CDMS Note 9806003 June 25, 1998

The CDMS Ntuple – Data Sheet

Steve Eichblatt, Fermilab
Dr. Sae Woo Nam, Stanford University
Thushara Perera, Case Western Reserve University

1 Introduction

This note is intended as a byte-by-byte description of the contents of the new CDMS ntuple. This note will not focus on possible uses of the ntuple, or the reasoning behind the choices of what is included and what is omitted.

The most recent copy of this document can be found at
http://fnphyx-www.fnal.gov/experiments/cdms/software/ntuple/

2 Design

Some of the goals that drove the creation of the ntuple file were

- The data contained in the ntuple should allow a user to reject bad events, plot spectra, and do systematic checks of the data quality.
- The ntuple should allow a user to discover problems in the data or the processing, but not neccessarily to solve them.
- The file should contain quantities calculated with the most up to date publicly accepted CDMS tools wherever possible.
- The format must be able to handle the CDMS II data requirements.
- The format must be FLIP/BLIP indifferent.
- The file must be readable accross platforms and data analysis tools.
- The file must be compact.

3 File Description

The ntuple file is contains 3 distinct parts, as shown in figure 2. The first part is the names of the variables stored for each event. The next part is the “housekeeping” section, which contains numbers of interest once per series. The last section is information about each event.

The entries are written to the file event by event, making the format both easy to read and scalable to large data sets.

The parts are stored as three different variables in a binary Matlab data file. The pre-Matlab 5 format was choosen for its simplicity. This file is loadable directly from within all versions of Matlab, easily readable using in custom programs written in C, FORTRAN, Pascal, etc., and relatively compact because of the binary format. The format is also platform architecture to avoid problems such as big endian to little endian.

Each of the three sections are shown in figure 2 with their Matlab header described below.

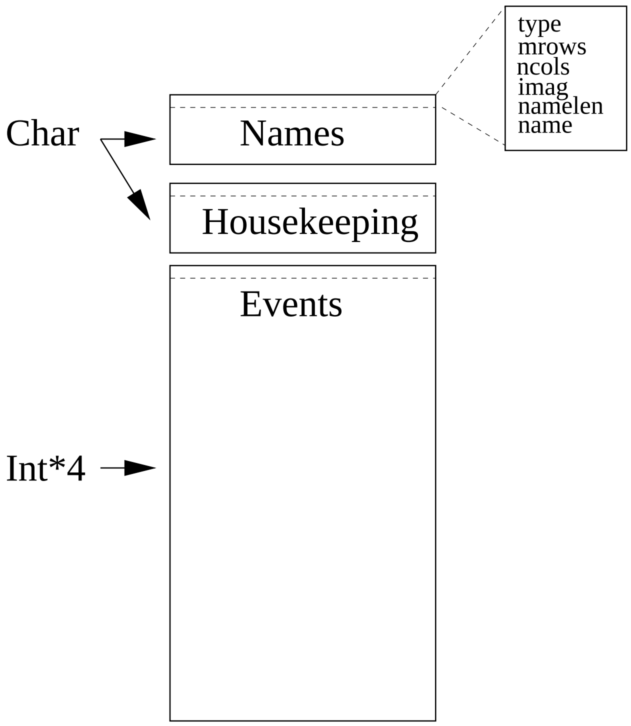

Figure 1: *A drawing of the CDMS ntuple file structure.*

3.1 Matlab header

The header tells the program that reads the file the type, size, and orientation of the contents.
The matlab style header is simple.

```
Word              type              Description
-----------------------------------------------
type(MOPT)       int*4             data type of matrix entries
mrows            int*4             number of rows
ncols            int*4             number of columns
imag             int*4             =0 if real, <>0 if imaginary
namelen          int*4             length of following word
matrixName       char*namelen      name of this matrix
```

The codes for the data types are most conveniently expressed in base 10 notation. It is a four digit number, which I will denote by *MOPT* for “Machine”,“Orientation”,“Precision”, and “Text”. So “O”, for example, is the 100’s digit.

Here are the code options for M (Machine ID/Architecture):

```
M Code   Desicription
-------------------
 0     PC/Intel
 1     Sun, Apollo, Macintosh, other Motorola
 2     VAX D-float
 3     VAX G-float
```

O is normally 0(zero), which means the data are stored in a column-wise orientation. This means that the data is stored columns at a time or varies quickly down a column. If $O = 1$, then the data is transposed and are stored row-wise.

The "P" represents the type/precision of the data. Here are the codes for "P".

```
P Code            type     typical
------------------------------------------
0               float*8 (double)
1               float*4 (float)
2               int*4   (int)
3               int*2   (short)
4               int*2   (unsigned short)
5               int*1   (byte)
```

"T" is normally 0. This indicates that the data that follows are a matrix of numbers. If T==1, then the variable is a text variable.

For example, to store a matrix of characters such as the names of the quantities, the code 1051 would be used on a Sun Workstation. The "1000" represents a Sun (big endian), the 50 represents a single byte per quantity, and the 1 represents the fact the information is text.

3.2 Names

The names matrix is the first matrix in the ntuple file. It contains a list of the names of the variables stored as characters.

3.3 Housekeeping

The housekeeping matrix is a set of information of the type:
`variableName = value`
written as characters to the file.

The currently stored variable names are `startDate`, the time (to the minute) and data of the beginning of the series. `startTime`, the time in seconds since Jan 1, 1904 of the first event of the series.

3.4 Ntuple

The ntuple is the last and usually the largest section of the file. It contains an entry for each event. The entries are all of type int*4.

```
Num   Name      Description
----------------------------
1     Series    Sequential series number.
2     Event     Event number from beginning of series.
3 (*) Trigger   Trigger type. Coded:
                detector type * 1000 + serial number * 10 + trigger type
4     EventTime Time since beginning of run in ms.
5     Tlive     Live time since previous event in ms.
6     vetoRate  in Hz.
7     vetoTime  time in ns to nearest veto hit.
8     vetoWord  bit pattern of nearest veto word.
9 (*) cutWord   Cuts performed upstream for up to 7 detectors.
                cuts are from 0 (good) to 15.
                The most significant 4 bits are global cuts,
```

```
                such as bad fridge temperature.
                |msb --- 24----- 16----- 8--- lsb|
                |cGl cD7 cD6 cD5 cD4 cD3 cD2 cD1 |
10    nDet      number of detectors with E > thresh.
11 (*)IDD1      id of first detector (D1)
                The detector ID is encoded:
                detector type * 10000 + serial * 100 + tower * 10 + position
12    prD1      recoil energy of D1 in eV.
13    chi2prD1  chi2 of recoil pulse of D1 * 1000.
14    eqD1      charge energy of D1 in eV.
15    chi2eqD1  chi2 of charge pules of D1 * 1000.
16    qpartD1   ((qi - qo) / qtotal )*1000
17    pr1D1     recoil energy in sensor 1 in D1 in eV.
18    pr2D1     recoil energy in sensor 2 in D1 in eV.
19    pr3D1     recoil energy in sensor 3 in D1 in eV.
20    pr4D1     recoil energy in sensor 4 in D1 in eV.
21 (*)xyzD1     position of hit in sensor 1. x,y, and z are measured
                in 1/10 mm and are packed into a single 32-bit word.
                |msb --- 24----- 16----- 8--- lsb|
                |<   z    ><    y    ><    x    >|
22    IDD2      Similar to D1
23    prD2          "
24    chi2prD2      "
25    eqD2          "
26    chi2eqD2      "
27    qpartD2       "
27    pr1D2         "
28    pr2D2         "
29    pr3D2         "
30    pr4D2         "
31    xyzD2         "
32    IDD3          "
33    prD3          "
34    chi2prD3      "
35    eqD3          "
36    chi2eqD3      "
37    qpartD3       "
38    pr1D3         "
39    pr2D3         "
40    pr3D3         "
41    pr4D3         "
42    xyzD3         "

(*) - See next section for details.
```

3.4.1 Ntuple Details

- Trigger Word: detector type*1000 + serial *10 + trigger type

```
detector type : 1 = BLIP, 2 = FLIP
serial number :  from 1 to 100.
```

```
trigger type:
  0 = Charge, 4 = Phonon,
  2 = Charge  IBAPACAP, 6 = Phonon IBAPACAP.
```

- The cutWord is a packed word with the most significant 4 bits indicating global cuts, such as environmental conditions making data taking bad. The lower 28 bits are divided into 7 groups of 4 bits, and cover up to 7 detectors. Each detector cut word ranges from 0 to 15. The codes are:

```
                       ENVIRONMENTAL CUTS
                      ====================
   Unspecified.

                         DETECTOR CUTS
                      ====================
 Code      Description
-----------------------------------------------------------
  0     Event good
  6     Abnormal pulse shape.
  8     More than 1 Charge trigger inside Phonon pulse.
 12     Glitch event (aka "invalid").
 14     Detector off (aka "bad time").
```

  A high value of the cut word implies that the event could also have been rejected for the lower values. Thus for a glitch event which also has an abnormal pulse shape, the cut will be 12.

- Detector ID word: det type * 10000 + serial * 100 + tower * 10 + posn

```
detector type :  = 1 BLIP,  = 2 FLIP
serial number :  from 1 to 100.
tower         :  see figure
position      :  lowest position = 1, highest = 6.
```

- The x and y positions are given 11 bits each. The z position uses the high order 10 bits. The positions are referenced to the center of each detector, with the positive x direction toward the east, positive y toward the south, and z upward.

  To translate from the bits to the co-ordinate, use `u = (bitval/10 - 80) mm`.

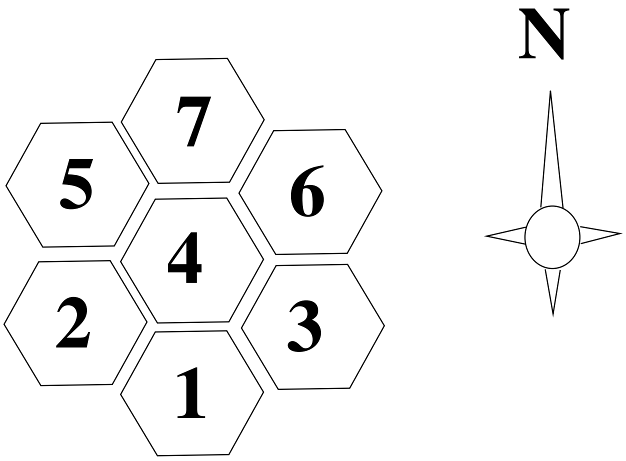

Figure 2: *The tower arrangement for CDMSII. The tower in runs 14-18 is called "Tower 1"*