERR=150) JANS
 1020   FORMAT(I1)
        IF(JANS.LT.1.OR.JANS.GT.6)GO TO 150
        IF(JANS .EQ. 6) GO TO 400
C
C  Confirm filename
C
  250      TYPE 1030,CFLNAM(JANS)
 1030   FORMAT(/'$Constants file = ',A12,' OK? (Y/N) -> ')
        READ(5,1050,END=250,ERR=250) ANS
        IF(ANS .EQ.'N') GO TO 210
        IF(ANS .NE.'Y') GO TO 250
C
C  Read filename & comment into common block
C
 280    CONFIL=CFLNAM(JANS)
        CONFIL(11:12)='  '
 350      COMMNT=CCMMNT(JANS)
        CLOSE(UNIT=DKLUN,ERR=351)
 351    GO TO 700
C
C  Input new constants file name & comment
C  Attempt to open new file
C
 400    TYPE 1040
 1040   FORMAT(/'$Enter new constants filename -> ')
        READ (5,1050,END=400,ERR=400) CONFIL
 1050   FORMAT(A30)
C
 401    TYPE 1041
 1041   FORMAT(/'$Enter comment (30 char) -> ')
        READ (5,1050,END=401,ERR=401) COMMNT
C
C  Check if new file is permanent & which file it replaces
C
        IF ( .NOT.YESNO('Is this file a permanent replacement')
     1  ) GO TO 600
C
 410    TYPE 1044
 1044   FORMAT(/'$Which file does the new file replace?  (1-5) -> ')
```

```
        READ (5,1020,END=410,ERR=410) JANS
        IF(JANS.LT.1.OR.JANS.GT.5) GO TO 410
C
 420      CFLNAM(JANS)=CONFIL
 421      CCMMNT(JANS)=COMMNT

C  Write new file name list to disk
C
C         REWIND DKLUN
        CLOSE (UNIT=DKLUN)
        OPEN(UNIT=DKLUN,NAME=NAMSFL,ACCESS='SEQUENTIAL',
     X       TYPE='OLD',ERR=5)
        DO 430 I=1,5
 430      WRITE(DKLUN,1100) CFLNAM(I),CCMMNT(I)
 1100     FORMAT(1X,A12,A30)
C
C  Close names file
C
 600    CLOSE(UNIT=DKLUN,ERR=700)
C
C  Try opening the new constants file
C
 700    OPEN(UNIT=DKLUN1,NAME=CONFIL,TYPE='OLD',ERR=999)
        CLOSE(UNIT=DKLUN1,ERR=701)
 701    GO TO 900
C
 999    CALL ERRMSG(1,CONFIL)
        GO TO 150
C
 900    BEGIN = .FALSE.
        RETURN

        END
```

```
C*****************************************************************
C DATTIM
C
C       Retrieve, verify or change the system date & time
C
        SSJ 25-JAN-79
C       REV. 13-SEP-83 LC
C       Rev 02-Jan-85 MPK       Include option to sync to date and
C                                time received from VAX and stored in
C                                the CVAX common. Adapt to the NSR/FMF
C                                generic program. Add automatic operator
C                                input of date and time if date and time
C                                are found to be unreasonable.
C       Rev 14-Jul-87 RDP       Modified for uVAX II
C       Rev 18-Sep-87 RDP       Remove Operator from initializing date
C                               and time.(IOPT=0)
C
C       Options:
C
C               0 - Get initial date and time from operator input
C                   and give to monitor.
C               1 - Update date and time from monitor.
C               2 - Update and let operator verify or change.
C               3 - Sync to VAX date and time (note that these
C                   must have been set up by a call to VAX
C                   communication immediately before calling
C                   DATTIM).
C
C*****************************************************************
C
        SUBROUTINE DATTIM(OPTION)

        INTEGER OPTION
C
C  The following common block files are generic for all instruments
C
        INCLUDE 'CDATE.CMN'
        INCLUDE 'CMONTH.CMN'
        INCLUDE 'CVAX.CMN'
C
        CHARACTER*9 INPSTR
        CHARACTER*23 DATE_TIME
C
        LOGICAL*1 VFD
C
        INTEGER*4 STATUS, SYSTIM(2), SYS$BINTIM
C
        INTEGER*2 SYS$SETIME
C
        EXTERNAL SYS$BINTIM, SYS$SETIME
C
        DATA AMONTH /'JAN','FEB','MAR','APR','MAY','JUN',
        1             'JUL','AUG','SEP','OCT','NOV','DEC'/
        DATA ALOWDT /0./          !Julian date for 01-Jan-1900
        DATA AHIDAT /36525./      !Julian date for 31-Dec-99
C
   Set internal option variable.  If it is zero, force operator
   input.  If it is 3, sync to date and time from VAX.
C
        IOPT=OPTION
```

```
        IF (IOPT.LE.0) GOTO 100
        IF (IOPT.GE.3) GOTO 300
        VFD=.FALSE.
C
C  Update alphanumeric date and time.

 10     CALL DATE (NOWDAT)
        CALL TIME (NOWTIM)
C
C  If the date and time have been operator verified, return now.
C
        IF (VFD) GOTO 500
C
C  For option 1, return if date and time from VAX are reasonable;
C  otherwise, make the operator input them by setting the internal
C  option to 0.
C
        IF (IOPT.NE.1) GOTO 40
        CALL JULDAT (NOWDAT,NOWTIM,ADAT,ATIM)
        VFD=.TRUE.
        IF ((ADAT.GE.ALOWDT).AND.(ADAT.LE.AHIDAT).AND.
     1      (ATIM.GE.0.0).AND.(ATIM.LE.86399.0)) GOTO 500
        IOPT=0
        GOTO 100
 40     CONTINUE
C
C  Print out date and time for operator verification.
C
        TYPE 1000,NOWDAT,NOWTIM
 1000   FORMAT(/1X,A,2X,A)
        VFD=.TRUE.

C  Allow operator to change date.
C
 100    WRITE(*,1003)
 1003   FORMAT(' '/'$Enter Date in the form 01-FEB-85    ->')
        IF (IOPT.NE.0) WRITE(*,1006)
 1006   FORMAT('$ (or RETURN for no change)')
        READ(5,1014,END=100,ERR=100) INPSTR
C!      READ(5,1014,END=100,ERR=100) NCHAR,INPSTR
C! 1014 FORMAT(Q,A)
 1014   FORMAT(A)
        IF ((IOPT.NE.0).AND.(INPSTR.EQ.' ')) GOTO 200
        IF (INPSTR(3:3).NE.'-') GOTO 100
C!      IF (NCHAR.NE.0) GOTO 100
        IF (IOPT.EQ.0) GOTO 110
        GOTO 200
C
C  We appear to have a date from the operator.  Call JULDAT to
C  parse it, and check its components for reasonableness.
C
 110    VFD=.FALSE.
        CALL JULDAT(INPSTR,NOWTIM,ADAT,ATIM)
        IF (IYEAR.GE.84 .AND. IYEAR.LE.99) GOTO 120
        TYPE 1024
 1024   FORMAT (//20X,'**** INCORRECT YEAR NUMBER ****'/)
        GOTO 100
 120    IF (IMONTH.GE.1 .AND. IMONTH.LE.12) GOTO 130
        TYPE 1025
 1025   FORMAT (//20X,'**** INCORRECT MONTH NAME ****'/)
```

```
        GOTO 100
 130    IF (IDAY.GE.1 .AND. IDAY.LE.31) GOTO 170
        TYPE 1020
 1020   FORMAT (//20X,'**** INCORRECT DAY NUMBER ****'/)
        GOTO 100
  170   DATE_TIME(1:8)=INPSTR(1:8)
        J = INDEX(DATE_TIME,'-')+1
        IF (J.GT.3) GOTO 175
        TYPE 8000
8000    FORMAT(//20X,' **** INCORRECT FORMAT  04-APR-86 ****')
        GOTO 100
  175   J = INDEX(DATE_TIME(J+1:),'-')+1+J
        DATE_TIME(J:J)='1'
        DATE_TIME(J+1:J+1)='9'
        DATE_TIME(J+2:J+2)=INPSTR(J:J)
        DATE_TIME(J+3:J+3)=INPSTR(J+1:J+1)
C
C  Allow operator to change time.
C
 200    WRITE(*,1004)
 1004   FORMAT(' '/'$Enter Time in the form 01:47:59    ->')
        IF (IOPT.NE.0) WRITE(*,1006)
        READ(5,1014,END=200,ERR=200) INPSTR
        IF ((IOPT.NE.0).AND.(INPSTR.EQ.' ')) GOTO 500
        IF (INPSTR(3:3).NE.':') GOTO 200
        IF (IOPT.EQ.0) GOTO 210
        GOTO 300
C
C  We appear to have a time from the operator.  Call JULDAT to
C  parse it, and check its components for reasonableness.
C
  210   VFD=.FALSE.
        CALL JULDAT (NOWDAT,INPSTR,ADAT,ATIM)
        IF (IHOUR.GE.0 .AND. IHOUR.LE.23) GO TO 220
        TYPE 1034
 1034   FORMAT (//20X,'**** INCORRECT HOUR NUMBER ****'/)
        GO TO 200
 220    IF (IMIN.GE.0 .AND. IMIN.LE.59) GO TO 230
        TYPE 1036
 1036   FORMAT (//20X,'**** INCORRECT MINUTE NUMBER ****'/)
        GO TO 200
 230    IF (ISEC.GE.0 .AND. ISEC.LE.59) GO TO 270
        TYPE 1038
 1038   FORMAT (//20X,'**** INCORRECT SECOND NUMBER ****'/)
        GO TO 200
  270   CONTINUE
C
C  Load DATE_TIME character string with time and pad with blanks
        DATE_TIME(12:12)=' '
        DATE_TIME(13:20)=INPSTR(1:8)
        DATE_TIME(21:23)='   '
C       TYPE 9000,DATE_TIME
9000    FORMAT(1X,23A)
        STATUS = SYS$BINTIM(DATE_TIME,SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
        STATUS = SYS$SETIME(SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
C
C  Repeat update and verification.
C
```

```
        IOPT=2
        GOTO 10
C
C  Sync to the VAX time.
C
 )0     CALL JULDAT(VAXDAT,VAXTIM,ADAT,ATIM)
C. Get VAX time from comm
C!      CALL COMM('TS')
C  Load DATE_TIME character string with time and date (modified date
C    to meet Date String requirements.
        DATE_TIME(1:9)  = VAXDAT
        DATE_TIME(10:11)= DATE_TIME(8:9)
        DATE_TIME(8:9)  ='19'
        DATE_TIME(12:12)=' '
        DATE_TIME(13:20)= VAXTIM
        DATE_TIME(21:23)='   '
C       TYPE 9000,DATE_TIME
C Update Time
        STATUS = SYS$BINTIM(DATE_TIME,SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
        STATUS = SYS$SETIME(SYSTIM)
        IF (.NOT.STATUS) CALL LIB$SIGNAL(%VAL(STATUS))
 500    RETURN
C
C
        END
```