prova di nota

The need for unpacking of SAX science data and the applicability of FITS binary table format to SAX Final Observation Tapes.

A report to the SAX Ground Segment Scientific Working Group
prepared by:

L.Chiappetti, IFCTR

22 May 1991

Table of content

During the SAX Ground Segment meeting held in Bologna on March 21, the issue of FOT formats was discussed for a while, and a number of points were considerered [Ref. 1] :

a) whether we should consider different supports than 1/2 inch magtapes for distribution (DAT tapes, cassettes etc.)

b) whether, irrespectively of the support, we should relax the requirement of one tape per experiment per observing period

c) whether we should modify our baseline for FOT formats (as contained in the SDPUR document).

A full report on the discussion is beyond the scope of this note, which is prepared in fulfillment of an action assigned to me as a result of the discussion on item c. I remind that this is just a note representing the author's opinion : any decision shall be endorsed by the GSWG and communicated officially to ASI by the Ground Segment Project Scientist. I would however briefly summarize as follows :

a) the 1/2 inch tape remains so far the only STANDARD medium which can be read anywhere. The issue of other supports is deferred to a future when other media standards would be better consolidated (this could include also network transmission)

b) it is possible that the requirement could be relaxed if space allows (this could be handled at a technical meeting together with item c)

c) this item is covered by the present note, and should be discussed in a dedicated technical meeting once the Space Segment Prime Contractor has released information on the telemetry formats.

This note is organized as follows: in section 2 we remind about the current baseline (which foresees an expansion/unpacking); in section 3 we give some examples of how this baseline could be implemented, and compute the overheads; in section 4 we present the alternate solutions (no unpacking at all, or usage of FITS binary tables); in section 5 we illustrate the possible applications of FITS binary tables; finally in section 6 we draw some conclusions.
.

2. A summary of the current baseline

This section summarizes items already discussed in [Ref. 1].

The telemetry data will be in form of source packets. The details of their layout are not yet known. As indicative references we use [Ref.2] and [Ref.3]. This can be summarised as :

The content of the *PACKET HEADER* is dictated by the ESA standard, its content are relevant for sorting the data, but not for the scientific analysis.

The content of the *DATAFIELD HEADER* (eg. packet times) may have some interest for scientific analysis.

The *data field proper* is made of subfields all of equal length, plus a *spare part* at the end which is unused.

The subfields are individual *events* in the case of direct modes, and *spectra*, etc.in the case of indirect modes. Subfield length is by definition an integral number of bytes.

Direct mode event sub-subfields (like X,Y,E, etc.) can sometimes be bit fields other than 8 or 16 bytes, even if the entire event uses an integral number of bytes.

The source packets are encapsulated for transmission in the Transfer Frames and Virtual Channels as dictated by ESA standard. This implementation is transparent and of no concern for scientific analysis.

The format and organization of such data is of no concern for scientific analysis, and will be decided autonomously by Telespazio.

We refer to [Ref.1] for a summary of our main intentions in defining the RRD (*Reformatted Raw Data*).

So far we implicitly assumed that Raw means that the packets are unmodified, while Reformatted means they are extracted from the Transfer Frame and somehow sorted in files (see [Ref.1] for examples). However the implementation details have been left to Telespazio, and are transparent to scientific analysis.

The definition of the requirements for FOTs (and particularly for the "unpacking" which is the object of this note) is given explicitly in chapter 5 (pag. 47, section 5.2) of the SDPUR document, approved by the experimenters.

Such requirements were explicitly based on the Exosat FOT example, with some major differences (explicitly recalled in [Ref.1]) intended to avoid some of the main unfriendly and unpleasant aspects of the latter (from the point of view of the final user).

The following is an explanation of the meaning of such requirement in the framework of 2.1-2.4 (not yet defined at the time SDPUR was written):

a) PACKET HEADER : the packet header can be omitted from the packet written on the FOT

b) DATA FIELD HEADER : it has to be decided which fields have to be written to the FOT packet header, and how do they have to be edited (modified)*

c) DATA FIELD : the data field content is written in its totality to the FOT packet, and is subject to the "expansion" to 8-16-32 bits of fields of different length (further motivated in the next section).

d) spare area : this can be omitted from the FOT

This is the resulting FOT packet format:

The resulting FOT packet lengths after unpacking may be different for each packet type, as the

* This is consistent with Exosat usage (which did not implement the full ESA standard packets, but just a rough prototype): in fact only a reference time and a quality flag appeared in the FOT packet header, while all other information in the original packet header (packet id, checksum, etc.) was omitted.

amount of reformatting is different. The length will however be fixed for each FOT packet type.

In SDPUR it is also requested that the mentioned packet length is the *logical record length LRECL* of the corresponding FOT file, and that logical records be *blocked* in tape blocks of length multiple of LRECL (what is called an FB-format in IBM terminology). It was also requested to have one file per observation per data type.

The expansion of all "funny" (not 8 or 16) bit fields to bytes or words at FOT production time will avoid that the user has to do programming with bit fields, which is intrinsically not portable, prone to efficiency problems, and unpleasant for most astronomers. It will also ensure it is made once forever, and in the correct way. It will also allow the scientific Institutes to supply the users with standard, portable, accumulation software (some special features will be devised to handle BYTE quantities which are not standard in Fortran 77), which at the same time may remain efficient (in alternative bit field handling within our programs is possible using Fortran 77 intrinsic functions like IBTEST, IBSET, but they are sometimes much less efficient than dedicated assembler routines ... and we have lots of unpacking/conversions to do).

In this respect the tape space expansion was considered of much lesser importance w.r.t. the above advantages.

We attempt here to give some examples (inclusive of quantitative estimates) of the expansion needed in the case of SAX data (making so far reference to the modes as described in [Ref.2], and to the content of the data field header as given in [Ref.3]).

Note that one of the design goals is to have a FOT layout which could be dumped to disk into a natural format (one record per packet, same record length

as LRECL on tape) with
a simple system command (e.g. dd under Unix, MOVEFILE under IBM VM, etc.).

As suggested in 2.5.1 above most of the information in the headers is used at OCC to sort the various telemetry streams, and/or verify the data quality.

The data in the packet header do not need to be put anywhere on the FOT. Application Process ID (and additional mode fields in the data field header specifying the packet format) are used to sort packets by data type into separate FOT files. Source sequence count will be used at OCC only to verify the continuity of the data stream. Packet length information is also non relevant (the information on the FOT packet length will be in the observation directory).

The exact content of the data field header is so far not known in detail. The only information presented in [Ref.3] which is necessary to propagate in the FOT packet header are the *start* and *end times*.

It is likely that such fields (32 bits each) do not need expansion. However it might be desirable to supply the end user with times in a more handy unit than spacecraft clock (e.g. UT with some TBD time resolution). This might require some expansion (but no more than a factor 2), and should be discussed in a specific meeting.

As a rule, the spare part at the end of a packet shall not be put on the FOT. For the rest, we give below some specific examples about direct and indirect mode packets. We are unable to give specific indications about ratemeter packets until the detailed layout is known.

It should be noted that part of the discussions with the Space Segment prime contractor resulted in the fact that most event sub-subfields are already 8 or 16 bits.

The first example concerns *MECS direct mode 1* (diagnostic, see [Ref.2]). After the recent request to have position and burst length at 8 bits, the format is:

4 anodes, energy, X, Y, BL all with 8 bits
time with 18 bits
total length 11 bytes

It has to be noted that, as 24 bits are reserved for time, it would be possible (and has actually been suggested) to use the full 24 bits for time already on-board (out of the 32 available to the experiment).

In this case the only expansion would imply using 32 bits for time (putting the 18-24 bits in the least significant part, either with leading bits zero, or with leading bits derived from the part of the header full time with same resolution).

Each event will therefore require 12 bytes, with a 9% expansion.

A similar expansion for time (from 7 to 8 bytes; 14%) applies to the *MECS direct mode 2* (normal; energy, X,Y, BL, time).

In the further example of *MECS direct mode 7* one has:

energy, BL with 8 bits
time with 7 bits
in/out flag with 1 bit
total length 3 bytes

In this case one expands time at least to 8 bits,

and also the in/out flag as 8 bits. The resulting increment is from 3 to 4 bytes (33%). Note that one could have no increment by sorting data with different in/out flags in separate files.

Considering another example, for *PDS direct mode 1* (diagnostic), one has:

coincidence with 3 bits	(expand to 8)
PSA with 9 bits	(expand to 16)
unit id with 2 bits	(expand to 8)
energy with 10 bits	(expand to 16)
time with 16 bits	(unchanged)
total length 5 bytes	(become 7 bytes)

requiring an expansion of 40%.

For *PDS direct mode 3* one has instead a 75% expansion :

PSA with 9 bits	(expand to 16)
unit id with 2 bits	(expand to 8)
energy with 9 bits	(expand to 16)
time with 12 bits	(expand to 16)
total length 4 bytes	(become 7 bytes)

but for *PDS direct mode 4* one has just a 25% expansion:

PSA with 6 bits	(expand to 8)
unit id with 2 bits	(expand to 8)
energy with 8 bits	(unchanged)
time with 16 bits	(unchanged)

total length 4 bytes (become 5 bytes)

In the latter three cases some saving could be gained by sorting events by unit id.

In the case of indirect modes, all spectral and most timing modes already foresee 8-bit or 16-bit channels. Therefore no expansion whatsoever is required (in the case of HP-GSPC packets mixing different ESP and/or BSP spectra, it may be desirable to put them into separate FOT files).

The only peculiar case involves some high resolution timing modes, which use 4-bit wide bins. This modes are likely to be used seldom. Unpacking might not be strictly required. If it is (into 8-bit bins), it involves of course a 100% expansion (doubling).

The main topic for discussion was the need and extent of unpacking/expansion at FOT production level.

One argument suggested NO REFORMATTING at all (except perhaps for headers ?) could be better. Packets are copied straightforward to FOT. The following motivations were put forward: first, the number of changes from the original layout is less, therefore *the possibility of errors is diminished* (the same argument used to minimise *reformatting* at RRD level); second, no unpacking may simplify the software development *from the point of view of hardware groups* (in case they intend to re-use the SCOE software and could in this case use the same routines to access telemetry packets). Incidental advantages of this possibility are : less tape space used (also less disk space after tape filing), and FOT production is simplified (it is essentially a plain copy of selected sections of RRD to tape).

Though the previous arguments are perfectly

reasonable from the point of view of the experimenters, my personal opinion is that such simplification of the current baseline shall not occur. Whatever advantage is gained from the part of the hardware groups, will reflect either as a *disadvantage on the general observer* (he will have to do all bit-field-programming himself) and also as a disadvantage on *everybody including the hardware institutes* (as performance on one hand, as we will have to do the unpacking all the times as part of each accumulation program again; and also as additional effort on software development, since the unpacking will be included in the scientific accumulation s/w and not in Telespazio s/w).

In order to meet the "friendliness versus general observers" and "portability" issues, a proposed solution is to adopt a STANDARD reformatting, based on *"binary tables" FITS.* This is the main topic of the remainder of this section.

The best account of what *currently* FITS is is given by [Ref.4] issued by the (recently established) NASA FITS office (the next release of this document will be available electronically). The above document provides a good uniform introduction to the basics and philosophy of FITS (in this sense it is better than the four standard reference papers on *Astronomy and Astrophysics Supplement* (refs. in [Ref.4]). It also describes official FITS (that is, endorsed by the IAU). The only main addition to the known standard which has been officially endorsed so far is the use of IEEE floating point data in FITS images.

The above document includes some chapters only in part (as specified in its Table of Content). *Binary table FITS* is described in section 5.2 of [Ref.4]. It shall be noted (see 4.2 below) that it has not yet been officially endorsed by IAU.

The basic philosophy of FITS, and the other FITS formats are not recalled here, and one is referred to [Ref.4] (or [Ref.1] for a short account).

FITS datasets consists of an header and a data area. The header is structured in 80-byte ASCII card images, in the form "keyword=value", with a fixed format.

The data area for binary tables is binary data (2-complement integer, or IEEE floating point; see [Ref.4], 5.2.3), including table "columns" in the order defined by the header. Each "column" may actually be an array (so called "3D" tables).

Header and data area are packed in 2880-byte records. These records may then be blocked up to a factor of 10 on the tape.

4.2 Approval status of FITS binary tables

The information provided here were supplied courtesy of Dr. Preben Grosbol of ESO Garching (Chairman of the IAU FITS Working Group), and of Dr. B.Schlesinger (NASA FITS Support Office). I would add here that in the course of the correspondence I have been discouraged to use FITS for very mission-specific telemetry data.

First of all, FITS binary tables are not yet an IAU standard (therefore the information supplied in [Ref.4] is subject to change, and in particular it is foreseen that the official name of the extension will change).

However the definition of the binary table format, which has been going on for some time, is virtually finished. Many major astronomical packages are in process of adopting it, or use it unofficially. A Fortran callable library to write FITS data (including binary tables) is going to be put soon in the public domain by NASA GSFC [Ref.5].

The formal process of approval by IAU involves motions by the Regional FITS Groups (the European one has met in April 91), and a formal test of

exchange between two independent sites. This is likely to occur in summer or autumn 91, and a final endorsement by IAU in late 91-early 92.

As every FITS dataset, a FITS binary table needs one full *FITS header* record (2880-byte, blank-padded) with the following mandatory keywords:

```
SIMPLE  = TRUE
BITPIX  = 8
NAXIS   = 0
END
```

A binary table includes next one or more *extension header* records, with the following keywords (mandatory unless otherwise specified):

```
XTENSION= A3DTABLE      (the name will change)
BITPIX  = 8
NAXIS   = 2
NAXIS1  = bytes in 1 row
NAXIS2  = no. of rows
PCOUNT  = 0
GCOUNT  = 1
TFIELDS = no. of columns
TFORM1  = rT (format of column 1)
...
TFORMn  = rT (format of column n)
TTYPE1  = label for column 1 (optional)
...
TTYPEn  = label for column n (optional)
```

As one FITS record includes 36 keywords, it is likely that one extension header record is sufficient for SAX purposes.

This makes a total overhead of 2x2880 bytes of headers per FITS file.

We give here below possible *examples* of application of FITS binary table formats (data area) to some *representative* SAX cases.

In the case of direct modes, the reformatting needed to put event data in format of a binary table (which should be immediately usable as a photon list) includes:

a) the table has as many rows as events in the *file*

b) each event sub-subfield (energy, X,Y, etc.) is a column

c) all sub-subfields which are more than 8 and less than 16 bits (e.g. PDS Energy, 10 bits; or PDS PSA; 9 bits) are naturally expanded as a single 16-bit integer (TFORMn=1I).

d) all sub-subfields longer than 16 bits could be represented as 32-bit integers (TFORMn=1J). This is likely to be the case only for time.

e) time information will most likely need additional reformatting. In fact, as the information about single telmetry packets (including the complete time in the header) is lost, the full representation of time shall be associated to each event.

f) all 8-bit sub-subfields could be represented as characters (TFORMn=1A); see however also g below.

g) all sub-subfields shorter than 8 bits could be expanded to a byte and handled as in f. An alternate solution, applicable to all lengths of sub-subfields, is to use the bit-array representation. Note ([Ref.4] pag.34) that the actual field is anyhow expanded to an integral number of bytes with **trailing*** bits zero. In this case one has (examples) :

```
              PDS id     (2  bits)   TFORMn=2X
(occupying 1 byte)
              MECS energy            (8  bits)
TFORMn=8X     (occupying 1 byte)
              PDS energy             (10 bits)
TFORMn=10X    (occupying 2 bytes)
                etc.
```

* Usage of trailing bits instead of leading bits may be an annoyance.

It is immediately visible that the expansion (mandatorily required by FITS) is **identical** to the one required by our current baseline (see above 3.2.1).

A possible exception to that is that time requires more space, and also a processing which is not just pure extension (but involves pasting the most significant part of the header full time in front of the event fine time, or some other form of time normalization). This is is a more complex processing than the one foreseen in the baseline.

However, the bigger expansion required for time, and the relatively large header overhead is probably compensated by the omission of the packet (datafield) headers.

Indirect mode data are not naturally covered by FITS unless one adopts one of the following approaches.

One possibility is to define an *ad-hoc FITS extension,* that is to devise a "FITS wrapper", in which one has a FITS (full + extension) header, and then forces whatever telemetry into the data area. This is quite innatural, and gives no advantage whatsoever, as nobody will ever has a FITS reader for such a peculiar format. This possibility will therefore not be considered any further.

The second is to use a binary table format as a general format. For indirect mode data, the spectra are naturally already in a table form (each spectrum is one row with as many columns as PHA channels), and also the time profile data are naturally in a table form (the rows being time bins, and the column(s) being the individual energy band(s)). Spectra and time profile are just a different way of looking at a table (time-energy) with counts as content.

Unfortunately there is more in the packet than just spectra : the only useful information in the datafield header is however so far represented by the start and end time (TBV with the detailed

formats).Other information may be derived by the position of the spectrum in the packet (e.g. if four PDS spectra from the four units are packed together; or when HP-GSPC ESP1 and ESP2 are packed together, etc.). This information could be covered adding extra columns in front of the table.

For a single spectrum (with p channels of 8 bits each) one will have:

```
NAXIS1  = bytes in 1 row (p+8)
NAXIS2  = number of packets
TFIELDS = 3
TFORM1  = J
TFORM2  = J
TFORM3  = pA
TTYPE1  = "START TIME"
TTYPE2  = "END TIME"
TTYPE3  = "ENERGY SPECTRUM"
```

(If the spectrum channels are 16-bit wide, TFORM3=pI, and NAXIS1=2p+8. If the channels (bins) are 4-bit wide (as in the case of some high resolution time profiles), one may either expand them to bytes, or have TFORM3=4pX, NAXIS1 unchanged).

For the case of more spectra (e.g. 4 PDS spectra, with p 8-byte channels) in same packet one has instead:

```
NAXIS1  = bytes in 1 row (4p+8)
NAXIS2  = number of packets
TFIELDS = 6 (number of spectra + 2)
TFORM1  = J
TFORM2  = J
TFORM3  = pA
...
TFORM6  = pA
TTYPE1  = "START TIME"
TTYPE2  = "END TIME"
TTYPE3  = "UNIT 1 SPECTRUM"
...
TTYPE6  = "UNIT 4 SPECTRUM"
```

Note that (for what concerns original packing on-board), if one packet contains more spectra, they

may be spectra taken *at same time for different units* of same experiment (and in such case the latter arrangement is preferred); however if they are *just consecutive in time*, but originating from the same experiment, it might be better to split the packet into more table rows (each row containing one spectrum) and use the former arrangement. Finally, if the spectra in the same packet are altogether different (as in the case of HPGSPC mixing ESP and BSP) it could even be better to separate the different types in *separate FITS files*.

(A different way of associating packet headers with data would be to use a "group" arrangement, similar to the HST "GEIS" format, but this looks in my opinion quite awkward to handle).

As in the current baseline, the actual expansion factor for indirect mode data put in FITS tables is nil. Also the amount of packet reformatting is very similar.

The usage of FITS binary tables to store telemetry data is in line of principle possible, but seems not to offer any significant advantage w.r.t. to the current baseline.

The expansion factor (space needed) is very similar to the baseline approach.

Also the amount of processing looks quite similar.

On the other hand FITS data are packed into an "innatural" record length, which means more and more complex i/o operations to read the FITS file to disk into a natural format (ie. a table with as many records as rows, each record being as long as one row), or to read the FITS file in memory in case it is dumped straight to disk.

It is also unlikely that the format devised within FITS (section 4 above) will be general enough to be immediately accessible by FITS readers already available at a generic site.

Therefore my recommendation is not to adopt a FITS format for telemetry data to be written on SAX FOTs.

Concerning the fact whether we should perform expansion/unpacking when writing packets to FOTS logical records, I believe the arguments given above confirm this approach both in terms of opportunity (see 2.5.2) and not excessive overheads (see 3.2).

Therefore my recommendation is that we mantain the current baseline as described in SDPUR and further supplemented in the present note.