F2000 (Version 1.5)

AMANDA offline format
An ASCII data format for handling data and MC events
updated June, 2001

This manual incarnates in a `postscript` (about 25 pages),
`HTML` (`http://www.ifh.de/~steffenp/f2000/f2000_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)

rdmc manual
(http://www.ifh.de/baikal/software/siegmund/rdmc_toc.html)

F2000 example page
(http://www.ifh.de/~steffenp/f2000)

In case of bugs, questions, remarks, suggestions or flowers, please send them via e-mail to:
`f2000@mail.ifh.de`
This is a common mailing list available for discussion of AMANDA data format.

1 Why a common ASCII offline data format?

As announced in the Stockholm meeting (summer 1997), 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 millennium `:-)`) 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 Carlo 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 utilities like `grep`, `perl`, ... Whitespace within lines is allowed, and empty lines are allowed. Float numbers may written in any way like '`10000`', '`10000.0`', '`1.0E+5`', '`1.0e+5`', '`.1e+6`', '`1000000.e-2`', '`1e+5`', ... as known from C and FORTRAN I/O.

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. If line is longer than maximum allowed length, it can be continued on the next non-comment line (see Section 2.2 [Units and conventions], page 3).

All coordinates and times should be consistent within a file, e.g.:

> If times need to be shifted it must be done such that all times in the event (trigger, hits, tracks) have to be shifted by the same amount.
>
> Coordinates of spase, amanda_a, amanda_b must refer to the same origin even after some transformation.

All kinds of AMANDA events are to be included: muon, snmp/GRB, housekeeping, ...

The format is "basically" the same for all steps of analysis for both, MC and data. Most parts of format are optional. Raw data won't have a calibration or geometry block, while event generator programs (e.g. `airshow`) won't produces hits.

The basic structure for data file is as follows:

```
V 2000.X.Y      ! Header line including version, must be present
ARRAY ...  ! Detector type and common geometry information, must be present
(History information)
(Calibration and geometry constants)
(Other definitions, e.g. trigger or user blocks)
ES ...    ! Slow event header
   (Status information)
EE             ! End of event
...
EM ...    ! Muon event header
   (MC particle information)
```

```
    (OM hit information)
    (Trigger information)
    (Fit track/shower information)
    (User and other information)
EE            ! End of event
...
END           ! End of file
```

Ellipses between events mean that event information repeats. Indentation above is only for clarity. Lines in parentheses () are optional, while muon and slow events can come in any order. A few examples can be seen below.

The format is whitespace-invariant within a line.

2.2 Units and conventions

All units are considered to be in nanoseconds, meters, and GeV, except event times and event times corrections which are given in seconds. Planar angles are in degrees and solid angles are in steradians. Rates are in Hertz. Relative numbers are 0.0 to 1.0 (not percent). Calibrated amplitude is given in photoelectrons, while all uncalibrated values (including TDCs and TOTs) are given in raw data counts specific for that variable.

Data is stored in form of numbers and single word strings. If Required, numbers can be treated as strings. All numbers are considered to be float point numbers (unless explicitly stated otherwise in this document), but they don't require trailing point if not needed (e.g. '`10`' and '`10.`' are both valid).

> **Please note:** No constraints are placed on how software processing F2000 data deals with numerical values. Value '`10`' can be treated as float, double, integer, or anything else.

All numbers should be in decimal format, i.e. binary, octal, and hexadecimal representations are not allowed.

The numbering used is `1..n`. C-like numbering `0..(n-1)` is **NOT** allowed.

2.2.1 Special characters

Following special strings may be stored in the place of numbers:

`NaN` Not a Number. This usually indicates error.

`?` Not available. This variable is unknown. It can also represent an unknown string value.

`*` Same numerical value as before, e.g. ADC value for multiple hits on the same channel. It can't represent repeated string.

`inf`
`-inf` Positive and negative infinity

2.2.2 Absolute geometry

The coordinate system used is right-handed, with the Z direction upward and the X direction eastward (grid-east direction at the South and North poles). The origin of the AMANDA coordinate system is at module 10 on string 4 (OM 70) as defined by geometry files prior to 1998. Currently, AMANDA coordinate system places OM 70 at (1.53,-1.63,-25.90). All coordinates (including SPASE) should be written with respect to this origin.

2.2.3 Channel naming

The channel naming convention is following:

`channel id = OM.i`
where
`OM = OM number`
`i  = 1, 2, 3,` ... for different readout channels of the same OM

The backward compatibility is insured through definition;
`channel id = OM` ≡ `OM.1`

It should be noted that prior to version 1.4, the secondary channel readout was encoded by 10000+OM number. This was never an official convention, but user should be aware of existence of such numbering scheme.

2.2.4 Fit naming

The fitted track naming convention is following;

`fit id = Fit.i`
where
`Fit = Id of a global fit`
`i   = 1, 2, 3,` ... for each track determined by the fit

The backward compatibility is insured through definition;
`fit id = Fit` ≡ `Fit.1`
which also can be used when global fit defines only one track.

2.2.5 Continuation lines

No line should exceed 255 characters. If data line is too long, it can be continued on the next line which should start with &.

```
STATUS spase word1 word2 ... word10       ! A very long line
& word11 word12                            ! Continuation of STATUS spase
```

A line can be broken only between words. Any number of comment lines and blank lines can appear between a line and its continuation. Comment lines can't be continued (they should be split into multiple comments).

> **Important:** A continued line is logically considered to be only one line, regardless of its actual length. Upper limit of 255 characters is imposed to prevent any incompatibilities with text processing software.

2.2.6 Document conventions

Following variable conventions are used in this document:

`id` is a unique number or string identifying object in question.

`word1 .. wordn`
: are strings.

`value1 .. valuen`
: are numbers or strings.

2.3 Format version line

The very first line **MUST** be a version line indicating which format version is used in the file. Nothing (no comments) can precede this line and this line should not be indented (i.e. 'V' should be first character of data stream).

`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 character, like `: ; # * ! % $ / , . " '`, except &. There is no need for a space after this first character, e.g. '/*' or '//' are also valid.

Inline comments start with `!` only, commenting out 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 programs processing data.

Blank lines are allowed after the version line. Similarly, comments are only allowed after the version line.

`! comments`
: Comment line

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

To enable the possibility of back-tracing the analysis procedure, the usage of history lines is strongly recommended. They indicate 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.

A program should add its history line after all existing history lines. 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

All information relating to all events in the file is stored in the header. This includes OM positions, channel calibration data, user and other object definitions.

2.5.1 Detector

The detector block consists of the ARRAY line indicating the detector type and common geometry information. This line **MUST** appear in an F2000 data file.

`ARRAY detector longitude latitude depth nstrings nmodule`
: Detector information line

  `string detector`
  : Name of the detector used; see Section 3.2 [Detector ids], page 20, for predefined examples

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

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

  `int nstrings`
  : Total number of detector strings

  `int nmodule`
  : Total number of detector modules

2.5.2 Calibration

The calibration block starts with a line indicating which type of calibration has been performed, followed by one line per calibration of the different channel properties, and by an event time calibration line if there was such a calibration.

`KH ADC TDC TOT UTC GEO`
: Calibration header line. The tokens on the line show which calibration has been performed on the data.

  `ADC` ADC calibration performed

  `TDC` TDC calibration performed

  `TOT` TOT calibration performed

  `UTC` Event time calibration performed

  `GEO` Geometry calibration performed

In format versions prior to 1.3, the `GEO` option was not present. Instead, the convention was that the presence of `OM` lines indicates the performed geometry calibration.

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

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

`int nr_str`
OM ordering on its string, starting from the top of the string

`int string`
String number of an OM, 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 (e.g. this OM is vertical, with `phi` equal to 180 degrees).

`string type`
Type of the module. If possible, use the values defined in Section 3.3 [Optical module types], page 20

`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 to this channel in P.E.

`KADC ch pedestal beta linearity`
ADC calibration constants

`string ch` Channel id

`float pedestal`
Pedestal value of the ADC

`float beta`
Calibration coefficient (channel counts per photoelectron) for the ADC

`float linearity`
Currently undefined

`KTDC ch beta shift alpha`
TDC calibration constants

`string ch` Channel id

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

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

`float alpha`
: Amplitude dependent time shift, in nsec/sqrt(P.E.)

`KTOT ch pedestal beta linearity`
: TOT calibration constants

  `string ch` Channel id

  `float pedestal`
  : Pedestal value of the TOT

  `float beta`
  : Calibration coefficient (nsec per photoelectron) for the TOT

  `float linearity`
  : Currently undefined

`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 of the time stored in the event

2.5.3 Custom definitions

To give the largest flexibility for encoding different data in F2000 format definition mechanism provides ability to customize what information is stored with each event in the data file. The format currently provides four predefined information categories, plus catch-all '`user`' defined category. All "define-type" information lines used in any event in the data file have to be defined in the header.

Each definition consists of two lines. `DEF` line specifies *exact* formating of corresponding line to be found in data events. The suggested use is to provide variable names for values found in data events. `PAR` line specifies certain fixed values associated with definition that hold true for all events in the data file. `PAR` line can't precede or appear without accompanying `DEF` line.

2.5.3.1 Trigger definitions

Trigger lines are intended to represent conditions that certain event satisfies. They can be split into two categories;

Hardware triggers
represent status of detector triggering hardware at the time when the event was recorded (e.g. AMANDA trigger, SN network alert coincidence trigger, ...).

Software triggers
represent that event satisfies certain criteria defined by the trigger (e.g. filtering level, time coincidence with some other event, ...).

In reality, the distinction is not that clear cut since MC code can simulate hardware triggers, etc.

`TRIG_DEF id word1 word2 ...`
Trigger definition line
(It has been "`TRIG_DEF id tag word1 word2 ...`" up to and including version 1.2).

`string id` Trigger id is a unique trigger name; see Section 3.4 [Trigger ids], page 21, for predefined trigger names

`string word1 word2 ...`
Meanings of the values on the `TRIG` line

e.g. `TRIG_DEF spase1_coinc tdc-time spase-gps`

`TRIG_PAR id tag1=value tag2=value ...`

`string id` Trigger id (from the `TRIG_DEF` definition)

`string tag=value`
Assign specific values to predefined tags

e.g. `TRIG_PAR amab-4 type=majority window=2000 fold=8`

2.5.3.2 Status definitions

Status lines are intended to represent state of detector and/or data gathering and processing system at the time when the data event was recorded. For example, this can be current temperature inside counting room, information on malfunction of certain data channels, or anything else.

`STAT_DEF id word1 word2 ...`

`string id` Status id is a unique name of the status line

`string word1, word2, ...`
Meanings of the values on the `STATUS` line

e.g. `STAT_DEF hv channel crate hv_request hv_supply`

`STAT_PAR id tag1=value tag2=value ...`

`string id` Status id (from `STAT_DEF` definition)

`string tag=value`
Assign specific values to predefined tags

e.g. `STAT_PAR hv crate1_model=1440 crate2_model=1458`

2.5.3.3 Fit definitions

Fit definitions declare which fits have been performed on data.

> **Important:** `FIT_DEF` doesn't specify formating of `FIT` lines which is predefined by F2000 format (see Section 2.6.8 [Event reconstruction results], page 16). It specifies formating of `FRESULT` line associated with given fit.

`FIT_DEF id word1 word2 ...`

`string id` A unique global fit id

`string word1, word2, ...`
Meanings of the values on the `FRESULT` line

e.g. `FIT_DEF rdmc-jk_1 rchi2 prob chi2`

`FIT_PAR id tag1=value tag2=value ...`

`string id` Fit id (from `FIT_DEF` definition)

`string tag=value`
Assign specific values to predefined tags

e.g. `FIT_PAR rdmc-jk_1 fitter=recoos type=linefit`

2.5.3.4 MC definitions

MC lines are intended to encode event specific MC information, e.g. various probabilities and weights associated with an event.

`MC_DEF id word1 word2 ...`
Define a unique MC id in the header.

`string id` A unique id identifying the MC information

`string word1, word2 ...`
Meanings of the values on the `MC` line

e.g. `MC_DEF corsika_1 weight seed1`

`MC_PAR id tag1=value tag2=value ...`

`string id` MC id (from the `MC_DEF` definition)

`string tag=value`
Assign specific values to predefined tags, e.g. Section 3.5 [MC list], page 22

e.g. `MC_PAR corsika_1 generator=corsika rng_type=run3`

2.5.3.5 User defined information

If predefined categories don't satisfy users' needs, users are allowed to define user specific lines. All responsibility for properly identifying information on those lines is up to the users. To increase flexibility, user defined information is allowed for entire events and for specific hits. See Section 2.6.10 [Event user information], page 17, for more details.

`USER_DEF id word1 word2 ...`

: `string id` A unique id identifying the user information.

: `string word1, word2 ...`
  Meanings of the values on the `US` line

`USER_PAR id tag=value, ...`

: `string id` User id (from `USER_DEF` definition)

: `string tag=value`
  Assign specific values to predefined tags

2.6 Events

Slow events are intended to store information about detector and/or data gathering and processing system in continual fashion independent from muon events. They can be also used to segment and keep track of muon event data in non-file oriented chunks. Each slow event starts with `ES` header line and it ends with an `EE` line. Presently, slow events are only allowed to contain status lines.

Muon events are intended for recording data coming out of main experimental DAQ and for MC simulation of such data. All additional information produced by subsequent processing of such events is also stored inside events.

> **Please note:** *Muon event* is a historical misnomer. Data contained in these events is allowed to be generated by any particle or calibration device emitting or simulating light inside of detector.

Each muon event starts with an `EM` header line and ends with an `EE` line. Event can contain generated tracks and showers, hit information, reconstruction results, all defined lines, and hit related info lines.

2.6.1 Slow event header

`ES name year day seconds`
Slow event header

`string name`
Type of the slow event. See Section 3.6 [Slow event names], page 22, for currently defined types.

`int year` The year of the event, e.g. 1996, ...

`int day` The day of the year, e.g. Jan 1st is day 1

`float seconds`
Event time of the day (UTC), in seconds. Accuracy up to 1 ns is possible (9 digits after decimal point).

2.6.2 Muon event header

`EM enr run year day time tshift`
Muon event header

`int enr` Event number

`int run` Run number

`int year` The year of the event, e.g. 1996, ...

`int day` The day of the year, e.g. Jan 1st is day 1

`float time`
: Event time of the day (UTC), in seconds. Accuracy up to 1 ns is possible (9 digits after decimal point).

`float tshift`
: Shift (in nsec) that has been applied to *all* times in the event. For example, centering all hit times on the hit lines around zero requires;
  - Set `tshift` here
  - Adjust times on hit, track, and fit lines simultaneously

2.6.3 MC tracks

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

`int nr` Track number

`int parent`
: Track number of parent track (in case of a secondary track)

`string type`
: Particle or shower type; see Section 3.1 [Particle ids], page 19, for valid values

`float xstart, ystart, zstart`
: Starting point of the track, a valid point for an infinite 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 infinite tracks it is `inf`

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

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

2.6.4 Hits

`HT ch adc id parent le tot edge`
: Hit information

`string ch` Channel id reporting the hit

`float adc` Amplitude value (`*` for repeated value) `int id` A unique pulse id for filtering, sorting, etc.

`int parent`
: Track number producing the hit in the case of MC data. Special values 'N' for noise and 'A' for afterpulse can be used. For real data this should always be ?.

`float le` Leading edge time of the pulse

`float tot` Time over threshold value of the pulse

`string edge`
: Number of TDC edges recorded; an integer that can be preceded by >, which marks that a TDC overflow bit was set, e.g. '>16'

2.6.5 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. A `USES` line relates to the closest preceding `TRIG` or `FIT` line in the event.

> **Please note:** To insure backward version compatibility (line continuation was not allowed before version 1.4), multiple `USES` line may be used to list all hits in the case when more than 255 characters are needed. However, this use is now discouraged and line continuation should be used instead.

`USES word1 word2 ...`

`string word1, word2, ...`
: These hits have been used by the above trigger or fit line. A `word` can be either a hit id (e.g. '19') or a hit id range (e.g. '21-31') implying that all hits between 21 and 31 have been used, including 21 and 31 (i.e. range [21-31]).

2.6.6 Waveforms

Multiple waveforms can be defined for each readout channel. It is assumed that they should not overlap in time.

> **NB:** Long waveforms (exceeding 255 characters) should be broken into multiple lines; see Section 2.2.5 [Continuation lines], page 5.

`WF ch id n le dt value1 ... valuen`
: Waveform information

`string ch` Id of a channel reporting a waveform `int id` A unique waveform id

`int n` Number of waveform bins

`float le` Beginning time of the first waveform bin

`float dt` Time width of each waveform bin

`float value1, ..., valuen`
Amplitudes of waveform bins

2.6.7 Event trigger information

`TRIG id value1 value2 ...`
Event trigger information. Multiple triggers with the same id can appear in the same event.

`string id` Trigger id, as defined by `TRIG_DEF` line in the header

`float value1, value2, ...`
Trigger values, as defined by `TRIG_DEF` line in the header

2.6.8 Event reconstruction results

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

`string id` Id of the fitted track; see Section 2.2.4 [Fit naming], page 4, for conventions

`string type`
Particle or shower type; see Section 3.1 [Particle ids], page 19, for valid values

`float xstart, ystart, zstart`
Starting point of the track fit, a valid point for an infinite track fit, or shower fit location

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

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

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

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

`FRESULT id value1 value2 ...`

`string id` The fitted track id, identical to the id of the fit line with which it is associated

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

Information about hits used in the fit, can be encoded via the `USES` line, Section 2.6.5 [Hit related information], page 15

Please note: `FRESULT` line can't appear without accompanying `FIT` line

2.6.9 Event status information

`STATUS id value1 value2 ...`
Status information line

`string id` Status line id, as defined by `STAT_DEF` line in the header

`float value1, value2, ...`
Status words, as defined by `STAT_DEF` line in the header

2.6.10 Event user information

User defined lines can refer to either entire event or only to a specific hit in the event. If a `US` line immediately follows `HT` line (ignoring any comment lines), that user defined line is associated with that specific hit. Otherwise the user defined line refers to the entire event.

`US id value1 value2 ...`
User defined line

`string id` User defined line id, as defined by `USER_DEF` line in the header

`float value1, value2, ...`
User defined values, as defined by `USER_DEF` line in the header

2.6.11 Event MC information

`MC id value1 value2 ...`
Event related MC information

`string id` MC id, as defined by `MC_DEF` line in the header

`float value1, value2, ...`
MC values, as defined by `MC_DEF` line in the header

2.6.12 Event end

`EE` End of an event.

3 Id tables

The following tables list majority of token values used in F2000. Format is not limited to values defined here, but all programs operating on F2000 data should treat the same values in the same fashion.

3.1 Particle ids

Muons

`mu, mu-` Muon
`mu+` Antimuon

Cascades

`em` electromagnetic cascade
`hadr` hadronic cascade

Muon energy loss results

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

Particle ids (+-, ~ for uncharged anti-particles)

`p+, p-` Proton, anti-proton
`pi+, pi-` Charged pions
`e+, e-` Positron, electron
`gamma` Photon
`nu_e, ~nu_e`
`nu_mu, ~nu_mu, ...`
Neutrinos, anti-neutrinos

Laser or calibration sources

`n2laser` Nitrogen laser
`yaglaser` Surface YAG laser, injected through fiber
`flaser`
`led` Light emitting diodes

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 1993/1994

`amanda-b-4`
  Amanda B as installed in 1995/1996

`amanda-b-10`
  Amanda installed in 1996/1997

`amanda-b-11`
  Amanda-B as proposed in 1996

`amanda-b-13`
  Amanda installed in 1997/1998

`amanda-ii-20`
  The proposed Amanda-II detector starting in 1998

`amanda-ii`
  Amanda-II as deployed in 1999/2000 (19 strings)

`icecube` The proposed Amanda km3 detector

`baikal-nt-36`
`baikal-nt-36b`
`baikal-nt-72`
`baikal-nt-96`
`baikal-nt-144`
`baikal-nt-192`
  The various stages of the Baikal detector

`baikal-nt-200`
  The proposed Baikal detector

`julia` Ask Christopher Wiebusch

3.3 Optical module types

Common naming scheme is PMT-sphere-cable

`xp2600` Phillips XP-2600 (14 inch)
`r2018` Hamamatsu R2018 (15 inch)
`r5912` Hamamatsu R5912 (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

`r5912-bent-coax`
Standard Amanda-B-4 OM

`r5912-bent-tp`
Standard Amanda-B-10 OM

`r5912-bent-opt`
Optical fiber OM in Amanda

`r5912-bent-dig`
Digital readout OM in Amanda

`q370-russ-coax`
Standard Baikal OM

3.4 Trigger ids

`spase1_coinc`
coincidence with SPASE 1 trigger

`spase2_coinc`
`spase2` coincidence with SPASE 2 trigger

`amaa` Amanda-A trigger

`amab4` Amanda-B-4 multiplicity trigger

`amab10` Amanda-B-10 multiplicity trigger

`amaab` Coincidence between Amanda-A and Amanda-B multiplicity triggers

`string` String multiplicity trigger

`Bsc1` Downscaled Amanda-B-10 low multiplicity trigger

`string`	String trigger
`LTDC`	Laser calibration trigger
`main`	Combined trigger

3.5 MC list

`basiev`	muons generated and traced with program basiev
`corsika`	muons from airshower program corsika
`muo0`	muons generated by muo0
`MMC`	muons propagated with MMC
`mudedx`	muons propagated with mudedx
`mum`	muons propagated with mum
`music`	muons traced through the ice with music-program
`amasim`	detector response simulated with amasim

3.6 Slow event names

`fbegin` `fend`	start and end of data file
`rbegin` `rend`	start and end of data run
`hv`	high-voltage supply status
`temp`	temperature status
`scaler`	readout of scaler data
`SN`	readout of supernova DAQ

4 Changes to previous format versions

4.1 Changes to Version 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 () around values on `KH` line
- `version` field added to history line

4.2 Changes to Version 1.1

- Added TDC edge field to `HT` line

4.3 Changes to Version 1.2

- Removed TDC edge field from `HT` line
- `ARRAY` line is a MUST; it describes the detector type
- "*" describes repeated values
- Conventions added to units and coordinate system
- new `MC_DEF`, `MC_PAR` and `MC` lines
- `OM` line description moved to Calibration section
- `KH` line with new indicators for `TOT` and `GEO` calibration
- `TRIG_DEF` line without the tag; id is a unique identification
- `FIT_PAR` line defining fit_id and type of fit
- `FIT` line uses tag for correlation with `FIT_DEF` line
- `STAT_PAR` line defined in an analogous way
- `USER_PAR` line defined in an analogous way
- Detector ids changed
- Particle ids; "~" for neutral anti-particles only
- Fit ids
- MC ids
- `CH` lines eliminated **retroactively**

4.4 Changes to Version 1.3

- Line continuation procedure defined
- Slow events introduced and conventions for slow_event_names added
- Channel naming convention defined
- Clarified relation between `FIT` and `FRESULT` lines
- Clarified the meaning of `US` line placement within data file
- `TBEGIN`/`TEND` lines are removed from the format. Their functionality is taken over by `fbegin`/`fend` slow events.
- Removed any requirements for `PAR` lines. They can encode any (`tag`,`value`) pair in any order.
- Removed fit ids
- Removed `chi2` field from `FIT` line description
- Removed one digit from F2000 version numbering scheme

4.5 Changes to Version 1.4

- TDC edge field re-introduced to `HT` line
- Waveform (`WF`) line defined
- Fitted track id modified to account for multiple tracks per one fit
- Introduced special symbol '`A`' for afterpulse as a hit parent
- Tables expanded and clarified

Table of Contents