PHENIX MUON DAQ SYSTEM
(CODA)

Elham Nikanjam

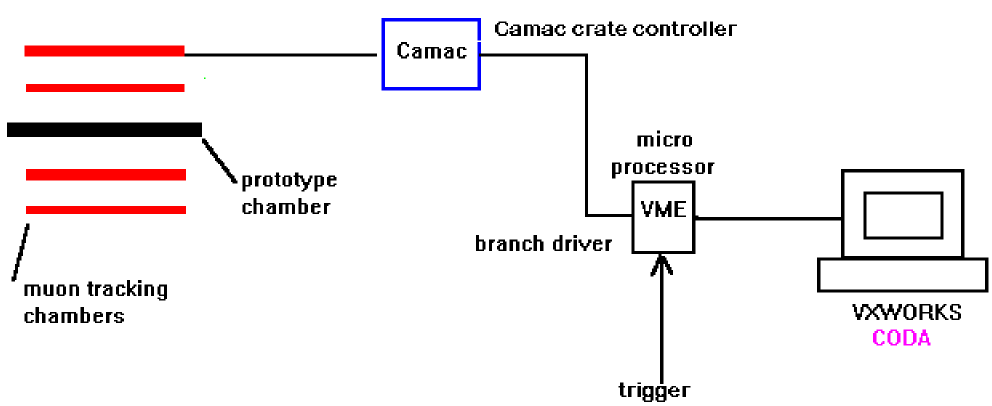
email: nikanjam@p25hp.lanl.gov

A. Introduction:

The main goal for setting up a DAQ system is to be able to read raw data from a CAMAC module, graph it, analyze it, and finally be able to store it for further study. This article is a detailed guide on how to set up a DAQ system and be able to run it. For this purpose, an overview of the system is presented and detailed descriptions follow. Note that there are different ways of setting up a CODA(CEBAF On-line Data Acquisition) DAQ system. The system presented in this article is the one currently being used at Los Alamos National Laboratory for the Phenix muon project.

B. General view of the system:

A schematic of the DAQ hardware is shown in this figure.



C. Hardware:

To set up a CODA DAQ system, the hardware modules listed in table 1 are required. The core of the system is composed of a UNIX workstation, a VME crate which contains at least a microprocessor and a VME CAMAC Branch Driver, and (if you want to read out CAMAC electronics), a CAMAC crate with a CAMAC crate controller.

Table 1: Hardware

requirements	
VME micro processor	CAMAC crate controller
VME-CAMAC branch driver	CAMAC crate
VME crate	UNIX workstation

Table 1: A brief list of hardware requirements and their specifications are shown in the this table.

1. Setting up the hardware:

The VME CAMAC branch driver that sits in the VME crate is connected to the CAMAC crate controller in the CAMAC crate through a branch driver cable. Table 2 lists the microprocessors that can be used with the CODA system. The PHENIX Muon DAQ uses the MVME 162-13 microprocessor. Note that Jumpers on the module should be in a specific position so that the microprocessor can work with CODA (see figure 2). The microprocessor is connected to the UNIX workstation through the ethernet. The workstation contains Vxworks and CODA application software through which we can talk to the VME microprocessor. It also contains CAMAC codes to read the CAMAC modules and analyzer programs. Vxworks is a multi-tasking real-time operating system that will boot up into the VME microprocessor.

Table2: microprocessors
MVME 167
MVME 162

Table 2: Microprocessors that can be used with CODA are listed in this table.

2. Trigger:

The trigger is an electronic signal resulting from external electronic logic that defines the occurrence of a physics event for which we want to accumulate data from the CAMAC modules. In the CES branch driver the trigger should be a NIM signal going into the INT4 on the branch driver with minimum duration of 50 nano seconds. The trigger should be delayed long enough so that the CAMAC modules have time to convert before being read.

D. Software:

1. Software required for microprocessor:

On the UNIX workstation two files need to be modified to boot the VME microprocessor. These are .rhosts that resides in your root directory and the bootest file that is referred to by the microprocessor. The .rhosts file allows the microprocessor to remotely login to the UNIX workstation without requiring a password. The name and/or IP address of the microprocessor must be added to this file (see appendix 1). An example of a bootest file is shown in the appendix 2. The name and IP address of the UNIX workstation must be put in this file also.

Table 3: Files to be edited	
.rhosts	.cshrc
bootest	rcNetwork
*.crl	rcRunTypes
analyzer files	*.config

Table 3: This table lists all the files that need to be edited in order to setup and run a DAQ system.

2. Setting up the environment:

A complete list of environment variables is stored in a file called CODA_setup. After this file is edited to correspond to your particular system, it should be sourced in the .cshrc file. Examples of these two files are shown in the appendix 3. Note that there should be no commands in the .cshrc or .login file that writes to the screen, or you will get a Bus error when the VME microprocessor tries to log into the workstation to get Vxworks (and ..cshrc and .login are executed).

3. CAMAC and Analyzer Files:

There are a few more files that have to be edited before running CODA (note table 3). The *.crl file contains the CAMAC commands to be executed during data acquisition and uses a CODA specific format. Other FORTRAN files are used as analyzer programs. When CODA is run, an x-window will be displayed through which all the processes can be controlled. Different options can be chosen by clicking on the proper buttons. One of the CODA windows is shown below.

For each button on this window, there is a set of commands in the *.crl file that will be executed when that button is chosen. For example if prestart button is chosen, all the commands in the prestart section of *.crl will be executed. An analyzer file containing different subroutines is also run when CODA is running. Clicking on different buttons, activate the related subroutines. Userprestart is an example of one of the subroutines. Table 4 shows the relationship between the buttons on the CODA window, sections of the *.crl file, and the analyzer files' subroutines.

CODA button	down load	prestart	end	pause	go	auto	done

*.crl section	down load	prestart	end	pause	go	trigger	done
analyzer subroutine	usr-download	usr-prestart	usr-end	usr-pause	usr-go	usr-event	usr- dump

Table 4: This table lists some of the DAQ system buttons and the correspondence sections of *.crl and analyzer files. First column lists the source of comparison. Second row shows the related *.crl sections and the third row lists the related subroutines in an analyzer file.

0a. The *.crl file:

This file holds the CAMAC codes for the system and it is used to make the CODA "event builder". An example of this file is shown in appendix 5. Refer to appendix C of the CODA manual for the definition of any specific commands. Some of these commands are derivatives of the cnaf command. For example

write kkk into BRANCH,CRATE1,10,A1,F16

will write data word kkk into the module present in slot 10 and crate 1.

After this file is edited you can compile it using the following command:

makelist file-name 5.1

for example if your file name is camac.crl, give this command:

makelist camac.crl 5.1

Before moving on to the next step, check to make sure the file camac.o has been created. You will need this file later on in the process.

1b. The analyzer files:

The second file you need is a FORTRAN file which reads in the data, stores it and/or analyzes it. Note that the sections here should correspond to the sections in the *.crl file. Note the example in the appendix 6. You can compile this file using the following command:

CODAf77 file-name [other file names]

Example:

CODAf77 ebana_test

23. The Network files:

After these two files are ready, you have to let CODA know where they are located. To do this, edit the following files.

DAQ/rcDatabase/rcNetwork

DAQ/rcDatabase/physics.config

Replace the name of the *.crl and FORTRAN file with yours.

The rcNetwork file lists all the possible event builders and analyzers in the system. The physics.config file specifies which event builder and analyzer to use for running physics. Each application is called a RunType and should be specified in the network files. To create various RunTypes, follow these directions:

a. Choose a name for the RunType and include it in the file "rcRunTypes". CODA refers to this file for a list of possible RunTypes. Increment the number in front of the last RunType name for this file (see appendix 9). For example if the last RunType is "*wangdata 5*", your RunType will be "*your RunType name 6*". These numbers determine only the order in which the RunType names appear in the "options" table.

b. For running each RunType, CODA needs to know which event builder (EB), analyzer (ANA) and readout controller (ROC) to use. Create a *.config file and specify this information. For example for physics RunType (appendix 8) we have:

```
ROC0   /usr1/muondaq/daq/run/second.o
EBP
ANAP    /usr1/muondaq/daq/run/ebana_test.log
```

In this example `second.o` is a *.crl executable file. `EBP` and `ANAP` are just arbitrary names. For us, **P** represents **physics** RunType. These names are referred to in the next section.

c. You need to tell CODA where to look for event builder (EB) and analyzer (ANA) files. Open the file "rcNetwork". Here is an example of what you should add to this file:

Name Num Type Host BootScript

!---- --- ---- ---- ---------- -----------------

EBP 2 EB $NODE $CODA_BIN/coda_activate -p /usr1/muondaq/daq/run/ebana_test

ANAP 3 ANA $NODE $CODA_BIN/coda_activate -p /usr1/muondaq/daq/run/ebana_test

Note that the names **EBP** and **ANAP** are used here. The numbers 2 and 3 refer to event builder and analyzer and not to the listing numbers in the RunType file.

E. Running CODA:

At this stage the system is ready and you can run CODA. Type in RunControl at the prompt and enter. A window will appear. Click on CONFIGURE. The next window will ask you for the RunType. Choose one and click on OK. Down load the file you have chosen by clicking DOWNLOAD and begin data acquisition by choosing AUTOSTART. If the event number at the bottom right corner of the screen is changing, that means data is being collected.

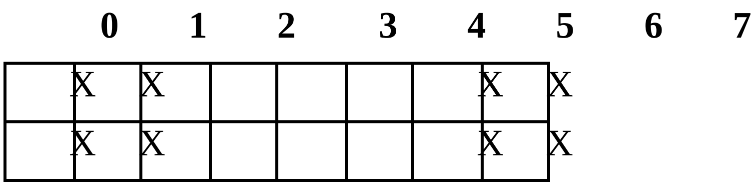
Depending on the commands in the FORTRAN file, data can be stored in a regular file, ntuple file (for histograms), or both. By adding a print command to the FORTRAN file, data can be looked at while they are being collected. In that case, the FORTRAN file should be started before CODA starts in some other window by typing ebana (if your analyzer is called ebana).

The FORTRAN program will end automatically when you exit CODA. Note that every time you start up the CODA, a new output file is created. For this reason, you need to save this file under some other name before collecting more data.

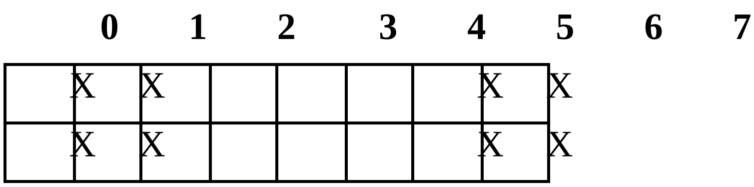
Figure 2:

The branch driver jumper settings are shown in this figure.

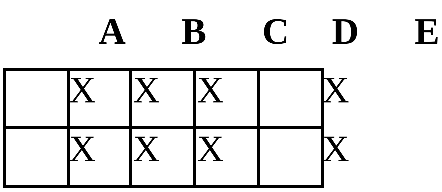
J1: Vector Number for Interrupt Level 4



J2: Vector Number for Interrupt Level 2

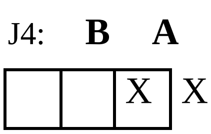


J3: Time-out selection - variable between 2 micro seconds and 134 seconds

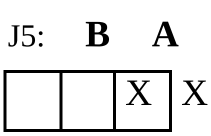


J4, J5, J6, J7 AND J8: Front panel level select plus ACK

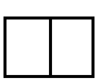
J4:



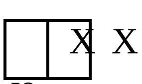
J5:



J6:



J7:



J8:

Appendix 1 : An example of .rhosts file

roc-muon.lanl.gov
128.165.86.79

Appendix 2: An example of bootest file

```
hostAdd "mudaq.lanl.gov", "128.165.86.74"
hostAdd "mudaq", "128.165.86.74"
< ~/boot/mybootsc
< $CODA/VXWORKS68K51/etc/boot040
cd "$CODA/CES"
ld < camacTest.o
ld < davesTest.o
```

Appendix 3: An example of CODA_setup file

```
#!/bin/csh
#  echo ' .CODA_setup executing...'
#
# File:
#       $CODAHOME/.setup
#
# Description:
#       Setup file for CODA
#
# Author:
#       Chip Watson
#       CEBAF Data Acquisition Group
# Modified by Tom Kozlowski (LANL) 9-jun-95
# added GCC_EXEC_PREFIX  14 dec 95 Hubert van Hecke
```

```
# Set up CODA specific environment variables

      setenv CODA       /coda-v1.4
      setenv OSTYPE      `uname|sed 's/-/_/'`
#      setenv RCDATABASE  $CODA/$OSTYPE/examples/rcDatabase
      setenv RCDATABASE  ~/daq/rcDatabase
      setenv RCDEFAULTS  $RCDATABASE/noDV

      setenv CODA_BIN $CODA/$OSTYPE/bin
      setenv CODA_LIB $CODA/$OSTYPE/lib
      setenv GCC_EXEC_PREFIX \
           /vxworks/dist-5.1.1/gnu/hp9700.68k/lib/gcc-lib/
      alias RunControl "rcLock"

# added this from .login; This file is executed from .cshrc, and therefore
# gets done also for spawned processes, where .login does NOT get done.
  setenv LIBVER pro
  setenv CERN_ROOT /cern/$LIBVER
# JPSullivan added cernlib to this October 23, 1992
set path=(/bin /usr/bin /usr/contrib/bin /usr/local/bin /usr/bin/X11 \
      $CERN_ROOT/bin /usr/local/bin /cern/pro/bin \
        /dd/v2.5/bin/HP-UX .)
set path = ($path $CODA_BIN /vxworks/dist-5.1.1/gnu/hp9700.68k/bin)

# Define host environment variables

      if (-e ../scripts/hosts) source ../scripts/hosts
```

Appendix 4: An example of .cshrc file

```
#
set notify
setenv CODA /coda-v1.4
 source ~/.coda_setup
 alias trigger ~/camac/cam01
 alias single_pulse ~/camac/single_pulse
```

```
 alias clearI 'cnaf 128.165.86.79 1 30 9 24'
source ~/.coda_setup
```

Appendix 5: An example of a *.crl file

This file contains all the CAMAC codes for the system. Each section of the file is activated when the proper button of the CODA window is hit. For example commands for clearing the ADC are executed when the prestart button is activated. Collecting data takes place when trigger is activated.

```
! this is file ~/coda/run/test_camlist.crl
!--------------------------------------!------------------
camac readout                                  ! camac hardware will be used
!polling                              ! (default is interrupt)
!--------------------------------------!-----------------------------
  BRANCH = 0
  CRATE  = 1
  NIM_IN   = 21            ! nim in
  OUTREG = 12          !Output registe
  LECROY4302 = 20
  SLOT26 = 26
  MAIN   = 1               ! these are all the
  SLOT5  = 5               ! modules that we
  SLOT9  = 9               ! are trying to read.
  SLOT13 = 13              !
  SLOT14 = 14              !
  qqq    = 64
  ggg    = 16385
  A0     = 0
  A1     = 1
  F0     = 0
  F2     = 2
  F9     = 9
  F10    = 10
  F16    = 16
  F17    = 17
  F26    = 26
```

```
  F24    = 24
  nwords = 70
  yyy    = 49152
                                        !  64 data + 6 headers, trailers
  variable nread, zero, one, three, testvar, channel, slot,xxx, mmm, kkk
!--------------------------------------!----------------------------
begin download
  log inform "TEST_CAMLIST:: nothing to download \n"
end download

!--------------------------------------!----------------------------
begin prestart
  log inform "TEST_CAMLIST:: prestarting - reset camac crate \n"
  reset crate 0
  clear CRATE inhibit 1
  control BRANCH,CRATE,30,9,24
mmm = qqq
  write mmm into BRANCH,CRATE,17,A0,F16                                ! declare
LAM location:

  control BRANCH,CRATE,1,A0,F9
  control BRANCH,CRATE,5,A0,F9
  control BRANCH,CRATE,10,A0,F9
  control BRANCH,CRATE,13,A0,F9
  control BRANCH,CRATE,14,A0,F9

  kkk = ggg
  write kkk into BRANCH,CRATE,1,A0,F16
  kkk = kkk + 4
   write kkk into BRANCH,CRATE,5,A0,F16
   kkk = kkk + 5
   write kkk into BRANCH,CRATE,10,A0,F16
   kkk = kkk + 3
   write kkk into BRANCH,CRATE,13,A0,F16
   kkk = kkk + 1
   write kkk into BRANCH,CRATE,14,A0,F16
```

```
  link trigger lam BRANCH,CRATE,SLOT26,A0
  log inform "TEST_CAMLIST:: Hardware initialised - ready to go\n"
  mmm = mmm - 1
  write mmm into BRANCH,CRATE,17,A0,F16
end prestart

!--------------------------------------!----------------------------
begin end
  log inform "TEST_CAMLIST:: end list executing - trigger disabled\n"
end end

!--------------------------------------!----------------------------
begin pause
  log inform "TEST_CAMLIST:: pause list executing - trigger disabled\n"
end pause

!--------------------------------------!----------------------------
begin go
  log inform "TEST_CAMLIST:: Go list executing - enabling trigger\n"
end go

!--------------------------------------!----------------------------
begin trigger
  if testvar is less than 512 then
    testvar = testvar + 1
  else
    testvar = 1
  end if
  mmm = qqq
   write mmm into BRANCH,CRATE,17,A0,F16
  write testvar into BRANCH,CRATE,OUTREG,A0,F17

    channel = 0
    while channel is less than 16
      read BRANCH,CRATE,1,channel,F2
      channel = channel + 1
    end while
```

```
    channel = 0
    while channel is less than 16
      read BRANCH,CRATE,5,channel,F2
      channel = channel + 1
    end while
    channel = 0
    while channel is less than 16
      read BRANCH,CRATE,10,channel,F2
      channel = channel + 1
    end while
channel = 0
    while channel is less than 16
      read BRANCH,CRATE,14,channel,F2
      channel = channel + 1
    end while

   mmm = mmm - 1
  write mmm into BRANCH,CRATE,17,A0,F16
end trigger


!--------------------------------------!----------------------------
begin done
control BRANCH,CRATE,MAIN,A0,F9        ! clear MAIN
  control BRANCH,CRATE,1,A0,F9
  control BRANCH,CRATE,5,A0,F9
  control BRANCH,CRATE,10,A0,F9
  control BRANCH,CRATE,13,A0,F9
  control BRANCH,CRATE,14,A0,F9
end done

!--------------------------------------!----------------------------
begin status
  log inform "TEST_CAMLIST:: nothing to do in status \n"
end status
```

Appendix 6: An example of an analyzer file

Each section of this file refers to a correspondence section in the *.crl file and also the CODA window. For example userevent is activated at the same time as trigger and that is when the go button on the CODA window is activated.

```
c**************************************************************************
c
c  file = ebana_test.f

# 1 "ebana_test.f"
c-------------------------------------------------------------------------*
c  Copyright (c) 1991, 1992  Southeastern Universities Research Association,
c                            Continuous Electron Beam Accelerator Facility
c
c    This software was developed under a United States Government license
c    described in the NOTICE file included as part of this distribution.
c
c CEBAF Data Acquisition Group, 12000 Jefferson Ave., Newport News, VA 23606
c      heyes@cebaf.gov   Tel: (804) 249-7030    Fax: (804) 249-7363
c-------------------------------------------------------------------------*
c
c       CODA
c
c       Example FORTRAN user analysis program.
c
c       The program calls rc_open to connect to the general CODA run-control
c       the routine rc_service MUST be called within any tight loops for
c       the run-control interface to retain control.
c       The routine da_getevent returns with a status of zero if there was
c       an event to be analysed. The other two parameters are an array and
c       an integer initialised to the size of the array, this integer value
c       is modified to reflect the true event size.
c=========================================================================c
      program usrmain

       implicit none
```

```
c-----------------------------------------------------------------------c
c       We will be an analysis program so need the analysis services
c-----------------------------------------------------------------------c
        real rc_service_ana, rc_service_eb

      common/rc_service_ana/rc_service_ana
        common/rc_service_eb/rc_service_eb
c-----------------------------------------------------------------------c
c       open communication with run control
      open(unit=8,file='/usr1/muondaq/daq/run/myout3.dat',
     &       status='unknown')
      call rcService(rc_service_eb)
      call rcService(rc_service_ana)

      write (8,*)' trying call to dalogopen'
      write (8,*)' back from call to dalogopen - calling dalogmsg'

      call daLogMsg(" ************   logging link open from ebana 1")
      call daLogMsg(' ***********    logging link open from ebana 2')

      write (8,*) ' **************    logging link open from ebana'

      call rcExecute()
c *** WARNING *** no code below rcExecute() gets executed! 21 Dec 95 HvH
c *** WARNING *** Put your code above
      end                              ! usrmain

c=======================================================================c
      subroutine usrEvent(event, len, status)

c-----------------------------------------------------------------------c

c       put here anything that you need to do with the event.  !hbook
c-----------------------------------------------------------------------c
        implicit none

        include 'hist.inc'
```

```
      include 'rawdata.inc'
      include 'scint.inc'
      include 'delaych.inc'

      integer i, j, k, numwrd
      integer status
      logical lfirst, baddata
      character*48 frmt
      common /flags/ lfirst
      data lfirst /.true./

C  Fill NTUPLE array with raw data values, and fill NTUPLE.

      DO I = 1, 64
        VALUES(I) = FLOAT(EVENT(I+9))
      END DO

      BADDATA = .FALSE.
      DO I = 1, 32
        IF (VALUES(I) .GT. 2000) BADDATA = .TRUE.
      END DO

      IF (.NOT. BADDATA) THEN
        call hfn(1,values)              !hbook
      END IF

c---------------------------------------------------------------------c
C  Put raw delay-line chamber data into array ICHMB

      J = 11
      DO I = 1, 4
       DO K = 1, 8
        ICHMB(K,I) = EVENT (J)
        J = J + 1
       END DO
      END DO
```

```
C  Put raw scintillator into appropriate variables

       S1T = EVENT(44)
       S2T = EVENT(45)
       S3T = EVENT(46)
       S4T = EVENT(47)
       S1PH = EVENT(48)
       S2PH = EVENT(49)

c     call daLogMsg("USREVENT:: here...")

C  Call routine which calculates positions in delay-line
C  drift chamber.

       call TIMING

C  Write out raw data

      if (lfirst) then
       lfirst = .false.
       write (6,20)
       write (8,20)
 20    format(/,
     &    '     <---- standard physics event ----- - - - - - - -    ',/,
     &    '            <---- event ID ----> <-- ROC - - - - - -    ',/,
     &    ' len ev 10CC 4= C000dt00 ev clas  fr 1ev.               ',/,
     &    '   type tag len tag 01nm #    sum len # <-- data --- - - ',/,
     &    ' --------------------------------------------------------')
      endif

      write (frmt,30) len-8
 30 format ('(2z3,z6,z2,z9,4z3,z4,',i2.2,'z5,z9)')
      write (8,frmt)   len, (event(i),i=1,len+2)
      write (6,frmt)   len, (event(i),i=1,len+2)

       write(6,*) values(32)
```

```
      numwrd = event(8)

      end                              ! usrevent

c==================================================================c
      subroutine usrdownload(fname)
c------------------------------------------------------------c
c       can"t think of anything to go here...
c       you could open a file and read in scale factors etc though...
c------------------------------------------------------------c
        implicit none

        character*(*) fname

C  Read in code constants:

      call datain

      call daLogMsg("USRDOWNLOAD:: here...")
      write (8,*) ' USRDOWNLOAD:: here...'
      end                         ! usrdownload

c==================================================================c
      subroutine usrprestart(rn,rt)
c------------------------------------------------------------c
c     These are all for hbook
c------------------------------------------------------------c
c     integer nvar
        implicit none

      include 'hist.inc'
        integer rn, rt
        integer istat

      call hlimit (500000)
      write(*,*)'booking histograms'
      call hbookn(1, 'rawdata', nvar,'aptuple', nvar*200, nnames)
```

```
      call hropen (66, 'aptuple', 'grout3', 'N', 1024, istat)
c---------------------------------------------------------------------c
c       put here anything you want doing when the analysis program is
c       prestarted, i.e. Just before a new run...
c---------------------------------------------------------------------c
      call daLogMsg("USRPRESTART:: here...")
      write (8,*) ' USRPRESTART:: here...'

      end                         ! usrprestart

c=====================================================================c
      subroutine usrend
c---------------------------------------------------------------------c
c       usrend does all diagnostic analysis of Ntuple 9 generated by coda.
c       The code below is almost identical to the stand alone test program
c       analize.f.  All histograms and the Ntuple are saved in the file
c       coda_ntup.dat.
c---------------------------------------------------------------------c
       implicit none

      logical     lfirst
      common /flags/ lfirst
       integer icycle
c---------------------------------------------------------------------c

      call hcdir('//aptuple',' ')    !hbook
      call hrout(0,icycle,'')         !hbook
      call hrend('aptuple')           !hbook
      close (66)                      !hbook
      lfirst = .true.
      call daLogMsg("USREND:: here...")
      write (8,*) ' USREND:: here...'
      close (8)
      end                         ! usrend

c=====================================================================c
      subroutine usrpause
```

```
c--------------------------------------------------------------------c
c      you probably won"t need anything here, this routine gets called
c      when pause run is selected...
c--------------------------------------------------------------------c
       implicit none

      call daLogMsg("USRPAUSE:: here...")
      write (8,*) ' USRPAUSE:: here...'
      end                   ! usrpause

c==========================================================================c
      subroutine usrgo
c--------------------------------------------------------------------c
c      put here anything you need doing just as the run starts or after
c      a pause. NB difference from usrprestart which gets called only
c      once per run while usrgo gets called after a pause too.
c--------------------------------------------------------------------c
       implicit none

      call daLogMsg("USRGO:: here...")
      write (8,*) ' USRGO:: here...'
      end                          ! usrgo

c==========================================================================c
      subroutine usrdump
c--------------------------------------------------------------------c
c      dump all the histograms to a file so PAW can read them...
c--------------------------------------------------------------------c
       implicit none

      call daLogMsg("USRDUMP:: here...")
      write (8,*) ' USRDUMP:: here...'
      end                          ! usrdump

c==========================================================================c
```

Appendix 7: An example of rcNetwork file

```
!+
! File:-
!    rcNetwork
!
! Description:-
!    Network Configuration file.
!
!    A Host of $NODE implies the same node as RunControl.
!
! -p lines after EB, ANA, give default event builders and
!  analyzers to use if the DOWNLOAD does not work on
!  configured event type
!-

!Name Num  Type    Host    BootScript
!---- ---  ----    ----    ----------

ROC0   0  ROC     128.165.86.79
  EBP     2  EB        $NODE       $CODA_BIN/coda_activate  -p /usr1/muondaq/daq/run/ebana_test
  ANAP    3  ANA           $NODE       $CODA_BIN/coda_activate  -p
/usr1/muondaq/daq/run/ebana_test
  EBE     2  EB        $NODE       $CODA_BIN/coda_activate  -p /usr1/muondaq/daq/run/ebana_elect
  ANAE    3  ANA           $NODE       $CODA_BIN/coda_activate  -p
/usr1/muondaq/daq/run/ebana_elect
  EBW     2  EB        $NODE       $CODA_BIN/coda_activate  -p /usr1/muondaq/daq/run/ebana_wang
  ANAW    3  ANA           $NODE       $CODA_BIN/coda_activate  -p
/usr1/muondaq/daq/run/ebana_wang
```

Appendix 8: An example of physics.config

```
!+
! File:-
!    physics.config
!
```

```
! Description:-
!    Physics RunType Configuration File
!
ROC0   /usr1/muondaq/daq/run/second.o
  EBP
  ANAP    /usr1/muondaq/daq/run/ebana_test.log
```

Appendix 9: An example of rcRunTypes

```
!+
! File:-
!    rcRunTypes
!
! Description:-
!    RunType File. Enumerates the available run types and their numbers
!-

Physics                 1
Calibration             2
Test                    3
electronics        4
wangdata           5
```