CDF/MIN-BIAS/DOC/CDFR/3251

NEW STRUCTURE OF MINIMUM BIAS MICRO-DST

V. Bolognesi and F. Rimondi

Abstract

In this note we present a work on a new type of micro-DST. The main objective is to allow a quick and easy access to a large amount of data recorded at CDF from the Minimum Bias trigger, but this method can be applied also to other data sets.

1 Introduction

The structure of the micro-DST we developed is completely independent on YBOS and Analysis Control. These packages were used just in unpacking banks from the original CDF DSTs. Once this micro-DST has been created, data analysis proceeds independently. The structure is set by ZEBRA data memory manager and it is specified by the HBOOK package. This package, its graphic interface HPLOT, PAW system and other utilities, are available on many computer installations through the world and have became widely used standard packages. Specifically we use the Column-Wise-Ntuple (CWN) mechanism, that allows direct access and easy compression of the data. With our bank selection and packing strategy, we obtain a data reduction according an average factor 2 % relative to standard CDF micro-DSTs.

Several reasons have determined this choice.

- The most important feature of the CWN storage mechanism is that the unused trailing array elements aren't stored if the array depends on an index variable: only as many array elements are stored as specified by the index variable. One can think of CWN as a large matrix of variable dimensions, where each row corresponds to one event and the number of the columns depends on the amount of informations to be stored for each event. It requires lesser memory than storing a fixed length Row Wise Ntuple at the largest foreseen dimension.

- The compression of the data is done also by distinguishing between data types: real *4 and *8, integer *4 and *8, unsigned integer numbers, booleans and character strings.

- The CWN structure allows 2 or more tracks direct combinations, relevant for our correlations studies.
- The reduced data size allows a more convenient file transfer from Fermilab to home institutions.
- The analysis stage can be greatly speeded because one has the possibility of retrieving the information from those columns needed in his job and make the relevant cuts by writing a normal FORTRAN program, to be run in batch mode, without the inconvenient limitations of PAW COMIS FORTRAN. In the same FORTRAN program one can fill histograms, make their information available in local variables for further elaboration and keep track of general counters in looping over the input ntuple files. All these possibilities are very useful and important in our studies of various correlations among tracks in the MINIMUM BIAS events at CDF.
- In general, reading a complete CWN takes longer than a RWN, due to type checking and compression mechanisms and because the data are not stored sequentially on disk. For this reason, one can obtain within a reasonable time only simple histograms and impose simple cuts using PAW commands in interactive mode. The speed advantage is instead achieved running the normal FORTRAN program mentioned above.

2 CWN creation in Analysis Control environment

The ZEBRA memory allocation is usually defined in the application program with the common block Common/PAWC/ipaw(nwords). In the Build Job frame one has just to indicate the memory size, by specifying

```
set size/hbk4=6500000
set utilities/hbk4=on
```

This is the maximum allowed size.

In the Analysis Control stage, while running the program, one has to use:

```
hist open/FILE=myntu/max_nrec=65000/reclen=4095
histogram on
begin
histogram write
histogram close
```

The first command is equivalent to call HROPEN, with overriding the default number of records for the created file containing the CWN (=32k, maximum 65k) and the default record lenght in machine words (=1024, maximum 4095). Since one calls HROPEN not only to write but also to read the file containing the CWN, one has to specify the same number of records and the same record lenght while opening the CWN in PAW or reading it in a batch job, because they are different from the defaults.

The CWN booking is performed in two stages: ntuple identifier and title are declared by calling HBNT and then the variables to be stored are described in one or more calls to HBNAME or HBNAMC for character variables. Among the arguments of these subroutines there are the name of the block in CWN containing the data and a character string corresponding to their list. There must be a direct correspondence between this list and the common declaration of the variables. The maximum dimensions of the variable arrays have to be specified in the same common, and not in a DIMENSION statement. The maximum total number of columns per event cannot exceed 5000. After assigning the values to the relevant variables on an event by event basis, each block is filled by calling HFNTB. The division into separated blocks allows one to fill each block in different routines (for example data from different banks) and to establish different lenghts for the variable arrays (for example looping on all CTC charged tracks or on all fired calorimeter towers).

The entuple is stored by default in directory //MYMOD. MYMOD is the name of the Analysis Control module where the CWN booking is performed.

The input files are specified in the INPUT FILES command from Analysis Control. They can be already on disk, or read from tape, or copied from tape to disk and then processed. The latter procedure is preferred in general, in order to avoid the frequent failures while processing data directely from tape, due to missing EOF, which are instead introduced by default in copying them to disk.

3 Structure of the ntuples

The content of the ntuple is organized in 5 named blocks:

1. in the first block, named MBRUNS, general informations about current run and event are stored;

2. the named block TRIGGER contains the levels and the names of all passed triggers obtained from TAGC/TAGZ banks and stored into two arrays of variable dimension, specified by the number NTRIG for each event (NTRIG=total number of passed triggers in the current event, maximum 100);

3. the named block VTXDAT contains the z-coordinate and the class of all vertices obtained from VTVZ bank and stored into two arrays of variable dimension, specified by the number NPVCWN for each event (NPVCWN=total number of vertices in the current event, maximum 20);

4. the named block CTCDAT contains some track parameters, their errors, numbers of hits in VTX modules and CTC superlayers and also various CTC and SVX track selection flags obtained from QTRK/CTCS/SVXS banks. They are stored into the corresponding arrays of variable dimension, specified by the number QNTRACK for each event (QNTRACK=total number of tridimensional tracks in the current event, maximum 150);

5. the named block CALDAT contains the theta pointer, the phi pointer, the electromagnetic, hadronic and crack energy values for calorimetric towers of type from 0 to 8, with electromagnetic energy greater than 50 MeV and/or hadronic energy greater than 150 MeV, from QTOW/TOWE banks. They are stored into 5 arrays of variable dimension, specified by the number NTOWS for each event (NTOWS=total number of towers with electromagnetic energy greater than 50 MeV and/or hadronic energy greater than 150 MeV, in the current event, maximum 400).

The complete printout of the CWN header looks as follows:

```
******************************************************************
* Ntuple ID = 11     Entries = 773       minbias
******************************************************************
* Var numb * Type * Packing *    Range     *  Block   *  Name    *
******************************************************************
*      1   * I*4  *         *              * MBRUNS   * cwnrun
*      2   * I*4  *         *              * MBRUNS   * cwnev
*      3   * I*4  *         *              * MBRUNS   * qtrcut
-------------------------------------------------------------------------
*      1   * I*4  *         * [0,100]      * TRIGGER  * ntrig
*      2   * I*4  *         *              * TRIGGER  * trlvl(ntrig)
*      3   * C*32 *         *              * TRIGGER  * trnam(ntrig)
-------------------------------------------------------------------------
*      1   * I*4  *         * [0,20]       * VTXDAT   * npvcwn
*      2   * R*4  *         *              * VTXDAT   * zvcwn(npvcwn)
*      3   * R*4  *         *              * VTXDAT   * clcwn(npvcwn)
-------------------------------------------------------------------------
*      1   * I*4  *         * [0,150]      * CTCDAT   * qntrack
*      2   * I*4  *         *              * CTCDAT   * ctcflag(qntrack)
*      3   * I*4  *         *              * CTCDAT   * svxflag(qntrack)
*      4   * R*4  *         *              * CTCDAT   * qpt(qntrack)
*      5   * R*4  *         *              * CTCDAT   * ptsg(qntrack)
```

```
*      6   * R*4  *         *             * CTCDAT   * qpz(qntrack)
*      7   * R*4  *         *             * CTCDAT   * pzsg(qntrack)
*      8   * R*4  *         *             * CTCDAT   * dedxctc(qntrack)
*      9   * R*4  *         *             * CTCDAT   * dedxsvx(qntrack)
*     10   * I*4  *         *             * CTCDAT   * dedxnh(qntrack)
*     11   * R*4  *         *             * CTCDAT   * qz0(qntrack)
*     12   * R*4  *         *             * CTCDAT   * z0sg(qntrack)
*     13   * R*4  *         *             * CTCDAT   * qd0(qntrack)
*     14   * R*4  *         *             * CTCDAT   * d0sg(qntrack)
*     15   * R*4  *         *             * CTCDAT   * qphi0(qntrack)
*     16   * R*4  *         *             * CTCDAT   * phi0sg(qntrack)
*     17   * R*4  *         *             * CTCDAT   * qfchi2(qntrack)
*     18   * U*4  *         *             * CTCDAT   * qhctc(qntrack)
*     19   * I*4  *         *             * CTCDAT   * qvtxh1(qntrack)
*     20   * I*4  *         *             * CTCDAT   * qvtxh2(qntrack)
*     21   * I*4  *         *             * CTCDAT   * ptocal(qntrack)
--------------------------------------------------------------------------
*      1   * I*4  *         * [0,400]     * CALDAT   * ntows
*      2   * I*4  *         *             * CALDAT   * pcell(ntows)
*      3   * R*4  *         *             * CALDAT   * eem(ntows)
*      4   * R*4  *         *             * CALDAT   * eha(ntows)
*      5   * R*4  *         *             * CALDAT   * ecrk(ntows)
******************************************************************
*  Block   *  Entries  * Unpacked * Packed *   Packing Factor    *
******************************************************************
* MBRUNS   *  773      * 12       * 12     *       1.000         *
* TRIGGER  *  773      * 3604     * Var.   *    Variable         *
* VTXDAT   *  773      * 164      * Var.   *    Variable         *
* CTCDAT   *  773      * 12004    * Var.   *    Variable         *
* CALDAT   *  773      * 6404     * Var.   *    Variable         *
* Total    *    ---    * 22188    * Var.   *    Variable         *
******************************************************************
* Blocks = 5            Variables = 35      Max. Columns = 4847  *
******************************************************************
```

A more detailed description of each named block follows.

MBRUNS

1. number of the current run
2. number of the current event
3. cut flag (=0 if the number of tracks is lesser than 150, =1 if that number exceeded 150 and the quality cut from CTCSEL has been already applied in order to reduce it)

TRIGGER

1. NTRIG number of passed triggers in the current event (max.100)
2. level of passed triggers (array of dimension NTRIG)
3. name of passed triggers (character string*32 array of dimension NTRIG)

VTXDAT

1. NPVCWN number of vertexes in the current event (max.20)
2. z coordinate for each vertex (array of dimension NPVCWN)
3. class for each vertex (array of dimension NPVCWN)

CTCDAT

1. QNTRACK number of tracks in the current event (max.150)
2. flag from routine CTCSEL for each track (=0 if ok)
3. flag from routine SVXSEL for each track (=0 if ok)
4. transverse momentum of each track, the sign of this word corresponds to the particle charge [1]
5. sigma on transverse momentum of each track[1]
6. z component of momentum of each track[1]
7. sigma on z component of momentum of each track[1]
8. average energy loss in CTC (from DEDX bank)
9. average energy loss in SVX (from DEDX bank)
10. packed numbers of hits used in dE/dx calculation
11. z_0 value for each track[1]
12. sigma on z_0 for each track[1]
13. d_0 value for each track[1]
14. sigma on d_0 for each track[1]
15. ϕ_0 value for each track[1]
16. sigma on ϕ_0 for each track[1]
17. χ^2 per degree of freedom[1]
18. packed numbers of hits in each CTC superlayer
19. VTX hits count for even modules per track
20. VTX hits count for odd modules per track
21. packed eta and phi pointers from track to the calorimeter tower, if any

CALDAT

1. number of calorimetric towers with electromagnetic energy greater than 50 MeV and/or hadronic energy greater than 150 MeV (max.400)[2]

---

[1] The current value of the variable is updated from the QTRK content, according to the status of the track quality selection routines CTCSEL and SVXSEL: if CTCSEL is ok, then the track paramaters from CTCS bank are stored and if SVXSEL is also ok, then the results of the combined fit CTC+SVX are stored.

[2] If this limit is exceeded, the ntuple isn't filled for the current event and a complete dump of the tracking, passed triggers, vertices, calorimetry is available for further investigation.

2. packed eta and phi pointers for each tower from TOWE bank (the eta pointer has been limited to the range [19,66], corresponding to tower types from 0 to 8
3. electromagnetic energy for each tower from TOWE bank
4. hadronic energy for each tower from TOWE bank
5. energy in cracks for each tower from TOWE bank

4 Reading CWN

Two ways of reading ntuples are possible:

1. PAW selection function,
2. user program for batch usage.

For direct PAW usage the revelant function skeleton is obtained by following the example procedure:

```
ntu/chain mbs    - list of the relevant files -
cd //mbs
cd tcalcwn
ntu/uwfun 11 myfunc.for
```

Users should pay attention to the variable type (integer, real or character) and take into account that in PAW functions it isn't possible to use GOTO statement and bit functions. Only a few routines from FORTRAN and KERNLIB libraries, HBOOK, ZEBRA, HPLOT, KUIP and HIGZ packages are available, as listed in the PAW manual.

For the FORTRAN user program, the revelant function skeleton is written with a call HUWFUN after booking ntuple. This subroutine has to be called in order to read each ntuple, its structure is standard and requires no changes. In the following this structure is shown for the CWN we use, included in a complete example of a program opening many files containing ntuples (these files are listed in NTULIS.DAT, opened and read once). Histograms of interest are booked and, after the filling, they are written to the output file MBSNTU.HIS. The filling is done in subroutine MYSUB, where the content of the relevant columns (see array VAR) is put into commons for all events in the ntuple (see loop over the ntuple entries). Cuts on the whole events and on each track can be applied at this stage. In the following example some cuts on events and on tracks are applied.

Events are accepted if they satisfy the following conditions:

1. passed level 2 trigger YMON
2. no tracks with p_T greater than 100 GeV/c
3. the main primary vertex z coordinate in the range [-28 cm,28 cm]
4. no vertex closer than 60 cm to the main one (no multiple interaction)

Tracks are accepted if they have:

1. $|\eta| < 1$
2. $p_T > 0.4$ GeV/c
3. $|impact\,parameter| < 0.5$ cm
4. z matching between track and main primary vertex < 5 cm
5. CTCSEL track quality cuts satisfied

```
=============================
            program myprog
=============================

        integer nwpawc,lrecl
        parameter(nwpawc=5000000)
        parameter(lrecl=1024)
        common /pawc/ ipaw(nwpawc)
        character*80 ntufile(200)

        integer numev,numg,ntus,intu

        call hlimit(nwpawc)
c       counters initialization
        numev=0
        numg=0

        call hbook1(101,'phi pointer CTC to CAL',73,0.,73.,0.)
        call hbook1(102,'eta pointer CTC to CAL',85,1.,85.,0.)
        call hbook1(103,'CAL phi pointer',73,0.,73.,0.)
        call hbook1(104,'CAL eta pointer',85,1.,85.,0.)
        call hbook1(199,'phi distribution',90,0.,360.,0.)
        call hbook1(200,'eta distribution',100,-1.,1.,0.)
        call hbook1(201,'CTC hits in superlayer 1',13,0.,13.,0.)
        call hbook1(202,'CTC hits in superlayer 2',7,0.,7.,0.)
        call hbook1(203,'CTC hits in superlayer 3',13,0.,13.,0.)
        call hbook1(204,'CTC hits in superlayer 4',7,0.,7.,0.)
        call hbook1(205,'CTC hits in superlayer 5',13,0.,13.,0.)
        call hbook1(206,'CTC hits in superlayer 6',7,0.,7.,0.)
        call hbook1(207,'CTC hits in superlayer 7',13,0.,13.,0.)
```

```
      call hbook1(208,'CTC hits in superlayer 8',7,0.,7.,0.)
      call hbook1(209,'CTC hits in superlayer 9',13,0.,13.,0.)
      call hbook1(210,'CTC hits in DEDX',90,0.,90.,0.)
      call hbook1(211,'SVX hits in DEDX',90,0.,90.,0.)
      call hbook1(300,'track charge',5,-2.,3.,0.)
      call hbook1(301,'track pT distribution',100,0.,4.,0.)



      intu=0
      open(33,status='old')
 99   continue
      intu=intu+1
      if(intu.gt.200)goto 299
      read(33,35,end=199,err=299)ntufile(intu)
 35   format(3x,a80)
      print *,intu,ntufile(intu)
      goto 99

 199  continue
      ntus=intu-1

      do 10 i=1,ntus
      print *,i,ntufile(i)
      inpfile=ntufile(i)
      call hropen(1,'mbsan',inpfile,' ',4095,istat)
      call hcdir('tcalcwn',' ')
      call hldir(' ','A')
      call hrin(11,9999,0)
      call mysub
      call hrend('mbsan')
      print *,' good events for file', numg
  10   continue

       call histdo
       call hrput(0,'file.his','n')

       stop

 299  continue
      print *,'error in reading ntu file name ',intu
      stop
      end
c^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


     SUBROUTINE MYSUB
*********************************************************
*                                                       *
* This file was generated by HUWFUN.                    *
```

```
*                                                     *
*******************************************************
*
*     Ntuple Id:      11
*     Ntuple Title:   minbias
*     Creation:       30/05/95 09.00.32
*
*******************************************************

       INTEGER cwnrun,cwnev,qtrcut
       COMMON /MBRUNS/ cwnrun,cwnev,qtrcut

      INTEGER ntrig,trlvl
      CHARACTER trnam*32
      COMMON /TRIGGER/ ntrig,trlvl(100)
      COMMON /TRIGGER1/ trnam(100)

      REAL zvcwn,clcwn
      INTEGER npvcwn
      COMMON /VTXDAT/ npvcwn,zvcwn(20),clcwn(20)

      REAL qpt,ptsg,qpz,pzsg,dedxctc,dedxsvx,qz0,z0sg,qd0,d0sg,qphi0
     + ,phi0sg,qfchi2
      INTEGER qntrack,ctcflag,svxflag,dedxnh,qhctc,qvtxh1,qvtxh2,ptocal
      COMMON /CTCDAT/ qntrack,ctcflag(150),svxflag(150),qpt(150)
     + ,ptsg(150),qpz(150),pzsg(150),dedxctc(150),dedxsvx(150)
     + ,dedxnh(150),qz0(150),z0sg(150),qd0(150),d0sg(150),qphi0(150)
     + ,phi0sg(150),qfchi2(150),qhctc(150),qvtxh1(150),qvtxh2(150)
     + ,ptocal(150)

      REAL eem,eha,ecrk
      INTEGER ntows,pcell
      COMMON /CALDAT/ ntows,pcell(400),eem(400),eha(400),ecrk(400)


        integer numev,numg,nloop,ierr,mult,multint,acc
        integer ind(150),trigflag,hitctc(9),isl,hind,dhsvx,dhctc
        integer icode,phip,etap
        character*32 trigname
        common /numbers/ numev,numg
        character*8 var(35)
        real vertz, etatr(150),pt,trch

        INTEGER    MASK3
        PARAMETER (MASK3 =       '7'X)
        INTEGER    MASK4
        PARAMETER (MASK4 =       'F'X)
        INTEGER    MASK7
        PARAMETER (MASK7 =      '7F'X)
*
*
```

```
      CALL HBNAME(11,' ',0,'$CLEAR')
      CALL HBNAME(11,'MBRUNS',cwnrun,'$SET')
*
      CALL HBNAME(11,'TRIGGER',ntrig,'$SET')
      CALL HBNAMC(11,'TRIGGER',trnam,'$SET')

      CALL HBNAME(11,'VTXDAT',npvcwn,'$SET')
*
      CALL HBNAME(11,'CTCDAT',qntrack,'$SET')
*
      CALL HBNAME(11,'CALDAT',ntows,'$SET')
*
*
*--   Enter user code here
*
c   establish the variables of interest
        var(1)='qtrcut'
        var(2)='ntrig'
        var(3)='trlvl'
        var(4)='trnam'
        var(5)='npvcwn'
        var(6)='zvcwn'
        var(7)='qntrack'
        var(8)='ctcflag'
        var(9)='qpt'
        var(10)='qpz'
        var(11)='qd0'
        var(12)='qz0'
        var(13)='qhctc'
        var(14)='dedxnh'
        var(15)='ptocal'
        var(16)='ntows'
        var(17)='pcell'
        var(18)='eem'
        var(19)='eha'
        var(20)='qphi0'

c   get the number of entries in the current ntuple
        call hnoent(11,nloop)
c   total events counter upgrade
        numev=numev+nloop

        call hgntv(11,var,20,1,ierr)

c   loop over all events in the current ntuple
        do 10 irow=1,nloop
        call hgntf(11,irow,ierr)
        if(ierr.ne.0) then
           print *,'error reading row ',irow
           goto 10
        endif
```

```
c   apply standard cuts on events
        ym=0
        l2=0
        trigflag=10
        do 59 i=1,ntrig
        if(trlvl(i).ne.2) goto 59
        trigname=trnam(i)
        if(trigname(1:4).eq.'YMON')ym=1
        if(trigname(1:4).eq.'L2_C')l2=-1
  59    continue
        if(ym.ne.1) goto 10

        vertz=zvcwn(1)
        multint=0
c   count multiple interaction candidates
       do 21 i=2,npvcwn
         if (abs(zvcwn(i)-vertz).lt.60.) multint=multint+1
 21    continue

       if(multint.gt.0) goto 10
       if(abs(vertz).ge.60.) goto 10
       if (qntrack.le.0) goto 10

        acc=1
        do 22 i=1,qntrack
        if(abs(qpt(i)).gt.100.) acc=0
 22     continue

        if(acc.eq.0) goto 10

C     count good events
        numg=numg+1

c   apply standard cuts on tracks
        mult=0
       do 20 i=1,qntrack
        if(abs(qz0(i)-vertz).gt.5.) goto 20
        if(abs(qd0(i)).gt.0.5) goto 20
        pt=abs(qpt(i))
        if(pt.lt.0.4) goto 20
        if(ctcflag(i).ne.0) goto 20

        etatr(i)= qpz(i)/pt
        etatr(i)= LOG(SQRT(1+etatr(i)**2)+etatr(i))
        if(abs(etatr(i)).gt.1.) goto 20

C     count good tracks (multiplicity of the event)
        mult=mult+1
        ind(mult)=i
 20    continue
```

```fortran
      if(mult.eq.0) goto 10


c   loop over good tracks for histogram filling
      do 30 i=1,mult
      it=ind(i)
      call hfill(199,qphi0(it),0.,1.)
      call hfill(200,etatr(it),0.,1.)

C  unpack CTC hits word
      hitctc(1)=iand(qhctc(it),mask4)
      hitctc(3)=iand(ishft(qhctc(it),-4),mask4)
      hitctc(5)=iand(ishft(qhctc(it),-8),mask4)
      hitctc(7)=iand(ishft(qhctc(it),-12),mask4)
      hitctc(9)=iand(ishft(qhctc(it),-16),mask4)
      hitctc(2)=iand(ishft(qhctc(it),-20),mask3)
      hitctc(4)=iand(ishft(qhctc(it),-23),mask3)
      hitctc(6)=iand(ishft(qhctc(it),-26),mask3)
      hitctc(8)=iand(ishft(qhctc(it),-29),mask3)

c   loop over the nine CTC superlayers
      do 31 isl=1,9
        hind=200+isl
        call hfill(hind,float(hitctc(isl)),0.,1.)
 31   continue

C  unpack DEDX hits word
      dhsvx=iand(dedxnh(it),mask7)
      dhctc=iand(ishft(dedxnh(it),-7),mask7)
        call hfill(210,float(dhctc),0.,1.)
        call hfill(211,float(dhsvx),0.,1.)

C  unpack CTC to CAL pointers word
      icode = ptocal(it)
      PHIP  = IAND(icode,MASK7)
      ETAP  = IAND(ISHFT(icode,-7),MASK7)
      call hfill(101,float(phip),0.,1.)
      call hfill(102,float(etap),0.,1.)

c   get the transverse momentum of each track and
        pt=abs(qpt(it))
c   its electrical charge
        trch = sign(1.,qpt(it))

        call hfill(301,pt,0.,1.)
        call hfill(300,trch,0.,1.)

 30   continue
```

```
C  unpack ETA and PHI pointers word for each fired calorimeter tower
        do 40 j=1,ntows
        icode = pcell(j)
        PHIP  = IAND(icode,MASK7)
        ETAP  = IAND(ISHFT(icode,-7),MASK7)
        call hfill(103,float(phip),0.,1.)
        call hfill(104,float(etap),0.,1.)
 40     continue

 10     continue  !  loop over the events in the current ntuple

        print *,' good events ', numg

        return
      END
```

5 Conclusion

Currently the production of the micro-DST, based on CWN structure, is in progress and already 15 tapes have been processed. The files containing the ntuples generated up to now can be found on tape RK5952 (8mm cassette). We hope to complete as soon as possible the micro-DST from run 1A data and to extend it to run 1B data.