# XAS accumulation programs

L.Chiappetti - IFCTR - Mon, 25 Sep 1995
DAWG-REP.9.0/95

This is an attempt to provide some information on existing accumulation programs, update the table of packetcap fields originally described in `DAWG-REP.5.0/93` and provide some basic rules for packetcap (and PCF) usage and composition.

Since the issue of the original note, a number of working accumulation programs have been written using a set of library routines built around the packetcap concept. Recently I have also added support to Housekeeping data, elaborating on the packetcap concept into the Parameter Characteristics File (PCF) concept.

## 1 Accumulation program basics (user point of view)

The following is the way *default simple* accumulations are called. By *simple* I mean that instrument-dependent corrections are not yet included (prototype examples and considerations exist in the MECS calibration documentation but are outside of the scope of the present note). *Default* accumulations refer to standard cases. First we present the procedure and later comments to each numbered line :

```
1       xasset rootdir, fotdir, datadir as appropriate
2       xasset spacecraft sax
3       xasset instrument eeee
4       xasset flight on
        xasset slew n|i|f
5       xasset quiet on
6       concatenate obs-list
7       accumulate image file [pkt] x1 x2 xz y1 y2 yz a1 a2 b1 b2 ....
8       accumulate spectrum file [pkt] e1 e2 eb a1 a2 b1 b2 ....
9       accumulate time file [pkt] t1 t2 tb a1 a2 b1 b2 ....
10      accumulate photon file [pkt] yna ynb ....a1 a2 b1 b2 ...
11      xasset hkconvert Enable|Disable
12      accumulate hk file parameter [pkt]  t1 t2 tb
```

1 these commands notify XAS the directories where input (FOT) and output data will reside see `DAWG-REP.7/92`

2 optional, will be set automatically by the first accumulation program

3 mandatory, to notify XAS of what instrument shall be processed

4 for programs which can process both flight (FOT) and ground calibration data. For convenience current default is off, i.e. ground data. Will be changed in the future.
For FOT data it might be necessary to specify if data pertain to an `i`nitial or `f`inal slew, or `n`o slew (i.e. `n`ormal pointed data, default)

5 some programs display "progress" messages every n packets or events processed to show they are working. To prevent this, which might slightly slow down the accumulation, quiet mode can be set.

6 to notify XAS the list of observations to be concatenated. Support to discriminate slew (in,out) and pointed data is not implemented yet (via xasset). For ground calibration data (MECS) run list can be used instead of observation list. Lists are in form *a*+*b*-*c*+*d*, i.e. a + separates individual items, and item can be a single observation (*a*,*d*) or a dash-separated range (or interval *b*-*c*).

7 to accumulate a default spatial image (with quantities named X and Y on the axes). The image size is determined by the selected range in X (*x1 x2*) and Y (*y1 y2*), and by the zoom factors (*xz yz*). Additionally one can select ranges (*a1 a2* etc.) for any other quantity. The list of other ranges depend on the content of the packet.
Also if more packets exist compatible with image accumulation one must indicate the packet type *pkt* (e.g m1dir003)

8 to accumulate a default energy spectrum (with quantity named PHA on the axis). The spectrum size is determined by the selected range in PHA (*e1 e2*), and by the binning factor (*eb*). Additionally one can select ranges (*a1 a2* etc.) for any other quantity. The list of other ranges depend on the content of the packet.
Also if more packets exist compatible with spectrum accumulation one must indicate the packet type *pkt* (e.g m1dir003)

9 to accumulate a default time profile (with quantity named TIME on the axis). The file size is determined by the selected range in time (*t1 t2*), and provisionally by the binning factor (*tb*), laster to become a bin size. Additionally one can select ranges (*a1 a2* etc.) for any other quantity. The list of other ranges depend on the content of the packet.
Also if more packets exist compatible with time accumulation one must indicate the packet type *pkt* (e.g m1dir003)

10 to accumulate a photon list file. Since the list of quantities (a,b,...) present in the input packets depend on packet type, exact syntax is not constant. For each quantity *q* one can select whether it is included (yn*q* is Y) or not (yn*q* is N) in the output file. For all quantities on can select ranges (*a1 a2* etc.).
Also if more packets exist compatible with photon accumulation one must indicate the packet type *pkt* (e.g m1dir003)

11 controls whether HK time profiles are in digital or physical units (conversion disabled or enabled, the default is enabled when meaningful).

12 to accumulate a time profile of a single HK *parameter*. See below for mnemonic names. Also time limits and binning factor (t1 t2 tb) are to be specified.
Also if more packets exist compatible with HK accumulation for that parameter one must indicate the packet type *pkt* (e.g m1eng000)

For non-default accumulations (e.g. burstlength spectra, pseudoimages etc.) there are two ways (indicated as a and b) of doing it.

```
13      xasset xquantity ax
        xasset yquantity ay
7a      accumulate image file [pkt] x1 x2 xz y1 y2 yz a1 a2 b1 b2 ....
14      xasset xquantity ax
8a      accumulate spectrum file [pkt] e1 e2 eb a1 a2 b1 b2 ....
or
15      xasset accummode pseudo
7b      accumulate image file [pkt] ax ay x1 x2 xz y1 y2 yz a1 a2 ...
8b      accumulate spectrum file [pkt] ax e1 e2 eb a1 a2 b1 b2 ....
```

13 notifies XAS to use non-default quantities for the X and Y axes of an image. The setting remains for all subsequent accumulations until reset.

14 the same for unidimensional accumulations (spectra)

7a 8a this way the syntax of the accumulate command remains the same as in the default case

15 setting the pseudo accumulation mode, XAS has no predefined x and y quantities, and is instructed to ask the user, therefore

7b 8b the user must pass the wished quantities as extra arguments (in **bold**) on the runstring or when prompted.

Similar syntax are used (some improvements necessary) for the cross accumulations (xaccumulate) and will be documented later.

There are also two programs in non-final form (`dtaccum` and `dtover`) which use similar syntax to accumulate time profiles or statistics using the information in the packet header (e.g. events/packet, events lost for packet unavailability).

# 2 Accumulation program basics (programmers' point of view)

## 2.1 Dispatcher concept

The `accumulate` and `xaccumulate` programs are actually just front-end *dispatchers* (the concept of dispatcher was presented at the DAWG meeting of 19 Jul 93 - see handout - and further at the XAS tutorial in Dec 94) which `z_run`s the appropriate program. The actual accumulators are called `sax*accum` and `x*accum`. (replace * with `i,s,t,p,hk`)
For the case of images and spectra the dispatcher concept extends on two level, and `saxiaccum` and `saxsaccum` are just front end which prepare the correct run string (retrieving `xquantity` and `yquantity`) for the actual accumulators `saxgaccum` and `saxhaccum`. (g stands for generic, and h for histogram).
Cross-accumulators currently exist only at the level of `xgaccum` and `xhaccum`.

For more details see program code and comments.

## 2.2 Accumulation program structure

The typical accumulation program (look at any of `sax*accum` in `$XASTOP/source`) is constructed as follows:

There is an initial dialogue including :
- some program specific dialogue (to get the file name etc.)
- a call to `sax_which_data` to get the file type (packet type) used for input
- a call to `sax_pktcap_load` to load the content of the entire packetcap record related to the selected packet
- a call to `sax_acc_preload`
- calls to routines like `sax_acc_select`, `sax_acc_range` and `sax_acc_other_range` to select the quantities and the limits for the accumulation

Then
- the first telemetry file of the chain (selected via the `concatenate` program) is opened.
- memory is allocated for the data structure to be created (when not using static memory)
- if necessary the data file is initialized
- the accumulation loop is started with `sax_acc_loop`

Finally
- the output file is created (if not done before) and populated
- and the necessary header keywords are written

The `sax_acc_loop` routine descends in a series of library calls, and ultimately calls an increment routine (whose name receives as argument), which is included in the same source file as the main program.

All other library routines (the ones with `sax_` prefix) are in library `fotlib`.

## 2.3 Special arrangements for HK

For HK accumulation it is necessary to ask the user which parameter one wants (parameter mnemonics are described in section 5.2). A special file (PCF see section 5 and Appendix B) is used to keep track of parameter characteristics.
The current program handles a single HK parameter at a time (it is unlikely it will handle more parameters if they are present in different packet types).
The PCF content for the entry corresponding to the wished parameter is loaded in common by `sax_pcf_load` (equivalent to `sax_pktcap_load`), and the individual fields of the PCF are looked up by `sax_pcf_lookup` (equivalent to `pktcap_lookup`).

Usually the PCF for a given parameter points to a single packet. For MECS it might point to a "generic" packet (of which three separate forms exist on a unit basis) and this is the only case when `sax_which_data` is called.

The select and range dialogue is replaced by an ad-hoc routine `sax_acc_hkrange`.
For the rest the program is similar to normal accumulation programs.

# 3 Library routine description

## 3.1 sax_which_data(packet,accumulationtype)

returns the type of packet (a code like `'M1DIR002'`) suitable for the requested accumulation type (a code like `'Image'`).

To do this opens the observation directory for the first observation in the chain, and verifies if there is at least one type of packet compatible with the accumulation (this is an hard-coded list). There are three possibilities :

No packets are compatible or exist. The routine prints a warning and exit (letting the program fail later).
Only one type of packets exists, the routine returns it and the program continues.
More than one exists (this will e.g. always be the case for simultaneous operation of MECS units), and the user is asked which one to select.

NB: Error checking is very rough.
There is no support yet for slew data selection

## 3.2 sax_pktcap_load(packet)

loads the entire packetcap entry for given packet type (a code like `'M1DIR002'`) into a common block.

This is a wrapper around the general `pktcap_load(satellite,instrument,mode,packet)` routine, which constructs the argument list according to hardcoded SAX conventions.
The lower level routine (in `xaslib`) locates and opens the packetcap file and interprets its contents (concatenating all tc entries as necessary in a single string)
At the moment the maximum length of a concatenated entry is 1024 characters.
Another limitation (which reflects on all accumulation programs) is that the common block `PKCOMMON` supports a single packet type at a time

## 3.3 sax_acc_preload(ndimens,nformat,xstuff,stuff)

determines where the "interesting" quantities for accumulation (e.g. X,Y for images, PHA for spectra, time for time profiles) are located in the input packets and pre-loads.in a common block some values of packetcap fields. It also does some other preliminary settings (like determining whether byteswap is required, see also 4.2). The arguments are as follow :

`ndimens` is preset to 2 for images, 1 for spectra and time profiles (including HK), 0 for photon files
`nformat` is preset to 2 for images, 1 for spectra, 3 for time profiles (and HK), 0 for photon files
`xstuff` is an array of strings with the names of the interesting quantities
`stuff` is an array of strings, returning the names of all quantities present in the packets

Note that argument `xstuff` are no longer in full use since the introduction of `sax_acc_select`.

The relevant packtecap fields accessed are `nf`, `bt`, `i2` and `i4`, `du`, plus `f`*n*, `s`*n* (and `u`*n* for WFCs) for the interesting quantities in direct mode, `sf`, `ni` and `ff` for indirect mode data, `sf` for HK data

### 3.4 sax_acc_select(ndimens,nformat,xstuff,stuff,ierr)

is the one which fills `xstuff` (arguments are as per previous call, plus an error code ierr) handling the necessary dialogue with the user, asking which are the interesting quantities when appropriate (also not all programs call it)

### 3.5 sax_acc_range(ndimens,nformat,xstuff,stuff)

is the one which handles the dialogue for the limits (and zoom/binning factor) on the interesting quantities and fills a common block appropriately. For photon files asks whether (yes or no) a quantity is considered interesting (i.e. included in output)

At the moment the default min and max of the range are 0 to $2^n$-1 where the nominal size of a field is n bits. In a future special packetcap fields (foreseen but not implemented) might be used to restrict this range when not all the width is actually used.

### 3.6 sax_acc_other_range(ndimens,nformat,xstuff,stuff)

is the one which handles the dialogue for the limits on all quantities for which it has not been handled above. It is very similar to the previous one.

Here and in previous routines, the current dialogue for time (in whatever raw units it is in the file) must be replaced with a better dedicated dialogue asking for limits in UT (using OBT-UTC conversion) and time bin size in real seconds (instead of a binning factor).
Any arrangement existing (or not existing) for ground calibration purposes will be removed from the public distribution.

### 3.7 sax_acc_loop(routine)

looks up in packetcap the secondary type (`st` field) and calls the appropriate second level routine. These routines are currently called : .

| | |
|---|---|
| `sax_acc_bt_1` | for basic type 1 (direct mode data) |
| `sax_acc_bt_2` | for basic type 2 (indirect mode data) |
| `sax_acc_bt_3` | for basic type 3 (HK data) |

The argument routine is the name of a program-specific subroutine handling the accumulation in the program-specific structure (image, spectrum etc.)

### 3.8 sax_acc_bt*(ist,units,routine)

for each basic type, dispatches to the appropriate third level routine according to `ist` and `units` (the content of the packetcap `st` and `du` fields, available from previous lookups). The possible routines are currently the following : .

| | |
|---|---|
| `sax_acc_b1s1_y` | basic type 1, secondary type 1 (Laben NFI), FOT format (y=data in bytes) |
| `sax_acc_b1s1_i` | basic type 1, secondary type 1 (Laben NFI), ground format (i=data in bits) |
| `sax_acc_b1s2_y` | basic type 1, secondary type 2 (SRON WFC), FOT format (y=data in bytes) |
| `sax_acc_b1s2_i` | basic type 1, secondary type 2 (SRON WFC), ground format (i=data in bits) |

No secondary type yet defined for LECS

| | |
|---|---|
| `sax_acc_b2s1_y` | basic type 2, secondary type 1 only case currently defined |

`sax_acc_b3s1` basic type 3, secondary type 1 (Laben NFI BTB ratemeters)
`sax_acc_b3s2` basic type 3, secondary type 2 (Alenia IB ground format)
Other secondary types to be added for FOT common ASCII files, and spacecraft HK

The routines for ground data are not intended for public distribution (dummy `RETURN-END` routines shall be distributed), nor shall be implemented at all sites.

### 3.9 sax_acc_b*s*(routine)

are the work-horse doing packet reading, and are in general arranged as follows:

Do packetcap lookup as required for the given accumulation, and namely :

| | |
|---|---|
| `sax_acc_b1s1_y` | `hl,ni,ve,vl,vs` |
| `sax_acc_b1s1_i` | `hl,ni,ip,vl,vs` |
| `sax_acc_b1s2_y` | `hl,ni,ng,eg,lg,ve,vl,vs, u`*n*`,o`*n*`,m`*n*`,d`*n* |
| `sax_acc_b1s2_i` | dummy routine |
| `sax_acc_b2s1_y` | `hl`,`ve` or `ip,vl,vs` (plus time related info in future) |
| `sax_acc_b3s1` | `hl,ni,ve` or `ip,vl,vs,tl,nf` |
| `sax_acc_b3s2` | `hl,ni,tl` |

Enter a packet reading loop, which reads in turn each packet in the current observation
the loop can be terminated graciously with a control-C (`z_break` routine)
when all packets have been read, if there is another observation in the chain, restart loop
Within the loop, for each packet extract necessary header information

Then do a loop within loop on events or spectra or HK samples
For each event do a loop on fields, and extract them. If the data are in range event is accepted, therefore the increment `routine` can be called.
For each channel in spectrum extract them and call `routine` (which will extract the subset in range).
For each sample in time range call `routine`.

See comment in 4.2 below about byte swap.

### 3.10 increment_routines

are the routines which actually create the wished data structure. They have no arguments and use two common blocks, one receiving data from the `sax_acc_b*s*` routine, and one (usually called `LOCAL`) accessing the data structure in the main program.

For direct mode data the event fields are in array `ACCUMCOMMON_J`
For indirect mode data the spectrum is are in array `ACCUMINDIR_HISTOGRAM`
For HK data array `ACCUMCOMMON_J` contains the time and parameter value of one sample

The increment operation is :

for images increment by one the appropriate pixel (direct mode only so far)
for spectra increment by one (direct mode) or by channel content (indirect mode) the appropriate bin
for time profiles increment by one appropriate timebin
for photon files extract and write out relevant fields for current event
for HK increment the n-th bin by the value of the parameter (for binning factor greater than 1, the average is made later when outputting the time profile)
for special program `dtaccum` output one or two timebins (the first of two is relevant to a "packet lost" interval if any).
for special program `dtover` increment a running statistics (terminated in main)

### 3.11 pktcap_lookup(field,itype,ival,strval,found)

is the `xaslib` routine looking up a given `field` in the currently loaded (see `pktcap_load` above) packetcap entry. It returns `itype` (0=boolean, 1=number, 2=string) and the appropriate value in `ival` (number) or `strval` (string). `found` is `.TRUE.` if field is present (for boolean implies true value)

### 3.12 sax_acc_open_tlm(lu,packet)

connects to logical unit `lu` the file containing wished `packet` type for current observation in chain..

It does a packetcap lookup for the `pl` field (record length). It opens the `obsdir` for current observation and gets the number of record for the data file of wished `packet` type, then constructs its name using the `rootname` returned by `sax_open_dir` and opens the data file.

### 3.13 sax_open_dir(lu,nobs,rootname)

finds a free logical unit `lu` and connects to it the observation directory for current observation in chain (if `nobs` is `1` forces the first observation in chain). Returns also a `rootname` for data files.

Retrieves the global variable `SLEW` and construct the rootname for data files according to the naming convention described in `DAWG-REP.3.0/94`, the opens the `obsdir` file for the current observation.
In future could also handle `instdir` and `expconf` files perhaps.
Contains also provisions for ground BTB data (according to the MECS naming convention)

### 3.14 sax_acc_hkrange(ndimens,nformat,xstuff,stuff,packet)

is the dedicated routine (currently in `saxhkaccum` source) for select and range dialogue. It is similar to the other `fotlib` select and range routines.

It asks for the time limits, using as default the times of the first and last packet. Later it shall read from the `instdir` or `obsdir`. Also the dialogue on bin size shall be improved (as for the other range routines).

# 4 PACKETCAP FILES.

The packetcap files for instrument *eeee* are located in `$XASTOP/calib/sax/eeee`.
They are named `sax`*eeeetttt*`.packetcap`.
The type *tttt* may assume so far the values `dir`, `indir`, `hk`.

## 4.1 Format of packetcap files

A packetcap file can contain :

comment lines (preceded by #)
blank lines
data lines in the format of `termcap`

In particular each *entry* is composed by one or more data lines, as follows :

The first (or only) data line starts with a packet name (preferably in upper case), terminated with a pipe (`|`).
The field between the pipe and the next semicolon (`:`) is a descriptive comment.
All remaining *fields* are fully enclosed between semicolons (`:`) and are of the three following types :

| | |
|---|---|
| string | in the format `nn=value` |
| number | in the format `nn#value` |
| boolean | in the format `nn` (present for true) or nothing (absent for false) |

where `nn` is a two character code.

All data lines but the last are terminated by a backslash (\) as last character, after a semicolon. Continuation lines start with at least one blank, followed by a semicolon introducing a new field.

## 4.2 Content of packetcap files.

There is an entry for each packet with a different name.

When packets with different names have the same layout (like for the packets for different MECS units) the entry shall include only the name and descriptive comment, and make reference with a `tc` field to another "generic" entry. The name of the generic entry shall clearly identify it. E.g. the `M1DIR002`, `M2DIR002` and `M3DIR002` all refer to generic `MxDIR002`.

Packet with different layout shall not have the same name. The case of FOT and ground calibration packets with the same name and different layout can be solved in two ways :

give to the ground packets a different name (this is the solution used for the MECS, where e.g. packet `MxDIPyyy` is the ground (Laben BTB) version of FOT packet `MxDIRyyy`).
These entries should be removed before public distribution of the packetcap file, but are harmless if they remain present and unused. This also allows the same software to process flight and ground data.

create a packetcap for ground data, and later replace it with another one for FOT data.

Packets with very similar layout may have an entry containing only the fields necessary to specify the differences with respect to a common template, and then a `tc` field pointing to this template entry, with all other fields. This might be the case of :

packets which were different in data layout at telemetry level (e.g. different number of bits for time), but are almost equal after FOT reformatting (like `MxDIR002` and `MxDIR003`, which after FOT reformatting differ only for the record length, and the number of events/packet, but have events of the same layout). The name of the template entry shall clearly identify it (e.g. `MxDIR_23`).

packets in which the header has a common format (in this case e.g. all MECS Direct packets may refer to a template `MxDIRHDR`). The name of the template entry shall identify it (e.g. using the letters `HDR`).

another similar case (combining the two above) is for indirect mode spectra. Since FOT reformatting produces files with one spectrum per packet (irrespective of the original packet layout with one,two,four or half) a single `FOTSPEC` template can describe the header and data characteristics, while individual packet entries contain only items like number of channels etc.

The above mechanism of `tc` references can be nested (e.g. `M1DIR002` points to `MxDIR002` points to `MxDIR_23` points to `MxDIRHDR`).

Each packet must have defined a basic type `bt` and secondary type `st` and a packet length `pl`.
Each packet must also specify if its integer data are little or big endian separately for `i2` and `i4`.
The usage of the `i2` and `i4` fields for on-the-fly byteswap (when the internal representation is different from the one of the target machine) is TBD. Code for this exists, in particular for the special case represented by ground data (which are to be swapped as it were entirely `INTEGER*2`), but it might be wiser to do this once forever as the last step of FOT filing (this is unnecessary on DEC systems).

Each packet header must have defined its length `hl`, the number of fields in the header `nh`, and `nh` couples `hn` `zn` with the names and size of the field.

A boolean field `ve` indicates the presence of a valid event counter at location `vl` and size `vs` within the header.
Ground data use boolean field `ip` to indicate presence of invalid data pointer at same location.
The location of a "special" time field (usually start time) in the header is given by `tl`, its size by `ts` and its resolution by `tr`. Other fields related to time are not yet implemented. Most programs currently assume that other time fields follow *immediately after* the special one.

Direct packets must have defined `ni` as the number of events/packet, `nf` as the number of fields/event, and `nf` couples `fn sn` with the names and size of the field. Sizes are in bytes or bits according to `du`.

Indirect packets must have defined `nf` as the number of channel/spectrum, `sf` as the size of a channel in bytes or bits according to `du`, `ni` as the number of spectra/packet, and `ff` as the name of the quantity in the spectrum.

HK packets must have defined `ni` as the number of sample/packet, `nf` as the number of parameter/sample, and `sf` as the size of a sample in `du` units. In addition `cy` must be the duration of a cycle (packet) in seconds.

# 5 PARAMETER CHARACTERISTICS FILES (PCF)

The PCF for instrument *eeee* is located in `$XASTOP/calib/sax/eeee`.
It is named *eeee*`.pcf`.

## 5.1 Format of PCF files

It is identical to the one of packetcap files, with the only semantic difference that there is one entry per HK parameter (and not per packet). Of course the field names are also different.

## 5.2 Content of PCF files

There is an entry for each HK parameter.

The parameter names shall be less or equal to 8 characters, and are at discretion of instrument teams. They should be as mnemonic as possible. The following guidelines were adopted *provisionally* for the MECS :

> Ratemeter names start with `VAL` or `REJ` for valid event or rejected event counters, and for the rest are mnemonic. There is no need to distinguish MECS units, since the same parameters is present in packets separated per unit.
>
> Parameters of the same meaning, present in the same packet for different units, have a numeric suffix (thus `HVPMT1`, `HVPMT2` and `HVPMT3`, or `XELOM1`, `XELOM2`, `XELOM3` in which case the suffix is `Mn`).
>
> Voltages and temperatures start with the prefix HV or TEMP.
>
> Threshold setting names include a code, specifying analog or digital, a code of the quantity, a code `LO` or `HI` (for low or high thresholds). Thus `DYLOM1` is Digital Y LOw threshold for M1.
>
> The names of the parameters shall not be equal to the names of the fields in the experiment configuration files. These are currently provisionally defined in fax "Final proposal for experiment configuration files on FOT" by L.Chiappetti - IFCTR - 1.0 - 15 Dec 94. To keep them different one can reshuffle the components in names, thus `HVPMT1` is an HK "analog readout" of the quantity commanded to a digital level by `M1PMTHV`.

For a typical ratemeter one must define `pk` as the ratemeter packet type, `co` as the number of sample/packet (same as `ni` in packetcap), `le` as the size of a sample (usually this will be `16` bits), `of` as the offset (same as `sf` in packetcap), `po` is the position of the first sample, and `un` is usually `Counts`.

For other parameters in Alenia blocks `co` is `1` (in FOT packets it will be multiplied by the number of blocks of given type in packet) except for ratemeters which have `co 2`. `un` is `Volt` for all voltages, and so far also for temperatures. It is `Channel` or `Pixel` for thresholds, and is undefined for enumerative parameters. For voltages and temperatures `zp` and `sp` (according to Laben SUM) can be used to define the scale conversion from digital units to measured values (the voltage to temperature conversion is TBD).

## APPENDIX A - PACKETCAP FIELDS

This table reports all presently defined fields in alphabetic order.
Entries in *italics* are not yet implemented (irrespective of the fact whether present in some files or not present at all), entries in **boldface** are there for WFC only.

The field `cy` was added to support HK data, and additional interpretation of some other fields provided to support indirect modes and HK data.

| Field | Type | Description |
|---|---|---|
| "-" | n/a | packet name (terminated by pipe or colon separator) |
| "--" | n/a | packet description comment (optional, follows packet name, terminated by colon) |
| bt | number | Basic Type: so far 1=direct (event-by-event); 2=indirect spectral, 3=HK |
| cy | number | CYcle of a sample in sec (for bt#3 only) |
| **d<n>** | **number** | **if bt#1:st#2 might be present to indicate field content must be divided by this value** |
| du | string | Data Unit : all data lengths are in "bits" or "bytes" |
| **eg** | **number** | **Events per Group (bt#1:st#2) number of events in all groups but last** |
| f<n> | string | Field <n> (with n a number) name; fields listed in the order in which they appear in one event |
| ff | string | FFields name, whenever all fields have the same name (e.g. bt#2, channels of a spectrum) |
| h<n> | string | Header field <n> name; fields listed in the order in which they appear in the header. Unused fields can be combined into a single one. |
| hl | number | packet Header Length in bytes (includes any kind of headers) |
| i2 | string | I*2 endianness (LE for little endian, VMS,DEC; BE for big endian, standard 2-complement machines) |
| i4 | string | I*4 endianness (LE for little endian, VMS,DEC; BE for big endian, standard 2-complement machines) |
| ip | boolean | if set Invalid data Pointer is present in header (exclusive with ve) |
| **lg** | **number** | **Last Group (bt#1:st#2) number of events in last group (0 if all groups are equal)** |
| **m<n>** | **number** | **if bt#1:st#2 might be present to indicate field content must be taken modulo this value** |
| *m<n>* | *number* | *Minimum legal value for field <n>; (optional, default 0)* |
| *M<n>* | *number* | *MAXimum legal value for field <n>; (optional, default max allowed by binary field size)* |
| *mf* | *number* | *Minimum legal value for all fields ff; (optional, default 0)* |
| *Mf* | *number* | *MAXimum legal value for all fields ff; (optional, default max allowed by binary field size)* |
| nf | number | Number of Fields (for bt#1 number of fields present for each event) |
| nf | number | Number of Fields for bt#2 number of channels in each spectrum) |
| nf | number | Number of Fields for bt#3 number of parameters in each sample) |
| **ng** | **number** | **Number of Groups (bt#1:st#2) excluding last group if shorter** |
| nh | number | Number of Header fields (if hl .ne. 0) |
| ni | number | Number of Items (for bt#1 number of events/packet |
| ni | number | Number of Items (for bt#2 number of spectra/packet) |
| ni | number | Number of Items (for bt#3 number of samples/packet i.e. commutation) |

| | | |
|---|---|---|
| **o<n>** | **number** | **Offset of field <n> in group (used if bt#1:st#2:u<n>) in units TBV (E.G +du from start, -du from end of group; however how to handle last group ?** |
| pl | number | Packet Length in bytes |
| s<n> | number | Size of field <n> in du units |
| **s<n>** | **number** | **if bt#1:st#2 may be set to zero if field is packed jointly with previous field (previous requires d<n>, this requires m<n>)** |
| sf | number | Size of all Fields ff in du units (for bt#2 this is the byte width of a channel) |
| sf | number | Size of all Fields in du units (for bt#3 this is the byte width of a sample) |
| st | number | Secondary Type (for bt#1): 1="Laben" format (sequence of events)<br>2="SRON" format (sequence of groups of events, last may be shorter) |
| st | number | Secondary Type (for bt#2): 1 identically |
| st | number | Secondary Type (for bt#3): 1=ratemeter, 2=spacecraft payload HK *other TBD* |
| tc | string | reference to another packetcap entry (if present must be the last field in current entry) |
| tl | number | Time Location, location of a special time field in header (in bytes) |
| tm | *TBD* | *Time Mask: mask to correlate item time and header time and ultimately with UT; 0 means no mask needed* |
| tr | number | Time Resolution (expressed as base 2 logarithm, e.g. -14 is 2**-14 s) |
| ts | number | Time size, size of a special time field in header (in bytes) |
| tz | *number* | *Time Zero : start time for time counter, 0 means arbitrary, TBV* |
| **u<n>** | **boolean** | **if bt#1:st#2 and field is Unique (appears once in group) set this** |
| ve | boolean | if set Valid Event counter is present in header (exclusive with ip) |
| vl | number | Valid event counter Location in header (in bytes or du units TBD)<br>for convenience used also for invalid data pointer location |
| vs | number | Valid event counter Size (in bytes or du units TBD)<br>for convenience used also for invalid data pointer size |
| z<n> | number | siZe of header field <n> in du units |

## APPENDIX B - PCF FIELDS

| Field | Type | Description |
|---|---|---|
| "-" | n/a | parameter name (terminated by pipe or colon separator) |
| "--" | n/a | parameter description comment (optional, follows packet name, terminated by colon) |
| co | number | COmmutation of parameter (number of times recurs in packet) |
| le | number | LEngth of parameter field in bits |
| of | number | OFfset in bytes between subsequent recurrences (if co is not 1) |
| pk | string | PacKet : is the packet type where the parameter is located |
| po | number | Parameter Offset (or POsition) of parameter within packet in bytes<br>If co is not 1 is position of first occurrence |
| sp | string | SteP size : is a real number encoded as a string, and corresponds to the step size in physical units (or engineering units[1] ) of one digital step |
| un | string | UNit : is the string to annotate the y-axis when plotting the parameter, i.e. the name of the units in which the physical quantity is measured |
| zp | number | Zero Point : is the value in physical units (or engineering units[1]) corresponding to a digital value of zero, |

Note 1 : According to R.Jungk (in his history of atomic scientists) the mathematician Felix Klein was very fond of interdisciplinary meetings, and had to attend one at some German Ingenieurverein. Unfortunately he was sick, and sent his colleague David Hilbert in his stead. Hilbert began : "I have being told to talk against the fact that there are contrasts between scientists and engineers. This is not true : *in fact they have nothing to do one with each other* !".