Using E835 offline code

Diego Bettoni, Claudia Patrignani

This document describes the organization of the offline library and gives some examples on how to use it:

It is meant to be a short "how-to" guide to start writing and running the offline program, it is not a detailed documentation of the program itself.

1 Overview of the program

The program is written mainly in standard FORTRAN77, using only extensions that are supported by both achitectures used in our experiment (irix and osf1).

The variables used by the offline program are documented in detail in the include files where they are defined and can be easily consulted online while writing your code (see section 5).

The main program (PROGRAM TRIM) first reads (from data cards stored into a file assigned to the environmet variable $STEERPAR) the type of data to read (Raw data, DST or MC) and the kind of format choosen to write the events that will be selected by the user in his routine.

According to the read and write type the program opens the data file(s) defined by the environment variable(s) $INPUTDATA ($INPUTDATA2 etc.) and $OUTPUTDATA ($OUTPUTDAT2 etc..), then makes a call to the user-written routines to perform inizialization tasks for the specific analysis.

The event-loop routine (TRIMAIN for real data or MCMAIN for montecarlo event generation) read three more sets of data cards from the files defined by the following environment variables:

- $TRIMPAR: parameters used in the single data-file analysis[1]
- $GEOMPAR: parameters used to modify geometry (or most of the parameters loaded from database) if the user wants to supersede them with his/her own preferred values.
- $KINEMPAR: the center of mass energy used to evaluate kinematical variables such as β_{CM}, γ_{CM} etc...

For each event the reconstruction is performed by steps identified by the "execution level":

1: Get and unpack events (real data) or generate event (MC)

2: Decode detectors (real data) or follow tracks through detectors (MC)

4: Fill hit maps (both real data and MC)

7: Clusterize (pattern recognition at the single detector volume level)

10: Charged track pattern recognition

20: Charged track fitting

30: Charged and neutral tracks association

[1] In a multi-file analysis **$STEERPAR** refers to parameters common to all files processed, while **$TRIMPAR** refers to parameters specific of any single file

35: Vertex fitting (no constrained track re-fitting)

40: Evaluate standard kinematical variables

45: Electron weight or analogous tests on tracks (yet to be decided, of course!)

The list of detector to be used in the reconstruction can be controlled by "activating" or "deactivating" them through data cards.

After steps 1, 4, 10, 20, 40 and 45 the control of the program is given to the user-written filtering routines (`USER_FILTn`) which can decide whether to reject the event or pass it to the subsequent steps .

When the reconstruction level reaches the value specified by the user, control is given to the user-written analysis routine (`USER_EVENT`) which can tag the event for writing in the format chosen at the beginning of the program.

After processing the required number of events (or when the end of file is reached) control is again given to the user-written routines to perform any "end of run" task, such as closing ntuples-files or printing out statistics.

Theoretically the program should be able to handle multiple input files, but this feature has not yet been tested: use it at your own risk!

2 USER_FILTn and USER_EVENT routines

The filtering routines `USER_FILTn` and the analysis routine `USER_EVENT` have two arguments: `IOPT` and `EVWORD`.

`IOPT` is passed by the calling routine and assumes the values:

-1 when called at the beginning of the program

0 when called in the event loop

1 when called at the end of the program

`EVWORD` stores the return value of the user-routine

1 event rejected

0 event accepted

A sample routine can be found in `$E835_DIR/example/user_event.f` whose listing is given in appendix A.

3 Selecting the appropriate data cards

Data cards are read by the FFREAD package (CERN library) which assigns a key to all input variables in order to read their values without requiring a specified format.

The list of keys that can be used in data cards are all defined in the routine defkey.FOR where you can find the correspondence between keys and program variables.

All keys can be used either in STEERPAR or in TRIMPAR. The exceptions are:

RDTY: must be in STEERPAR

WTTY: only in STEERPAR

CMAS: must be in KINEMPAR,

DETE: only in TRIMPAR,

Keys assigned to input variables superseding values strored into the Database can only be in GEOMPAR

Keys assigned to variables defining the MC reaction generated can only be in STEERPAR

Examples of the data cards files needed to run the offline can be found in $E835_DIR/example and are listed in appendix C.

Other keys can be defined by the user by the USKEY routine. In most cases it will be sufficient to add the data cards for the new keys to one of the data card files.

4 How to run your program

4.1 How to define the appropriate environment variables

In order to use the offline you need to set a number of environment variables. For the moment e835code is not yet defined as a product, therefore you have to include in your .cshrc the following line:

```
% source /code/e835code/off96/uti/setup_e835
```

Some useful aliases that can be used to read/search the offline code are also defined (see section 5)

As for now the following products are needed to run the offline:

- cern : `% setup cern`
- daft : `% setup daft`
- ftt : `% setup ftt`
- *msql :* `% setup msql` *(not yet)*

All this products are setup by default by `setup_e835`.

4.2 Sample scripts to link and run the offline

In the area `$E835_DIR/example` you can find two sample scripts that can be used to compile (`trim_link`) and run (`trim_run`) the program on either architectures. Their listing is given in appendix B Just copy it to your area and make the appropriate changes.

5 Looking into the source code

The source code can be found in the directory tree `$E835_DIR/src/..` Each subdirectory contains source files for related subroutines (decoding, printout, clusterization etc.)

Each routine is written into a separate file whose name is *routine_name*`.FOR`. We have attempted to put detailed comments inside the source code itself which should constitute a reasonable documentation for the routine.

The variables in the offline common blocks are all defined in include files with name *name*`.cde` in the area `$E835INC`. Detailed comments in these files describe the contents of each variable.

The script `setup_e835` defines aliases to help people browsing source code. Those aliases are based on the `find` command (see man-pages) and can provide some sort of on-line documentation.

We give some examples to show their possible use:

```
% Ewhat trackpar
```

will type on your screen the include file where the variable `trackpar` is defined (and documented) using `more`.

```
% Emore trimain.FOR
```

will type on your screen the routine `trimain.FOR` using `more`

```
% Efind 'prt*.FOR'
```

will type on your screen the full name of all routines whose name matches the string `prtr*.FOR`

```
% Efind '*.FOR' -exec grep -iq  "INCLUDE 'chlines.cde'" {} \; -ok more {} \;
```

For any routine which includes the file `chlines.cde`, it will ask you if you want to type it on screen using `more`

```
% Efind user_filt4.FOR -ok cp {} . \;
```

will copy (requiring a confirm) the offline routine `user_filt4.FOR` (a template) to your current working directory.

A Sample analysis routine

```
      SUBROUTINE USER_EVENT(IOPT,EVWORD)
C---
C---  Event analysis
C---
      IMPLICIT NONE
      SAVE
      INCLUDE 'pardef.cde'
      INCLUDE 'hits.cde'
      INCLUDE 'clust.cde'
      INCLUDE 'chlines.cde'
C--- Local variables
      INTEGER IOPT,EVWORD istat,icycle
```

```
      CHARACTER*80 filnam

      INTEGER I
C--- EXECUTABLE CODE
      IF(IOPT) 1, 10, 100
   1  CONTINUE
C====================================================== Initialization: IOPT = -1
C--- Get file name from environment variable
      call getenv('OUTPAW',filnam)
      if(filnam(1:4).eq.'    ') then
        filnam='outpaw.dat'
      endif
C--- Open the file
      CALL HROPEN(34,'MYNTP',filnam, 'N',1024,ISTAT)
C--- Book histograms and Ntuples
* HOD1
      CALL HBOOK1(10*IVHOD1 + 1,'H1 hit multiplicity',8,-0.5,7.5,0.)
      CALL HBOOK1(10*IVHOD1 + 2,'H1 hit map',8,.5,8.5,0.)

      RETURN
  10  CONTINUE
C=========================================================== Event loop: IOPT = 0
      EVWORD = 1
C--- Reject events with more than 4 H1
      IF(NHH1AD.GT.4)         RETURN
C--- Reject events with  less than 2 CCAL clusters
      IF(NCLUST(IVCCAL).LT.2)   RETURN
C--- Event passed all the criteria
      EVWORD = 0

C--- Fill histograms
* HOD1
      Call Hfill(10*IVHOD1+1,Float(NHH1AD),0.,1.)
      DO I = 1,8
        IF(IHH1TD(I).GT.0) CALL HFILL(10*IVHOD1+2,float(I),0.,1.)
      END DO
      RETURN
```

```
  100 CONTINUE
C================================================================= End: IOPT = 1
      CALL HROUT(0,ICYCLE,' ')
      CALL HREND('MYNTP')
      RETURN
      END
```

B Sample script to link and run your program

B.1 trim_link

```
#!/bin/sh
# ====================================================================
# Create the executable file linked to the offline (debug library)
# ====================================================================
case $OSTYPE in
   irix)
#====================================================================
#  Options for IRIX
#
    F=""
    DOPT="-g -mips2 -C"
    FOPT="-mips2"
    A="-Wf,"
    LDS="-lds"
    ;;
   osf1)
#=====================================================================
#  Options for AXP
#
    F="-f"
    DOPT="-g -C"
    FOPT=""
```

```
    A=""
    LDS=""
    ;;
    *)
    echo "unsupported"
    ;;
esac
#=============================================================================
# define the area for offline and other include files
#
INC="$A-I$E835INC $A-I$DAFT_DIR/include $A-I$FTT_DIR/include \
  $A-I$MSQL835_DIR/include"
#============================================================================
# define the area for offline and other libraries
#
LIBD="-L$E835LIB -L$DAFT_DIR/lib -L$FTT_DIR/lib -L$MSQL835_DIR/lib -L$CRNLIB"
#============================================================================
# define libraries to which link the offline (offline debug library)
#
LIBS="-lE835-dbg -ldaft_dbg -lftt \
 -lphtools -lmathlib -lkernlib -lpacklib -lpawlib -lm $LDS"
#=========================================
# Main program extracted from library (needed for IRIX!!!)
#
 ar -x $E835LIB/libE835-dbg.a program_trim.o
 echo " program_trim.o exctracted from library"
#=============================================================================
# Routine user_event.f compiled with DEBUG
#
 f77 -c $DOPT $INC user_event.f
echo " user_event.f compiled with DEBUG"
#=============================================================================
# Link those routines with the debug library to create the offline
# program trim.x
#
f77 $DOPT program_trim.o user_event.o $LIBD $LIBS -o trim.x
echo " program trim.x linked"
#=============================================================================
# cleanup obj
```

```
test -s program_trim.o && rm program_trim.o
test -s user_event.o && rm user_event.o
```

B.2 trim_run

```
#!/bin/csh
# ======================================================================
# run the offline
# ======================================================================
#
# ----> Assign input and output data file
#
setenv INPUTDATA  myfile.dat
#
# ---> Assign files containing data cards
#
setenv STEERPAR  steerpar_raw.dat
setenv TRIMPAR   trimpar.dat
setenv KINEMPAR  kinempar.dat
setenv GEOMPAR   geompar.dat
#
# This works just on OSF1 (IRIX will produce fort.8)
#
setenv FORT8   dump.dat
#
# Now run (cross your fingers!)
#
trim.x > myfile.log
```

C Sample data cards files

C.1 $STEERPAR (raw data)

```
LIST
C               put here all cards general purpose
RDTY 1                                               READ TYPE
EXEC 4                                               EXECUTION LEVEL
```

```
WTTY 0                                          WRITE TYPE
C============================================== Dump 5 events for steps 1-10
GRPL 10*5
C======================= H1 and H2 cluster formed if more than 15 ADC counts
CSH1 15.
CSH2 15.
C==================== error handling parameters: ignore trigger type unknown
ISEV 131=0
END
```

C.2 $STEERPAR (MC)

```
LIST
RDTY 0                                           INPUT TYPE
EXEC 20                                          EXECUTION LEVEL
WTTY 0
C----------------------------------------- Dump 10 events at steps 2,7,10,20
GRPL 20*0 2=10 7=10 10=10 20=10
C
C-------- MONTECARLO CARDS
C
NPMC 7                  ! number of particles involved
REAC 1
C
C------ Description of particles involved
C------                 CTAU  -1. means stable
C------                 KIND 0,1,2 No interaction, EM shower, Hadron
C------ Name mass/GEV Wid/MEV Charg ctau/cm Kind
C
PART1  'CMAS'  2.990    0.2    0     0.      0
PART2  'PHI '  1.020    4.0    0     0.      0
PART3  'PHI '  1.020    4.0    0     0.      0
PART4  'K+  '  0.4936   0.     1    -1.      2
PART5  'K-  '  0.4936   0.    -1    -1.      2
PART6  'K+  '  0.4936   0.     1    -1.      2
PART7  'K-  '  0.4936   0.    -1    -1.      2
C
```

```
C------------- Reactions are defined step by step starting from CMAS
C-------        STEPn: PARTi -> PARTa + PARTb +....PARTx
C
STEP1 1 2 3
STEP2 2 4 5
STEP3 3 6 7
C------  MAX Weight : see GENBOD manual
MWEI1 0.                  ! set  maxweig to zero to have its distribution
MWEI2 0.                  ! or for two body phase space that needs no weight
END
```

C.3 $TRIMPAR

```
LIST
C================================= ACTIVATE NO DETECTORS EXCEPT THOSE
C================================= EXPLICITLY LISTED
DETE 'NONE' 'CNTR' 'STR1' 'STR2' 'FCAL' 'CCAL' 'SILI' 'FIBE' 'VETO'
     'FCAS' 'CCAS' 'TRIG'
C================================== ANALYZE 900 EVENTS SKIPPING 0
EVEN 0 900
END
```

C.4 $KINEMPAR

```
LIST
CMAS 3.556
END

LIST
C================================= ACTIVATE NO DETECTORS EXCEPT THOSE
C================================= EXPLICITLY LISTED
DETE 'NONE' 'CNTR' 'STR1' 'STR2' 'FCAL' 'CCAL' 'SILI' 'FIBE' 'VETO'
     'FCAS' 'CCAS' 'TRIG'
C================================== ANALYZE 900 EVENTS SKIPPING 0
EVEN 0 900
END
```

C.5 $GEOMPAR

```
LIST
END
```

D Installation

To install the offline to a different machine you can use either `rcp` copy or `ftp`. Be sure you have enough space. The following refers to an AXP machine. If you have an SGI replace *fn835i* with *fn835c* and *osf1* with *irix*

1. If you plan to use `rcp` edit appropriately the `.rhosts` file on BOTH your accounts

2. create the directory which has to hold the offline directory tree

```
% mkdir  /myarea/e835code
% mkdir  /myarea/e835code/lib
% mkdir  /myarea/e835code/src
% mkdir  /myarea/e835code/uti
% mkdir  /myarea/e835code/include
% mkdir  /myarea/e835code/examples
```

3. create a subdirectory for each of the needed products (not `cern` of course): `msql`, `daft` and `ftt`

```
% mkdir  /myarea/e835code/daft
% mkdir  /myarea/e835code/daft/include
% mkdir  /myarea/e835code/daft/lib
% mkdir  /myarea/e835code/ftt
% mkdir  /myarea/e835code/ftt/include
% mkdir  /myarea/e835code/ftt/lib
% mkdir  /myarea/e835code/msql
% mkdir  /myarea/e835code/msql/include
% mkdir  /myarea/e835code/msql/lib
```

4. copy the directory there the setups and utility scripts are `rcp`

```
rcp -rp fn835i.fnal.gov:'$E835_DIR/uti/'      /myarea/e835code/uti/
```

5. edit the file /myarea/e835code/uti/setup_e835 replacing each product setup with

```
##!/bin/tcsh
#
# Define environment variables
# E835
setenv E835_DIR     /myarea/e835code
# FTT
setenv FTT_DIR      /myarea/e835code/ftt
# DAFT
setenv DAFT_DIR     /myarea/e835code/daft
# MSQL
setenv MSQL835_DIR  /myarea/e835code/msql
# CRNLIB
setenv CRNLIB       /where/it/is/on/your/system
#
   :
   :
#
```

6. setup the offline environment variables

```
 % source /myarea/e835code/uti/setup_e835
```

7. put in your .cshrc the command

   source /myarea/e835code/uti/setup_e835

8. edit the file $E835UTI/e835copy replacing (if necessary) the remote node from which you want to copy the code and libraries

```
#/bin/tcsh
setenv remnode fn835i.fnal.gov
#
rcp -rp $remnode:'$E835LIB/'          $E835LIB/
```

```
rcp -rp $remnode:'$E835_DIR/src/'      $E835_DIR/src/
rcp -rp $remnode:'$E835_DIR/include/'  $E835_DIR/include/
rcp -rp $remnode:'$E835_DIR/examples/' $E835_DIR/examples/
#
rcp -rp $remnode:'$DAFT_DIR/lib/'         $DAFT_DIR/lib/
rcp -rp $remnode:'$DAFT_DIR/include/'     $DAFT_DIR/include/
rcp -rp $remnode:'$FTT_DIR/lib/'          $FTT_DIR/lib/
rcp -rp $remnode:'$FTT_DIR/include/'      $FTT_DIR/include/
rcp -rp $remnode:'$MSQL835_DIR/include/'  $MSQL835_DIR/include/
rcp -rp $remnode:'$MSQL835_DIR/lib/'      $MSQL835_DIR/lib/
```

9. copy the content of offline area and of the needed products directories

   ```
   % source $E835UTI/e835copy
   ```

10. set the appropriate protections to grant read to group (if necessary) e.g:

    ```
    Efind '*' -exec chmod g+r {} \;
    ```

11. execute the following commands to recreate the libraries (needed only if you want to use the debugger)

    ```
    % cd $E835UTI
    % % rm -i *.log ; rm -i $E835LIB/*.log ; rm -i $E835_LIB/lock_lib
    % newlib > lib-osf1.log &
    ```

NOTE: when copying whole directory trees you may be getting much more than you need (such as old versions of the libraries). Cleanup after copying each directory or `ftp` only the files you really need if you don't have enough space to copy all at once.