F2000 (Version 1.1.1) —FINAL DRAFT—

Proposed AMANDA offline format
An ASCII data format for handling data and MC events
August, 1997

John Jacobsen, Ole Streicher,
Serap Tilav, Christopher Wiebusch

This manual incarnates in a `postscript`, (about 10 pages), `HTML` (`http://www.ifh.de/~baikal/software/sie toc.html`) and in an `emacs-info` version.

We encourage the interested reader to look into following document:

AMANDA Software Resources and Documentation
(http://alizarin.physics.wisc.edu/amanda/datamc/resources.html)

F2000 example page
(http://alizarin.physics.wisc.edu/amanda/datamc/f2000)

$Id: f2000.texi,v 1.15 1998/02/19 17:06:58 wiebusch Exp $

In case of bugs, questions, remarks, suggestions or flowers, please send them via e-mail to:
`datamc@alizarin.physics.wisc.edu`
This is a common mailing list available for discussion of AMANDA data analysis.

This document is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
Copyright © 1997 John Jacobsen, Ole Streicher, Serap Tilav, Christopher Wiebusch (The AMANDA collaboration)

Several symbols and names occuring in this documentation and in the source are registered trademarks.

1 Why a common ASCII Offline data format?

As announced in the Stockholm meeting, some of us have gotten together to forge a standard muon event format for everyone to use. While the goal is simple to state, to achieve agreement within even a small sub-group has been difficult, not to mention the actual implementation of the format in software.

The goal of the format (called F2000 in the hopes that it will be finished well before the millenium `:-)`) is NOT to replace the raw data (at this time). It is, rather, to provide a flexible means to store information of use both for Monte Carloists and for the real data, from before calibration to final event selections.

We designed this new format trying to obey the following principles

- Self-consistency in units and coordinate systems, as well as some definition, in the file header, of data to follow.
- Ease of human readability without egregious expansion of file size (25% increase in compressed size relative to compressed raw data but still a significant increase compared to uncompressed raw data)
- Flexibility, thanks to user-definable additions and expandable definitions.
- Superior to previous formats, namely `RAVEN` and `SiEGMuND`. This is a must in order to justify all the re-coding work to be done on existing software.

2 Format Description

2.1 General file structure

The format is line-oriented ASCII. It should be possible to easily process it using standard Unix-like shell utils like grep, perl... Whitespace within lines is allowed, and empty lines are allowed (see below).

Although not part of the standard, the authors suggest the Gnu gzip format for handling compressed data.

A line should not exceed 255 characters. We suggest less then 80 characters for enhanced readability.

All coordinates and times should be consistent within a file.

All kinds of AMANDA events are to be included: muon, snmp/GRB, housekeeping, ... At this point there are no definitions for other than muon events.

The format is "basically" the same for all steps of analysis also for MC and data. Calibration and geometry are optional. E.g., raw data won't have a calibration or geometry block. Event generator programs (airshow) won't produces hits.

The basic structure for muon data is as follows:

```
V 2000.X.Y   ! Header line including version
(History information)
(Calibration and geometry constants)
(Other definitions, e.g. trigger or user blocks)
TBEGIN
  Event header
  (MC particle information)
  (OM hit information)
  (Fit track/shower information)
  (User information)
  EE      ! End of event
  ...
TEND
END       ! End of file
```

"..." means event information repeats. Indentation above is for clarity. Lines in parentheses () are optional. A few examples can be seen below.

The format is whitespace-invariant within a line, except for TAG, which must be at the beginning of the line.

2.2 Units and coordinate systems

All units are nanoseconds, meters, GeV, except event times are in seconds. Angles are in degrees. Relative numbers are 0.0 to 1.0 (not percent); Rates are in Hertz, Steradians for solid angles, photoelectrons for calibrated amplitude, ADC counts for uncalibrated amplitude, (analogous for TDCs).

Besides numbers following strings may be stored:

`NaN` Not a Number. This usually indicates errors

`?` Not available. This number is unknown

`inf and -inf`
Infinity

The used system of coordinates is right-handed, with the Z direction upward, and X into direction east. The origin of the AMANDA coordinate system is at OM 10 in string 4. All coordinates (also SPASE) should be written in respect to this origin. We use a numbering `1..n`. We do NOT do want crazy numbering like `C` (`0..(n-1)`).

2.3 Format version line

The very first line MUST be a V-line with the used format version. Nothing else (no comments) are allowed before this line.

`V 2000.x.y`
Format version line

`int x` Format major version

`int y` Format minor version

2.4 Comments and history lines

Comments are all lines which start with a non-letter, like: `: ; # * ! % $ / , . " '`. There is no need for a space after this first character: `"/*"` or `"//"` are also valid.

Inline comments are `!` only, commenting the rest of the line. Lines can have blank spaces at the beginning, except for the first V header line. Inline comments are not necessarily preserved by filter programes

Blank lines are allowed after the `V` header line. Similarly, comments are only allowed after the `V` header.

`! comments`
Comment line

`string comments`
Any up to 80 (255) characters

To enable the possibility of backtracing the analysis procdure, we strongly recommend the usage of history lines indicating which programs produced the current output file. History information is specified with the `HI` tag.

`HI program (version) parameters`
: History line

  `string program`
  : Program name

  `string version`
  : Program version number

  `string parameters`
  : All command line parameters (or similar) the program was invoked with.

Programs should add their HI tags after the first V and after all existing HI tags. That way the history reads from top to bottom.

Example:

```
V F2000.1.1
! File created 10 Oct 1966 by RAVEN/genevent, user SERAP on ALIZARIN.
! Atmospheric neutrinos in this puppy...
HI genevent (1.1) -atmos_nus -N1000
! Reconstructed by wiebusch@ifh.de using recoos (SiEGMuND 1.666)
HI recoos (1.19) -W -V -bxV -X w=tanze
```

2.5 Header

The OM positions as well as the calibration are stored at the file header if they are needed or available.

2.5.1 Geometry

The geometry block starts with a line indicating the common geometry information, followed by one geometry line per OM:

`ARRAY detector longitude lattitude depth nstrings nchannel`
: Detector information line

  `string detector`
  : Name of the used detector. See Section 3.2 [Detector ids], page 12, for valid values here.

  `float longitude, lattitude`
  : Location of the detector in Earth coordinates.

  `float depth`
  : Depth of the detector center (origin of the coordinate system)

`int nstrings`
Total number of strings

`int nchannel`
Total number of channels

`OM number nr_str string x y z orientation type serial sensit thresh`
OM geometry line

`int number`
OM number, ranging from 1 to `nch`

`nr_str` OM number in this string

`int string`
String number, from 1 to `nstr`

`float x y z`
Position of the OM center

`string orientation`
Orientation is `"up"` for upward looking, `"dn"` for downward looking and a string like `"+0+180"` for other orientations (in the example, the OM is vertical, phi is 180).

`string type`
Type of the module. If possible, use the values defined in Section 3.3 [Optical module ids], page 13

`string serial`
Serial number of the module, or `?` if not available.

`float sensit`
Relative sensitivity: usually 1.0. These values will be experimentally adjusted. A dead OM receives 0.0.

`float thresh`
Threshold applied in P.E. to this channel.

2.5.2 Trigger definition

Each trigger (at least: hardware triggers) used in the file is defined here.

`TRIG_DEF number name word1 word2 ...`
Trigger definition line

`int number`
Trigger number (or order in case of the interupt register)

`string name`
Trigger name. See Section 3.4 [Trigger ids], page 14, for predefined trigger names.

`string word1 word2 ...`
Meanings of the numbers in the TRIG line

e.g. `TRIG_DEF 2 spase1 tdc-time spase-gps`

`TRIG_PAR number type=value tag1=value tag2=value ...`

`string number`
: Trigger number or name (from the TRIG_DEF definition).

`string name`
: Trigger name. See Section 3.4 [Trigger ids], page 14, for predefined trigger names.

`string tag=value`
: Assign specific values to predefined tags, See Section 3.4 [Trigger ids], page 14

e.g. TRIG_PAR amanda-b-4 type=majority window=2000 fold=8

2.5.3 Calibration

The calibration block starts with a line indicating which calibration has done, followed by one line per channel for ADC and TDC calibration, and by a event time calibration line if there was such a calibration.

`KH ADC TDC UTC`
: Calibration header line

`ADC, TDC` There was an ADC or TDC calibration. The tokens in the paranthesises show which calibration was done.

`UTC` There was a event time calibration.

`KADC number pedestal beta linearity`
: ADC calibration constants

`int number`
: OM number

`float pedestal`
: Pedestal value of the ADC

`float beta`
: Calibration coefficient (channels per photoelectron) for the ADC

`float linearity`
: Currently undefined

`KTDC number beta shift alpha`
: TDC calibration constants

`number` OM number

`float beta`
: Calibration coefficient (Nanoseconds per channel) for the TDC

`float shift`
: Time shift (t0) of this OM

`float alpha`
: Amplitude dependent time shift, in nsec/sqrt(pe)

`KUTC unit offset`
: Event time calibration

`string unit`
: One of the strings `GPS`, `UTC`, `DAQ` indicating the source of the event time (GPS clock, UTC module or DAQ system)

`float offset`
: Time correction in seconds against the time stored in the event

2.5.4 Status flag definitions

The status definitions are stored in one `STAT_DEF` line per status word in the header. Status words in the event refer to these lines.

`STAT_DEF nr tag word1 word2 ...`

`int nr` Status word number

`string tag`
: One word indicating this status line

`string word1, word2, ...`
: Meanings of the numbers in the status information

2.5.5 Fit definitions

The fit definitions are stored in one `FIT_DEF` line per fit in the header. Fit lines (mostly the fit results) refer to this header.

`FIT_DEF nr tag word1 word2 ...`

`int nr` Fit number

`string tag`
: One word indicating this fit

`string word1, word2, ...`
: Meanings of the numbers in the fit result line

2.6 Run start and stop

Each runs starts with a `TBEGIN` line and ends with a `TEND` line which hold the times of run start and stop. For a MC run, arbitrary times may be stored here as long as they are constistent with the event times.

`TBEGIN year day time`
`TEND year day time`
: Start and stop times of the run.

`int year` Year of run start/stop

`int day` Day number of run start/stop, since Jan 0th.

`float time`
: Seconds of run start/stop, since Midnight.

2.7 Events

Although we plan to store all possible event types in the format, we still define only the muon event type here.

Each muon event starts with an `EM` line and ends with an `EE` line. Other lines are for generated tracks and showers, hit information, reconstruction results, event and hit related triggers, and User defined data.

2.7.1 Event header

`EM enr run year day time tshift`
Event header

`int enr` Event number

`int run` Run number

`int year` The year (1996, ...) of the event

`int day` The day of the year (Jan 1st is day 1)

`float time`
Event time of the day (UTC), in seconds with 9 digits after the decimal point (accuracy up to 1 nsec)

`float tshift`
Shift all times in the event relative to this value, in nanoseconds. E.g. centre all hit-time in the hit-line around zero, requires to put the shift here and also to shift the times in track-lines and Fit-lines.

2.7.2 MC tracks and information.

`TR nr parent type xstart ystart zstart zenith azimuth length energy time`
Generated track and shower information

`int nr` Track number

`int parent`
Track number of parent track (e.g. in case of a secondary)

`string type`
Particle or shower type. See Section 3.1 [Particle ids], page 12, for valid values here.

`float xstart, ystart, zstart`
Starting point of the track, a valid point at a infinity track, or shower location

`float zenith, azimuth`
Direction of the track. `zenith=0` is a vertical downward going track.

`float length`
Track length. For point like showers, it is zero, for infinity tracks it is `inf`.

`float energy`
Particle energy, or `?` if not available.

`float time`
: Time that corresponds to the point `xstart, ystart, zstart`.

2.7.3 Hits and channels

`HT ch adc id parent tdc tot`
: Hit information

  `int ch` Channel number

  `float adc` Amplitude value (`*` for repeat channel)

  `int id` Pulse id for filtering, sorting etc.

  `int parent`
  : Track number which made the pulse (TDC). `N` for noise.

  `float tdc` Time of the pulse (Leading edge)

  `float tot` Time over threshold

`CH ch adc id1 parent1 tdc1 tot1 id2 parent2 tdc2 tot2 ...`
: Channel related hit information (alternatively to the `HT` line). A quadrupel of numbers (`id parent tdc tot`) is added for every hit in the channel.

  `int ch` Channel number

  `float adc` Amplitude value

  `int id1, id2, ...`
  : Pulse id for filtering, sorting etc.

  `int parent1, parent2, ...`
  : Track number which made the pulse (TDC). `N` for noise.

  `float tdc1, tdc2, ...`
  : Time of the pulse (Leading edge)

  `float tot1, tot2, ...`
  : Time over threshold

2.7.4 Hit related information

Under certain circumstances it is important to store information, which hits had been used by a trigger or a fit. This is possible by adding of USES lines after a TRIG or a FIT line. All USES-lines relate to the latest TRIG or FIT line. Since the length of a single line is limited to 255 chars, an arbitrary number of lines may follow, until all hits are listed.

`USES hit1 hit2 ...`
: `int hit1, hit2, ...`
  : These hits have been used by the above TRIG, or FIT line.

It is also allowd to give ranges e.g. `21-31` means OM 21 to 31.

2.7.5 Event trigger information

`TRIG id value1 value2 ...`
: Hit related trigger information

  `string id` Trigger id, as defined in the header

  `float value1, ...`
  : Trigger values, as defined in the header

2.7.6 Reconstruction results

`FIT nr type xstart ystart zstart zenith azimuth time length energy`
: Fit information

  `int nr` Fit number

  `string type`
  : Particle or shower type. See Section 3.1 [Particle ids], page 12, for valie values here.

  `float xstart, ystart, zstart`
  : Starting point of the track, a valid point at a infinity track, or shower location

  `float zenith, azimuth`
  : Direction of the track. `zenith=0` is a vertical downward going track.

  `float length`
  : Track length. For point like showers, it is zero, for infinity tracks it is `inf`.

  `float energy`
  : Particle energy, or `?` if not available.

  `float time`
  : Time that corresponds to the point `xstart, ystart, zstart`.

  `chi2` Chisquare, or another value indicating the fit success.

`FRESULT nr value1 value2 ...`
: `int nr` Fit number

  `float value1, value2, ...`
  : Fit results, as defined in the FIT_DEF line in the header

  Information about hits used in the fit, can be coded via the `USES` lines, See Section 2.7.4 [Hit related information], page 9.

2.7.7 Event status information

`STATUS tag s1 s2 ...`
: Status information line

  `string tag`
  : Status line tag, as defined in the header

  `int s1, s2, ...`
  : Status words. If hexadecimal, with a leading `0x`.

2.7.8 User defined information

To increase flexibility, user-defined information is allowed for events and even for hits.

`USER_DEF tag name1 name2 ...`
: Define a user tag (in the header).

  `string tag`
  : Tag that indicates the user values.

  `string name1, name2 ...`
  : up to 16 names of the fields in the user defined lines

`US tag value1 value2 ...`
: User defined values

  `string tag`
  : Name of this user defined line, must correspond to a tag in a `DU` line.

  `float value1, value2, ...`
  : up to 16 values, corresponding to the fields defined in the `DU` line.

2.7.9 Event end

`EE` End of the event.

3 Id tables

3.1 Particle ids

Muons

`mu`
`mu+`
`mu-`

Cascades

`em` electromagnetic cascade
`hadr` hadronic cascade

Muon energy loss results

`brems` Bremstrahlung
`delta` Delta electron
`epair` Electron pair production
`munu` Muon nucleon interaction
`mupair` Muon pair production

Particle ids (+ or - or ~ for anti-particles)

`p+ p-` Proton
`pi+ pi-`
`e+ e-`
`gamma`
`nu_e ~nu_mu, ....`

Laser or calibration sources

`n2laser`
`yaglaser` Fiber laser
`flaser`

Primary particle from airshower programs

`Zxxx` Nucleon with charge number xxx
`Axxx` Nucleon with Mass number xxx

3.2 Detector ids

`Amanda-A` Amanda A as installed in 1994
`Amanda-B-4`
Amanda B as installed in 1996
`Amanda-B-10`
Amanda installed in 1997

`Amanda-B-11`
: Amanda-B as proposed in 1996

`Amanda-II-20`
: The proposed Amanda-II detector starting in 1998

`Amanda-km3`
: The proposed Amanda km 3 detector as installed in 2004

`NT-36`
`NT-36b`
`NT-72`
`NT-96`
`NT-144`
`NT-192`
: The various stages of the Baikal detector

`NT-200`
: The proposed Baikal detector

`Julia`
: Ask Christopher

3.3 Optical module ids

Common naming scheme is OM-sphere-cable

`xp2600`
: Phillips XP-2600 (14 inch)

`hr2018`
: Hamamatsu R2018 (15 inch)

`hr5212`
: Hamamatsu R5212 (8 inch, Amanda)

`emi8`
: EMI (8 inch, Amanda)

`q370`
: Quasar-370 (Baikal)

`naut`
: Nautilus sphere (jena-glas)

`bent`
: Benthos sphere

`bill`
: Billings sphere

`russ`
: Russian sphere (Baikal)

`coax`
: Coaxial cable

`tp`
: Twisted pair cable

`opt`
: Optical fiber

`dig`
: Digital data transmission

So, common OMs are named as follows:

`emi-bill-coax`
: Standard Amanda-A OM

`r5212-bent-coax`
: Standard Amanda-B-4 OM

`r5212-bent-tp`
: Standard Amanda-B-11 OM

`r5212-bent-opt`
: Optical fiber OM in Amanda

`q370-russ-coax`
: Standard Baikal OM

3.4 Trigger ids

`type=majority`
: Majority trigger as Amanda-B

`type=multiplicity`
: Simple multiplicity on total number of hits

`window=size`
: size of the trigger window

`fold=value`
: fold value of e.g. a mojority trigger

3.5 Fit ids

To be defined

4 Changes to previous format versions

4.1 Changes to Version 1.1.0

- DU renamed to USER_DEF
- USES line re-added
- added the TRIG_DEF and TRIG_PAR definitions
- HT-line ordering of tokens changed.
- added trigger ids for TRIG_PAR line
- Some ids for e.g. pmt ... slightly changed
- removed () in KH line

Table of Contents

1 Why a common ASCII Offline data format? 1

2 Format Description 2

2.1 General file structure 2
2.2 Units and coordinate systems 3
2.3 Format version line 3
2.4 Comments and history lines 3
2.5 Header 4
2.5.1 Geometry 4
2.5.2 Trigger definition 5
2.5.3 Calibration 6
2.5.4 Status flag definitions 7
2.5.5 Fit definitions 7
2.6 Run start and stop 7
2.7 Events 8
2.7.1 Event header 8
2.7.2 MC tracks and information. 8
2.7.3 Hits and channels 9
2.7.4 Hit related information 9
2.7.5 Event trigger information 10
2.7.6 Reconstruction results 10
2.7.7 Event status information 10
2.7.8 User defined information 11
2.7.9 Event end 11

3 Id tables 12

3.1 Particle ids 12
3.2 Detector ids 12
3.3 Optical module ids 13
3.4 Trigger ids 14
3.5 Fit ids 14

4 Changes to previous format versions 15

4.1 Changes to Version 1.1.0 15