The event format in the ATLAS DAQ/EF prototype -1

ATL-DAQ-98-129
2 Nov 1998

Authors: C. Bee, O. Boyle, D. Francis, L. Mapelli, R. McLaren, G. Mornacchi, J. Petersen
Keywords:

Abstract

This note presents the event format to be used in the ATLAS DAQ/EF prototype -1. It covers the format of data input to the Read-Out Crates from the Read-Out Drivers and up to the input of the Event Filter.
The proposed format is also the first step in defining the final ATLAS event format.

NoteNumber : 050
Version : 1.5
Date : 15-10-98
Reference : http://atddoc.cern.ch/Atlas/Notes/050/Note050-1.html

Document Change Record

Table 1. Document Change Record.

1. Document Title: The event format in the ATLAS DAQ/EF prototype -1			
2. Document Reference Number			**DAQ Technical note number 050**
4. Issue	**5. Revision**	**6. Date**	**7. Reason for change**
1	*0*	*01 Apr '97*	*Birth.*
1	*1*	*07 Oct. '97*	*General update of all sections.*
1	*2*	*20 Oct. '97*	*Muon sub-detector IDs changed at the request of S. Falciano.*
1	*3*	*14-Aug. '98*	*Add an offset to ROD trailer that indicates the relative order of Data/status information. General clean-up. Appendix with an initial .header file and an appendix of an example use of the .header file.*
1	*4*	*05-Sept.'98*	*Redefined last word of ROD trailer. Comments on sub-detector ID's from Philippe Farthouat. Comments from Jorgen Petersen.*
1	*5*	*15-Oct. '98*	*Tidy-up ready for release as ATLAS note*

1 Introduction

1.1 Purpose of the document

This document describes the event format and its initial implementation in the ATLAS DAQ/EF prototype -1 [1]. The proposed format is also the first step in defining the final ATLAS event format.

1.2 Overview of document

In section 2 the requirements, function, purpose and a high-level description of the event format is given. In section 3 a detailed description is given. Section 4 presents a description of the format of a fragment received by the ROB from the ROD over a ROL and is aimed principally at ROD designers. In section 5 an initial implementation of the event format is presented. It covers issues such as: sub-detector IDs, framing information and transmission errors on the ROL.

1.3 Boundaries

This document relates to the format of data into and within the DataFlow system and input into the Event Filter system. The definition of an event format for events output from the Event Filter is not considered to be an issue of the DAQ/EF prototype -1. The framing information, necessary to ensure the correct transmission of data between applications *e.g.* ROD-to-ROB, is technology specific and therefore not part of the event format.

1.4 Definitions, acronyms and abbreviations

See reference [2].

2 General description

2.1 Requirements

This sub-section lists a set of requirements on the various components of the event format. The categories of requirements follow the guidelines given in [3]. Requirements containing the word *shall* are mandatory. Those containing the word *should* are strongly recommended, justification is needed if they are not followed. Sentences containing the word *may* are guidelines, no justification is required if they are not followed.

The event format shall fulfil the following requirements:

1. The event format *shall* allow the size of an event to increase or decrease depending on the specific data taking configuration.
2. There *shall* be no minimum or maximum event data size implied by the format.
3. The event format *should* provide information redundancy to allow self consistency checks of the event to be made. The consistency checks will be defined at a latter stage.
4. The event formatting information *shall* not exceed 20% of the typical full ATLAS event data size.

5. The event format *should* be modular.
6. The basic unit *should* be a fragment. Fragments are: data coming from a ROD, ROB or Read-Out Crate (ROC), all the data associated to a single sub-detector and the data input to the Event Filter.
7. The fragments *shall* have identical structure and differ only in size and detail.
8. The event format *should* facilitate the identification of fragments.
9. The event format *shall* provide an event header.
10. The event format *shall* provide the event identifier and trigger type within the event header.
11. The event format *shall* provide a means of identifying whether the event has been corrupted during transmission within the DataFlow *e.g*. DMA time-out, truncation etc.
12. The event format *shall* provide a means of identifying whether the event has been corrupted due to hardware problems *e.g.* a bit stuck.
13. The event format *should* not demand additional computation by the processes which build the event format *i.e.* it *should* use data structures already existing within the DataFlow applications.

2.2 Function and purpose

The event format defines the structure of the data at various stages within DAQ/EF prototype - 1 and allows elements of the DataFlow and processing tasks to access the data without resorting to the use of other resources *e.g.* databases. It defines how the data of a ROB or crate fragment or a sub-detector data may be directly accessed by processing tasks. In addition, it defines additional data that is added to the detector data, by elements of the DataFlow, allowing processing tasks to quickly identify the type and origin of event.

2.3 General format

The general format of a Full Event is shown in Figure 1. As can be seen it is built from fragments (see requirement 6. in section 2.1). A Full event is an aggregation of sub-detector fragments and each sub-detector fragment is an aggregation of ROC fragments. In turn, each ROC fragment is an aggregation of ROB fragments. Each of the latter map on to a single ROD fragment. In addition, depending on the architecture of the DataFlow system, a sub-detector fragment may be an aggregation of ROB fragments (not shown in Figure 1.). Each fragment has a single header associated to it, opposed to a combination of a header and a trailer. The header contains all the event formatting information.

The object diagram of the event format is shown in Figure 2. Referring to the latter, it can be seen that a Full Event is one or more Fragments. A Fragment may be associated to zero or more other Fragments. Each Fragment consists of a Header object and zero or more Data objects. A Header is an aggregation of Generic and Specific objects. The latter, Specific object, being a generalization of: Event, Sub-detector, ROC, ROB and ROD. Headers are invariant of sub-detectors. However, the details of the header in a ROB fragment may vary with respect to a header in a sub-detector fragment.

As can be seen from Figure 2. the proposed event format is modular and based on event fragments (see section 2.1). In addition all event fragments have the same structure, thus fulfilling requirement 7. (see section 2.1).

Figure 1. The general event format

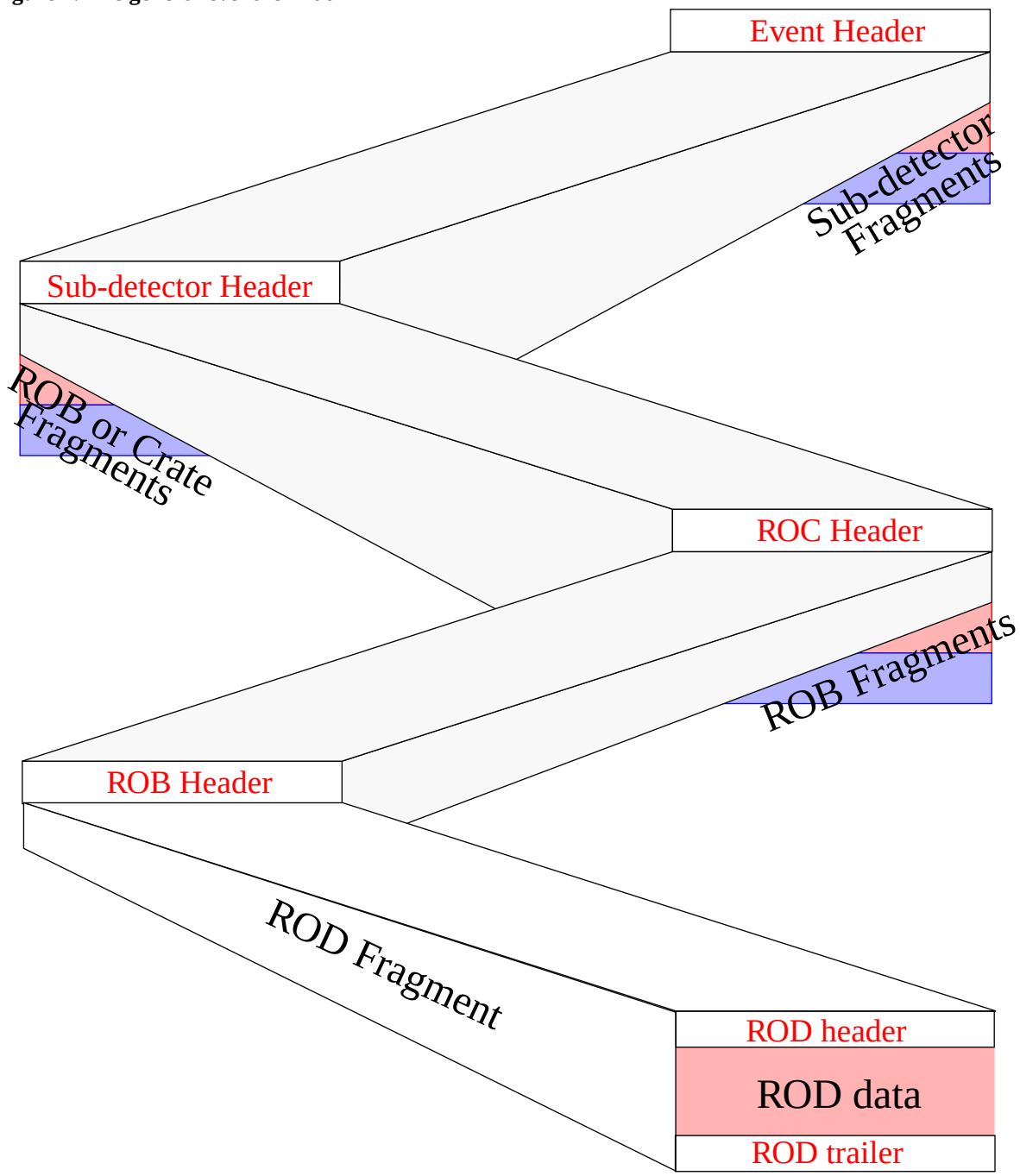
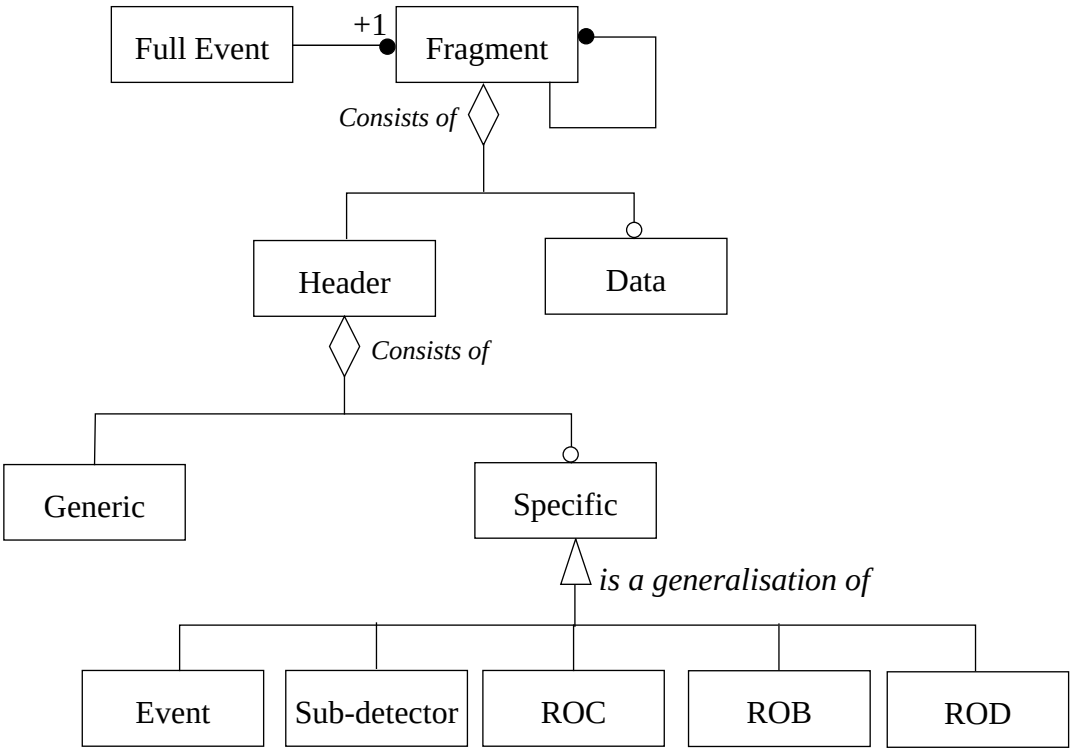


Figure 2. The object diagram of the event format.

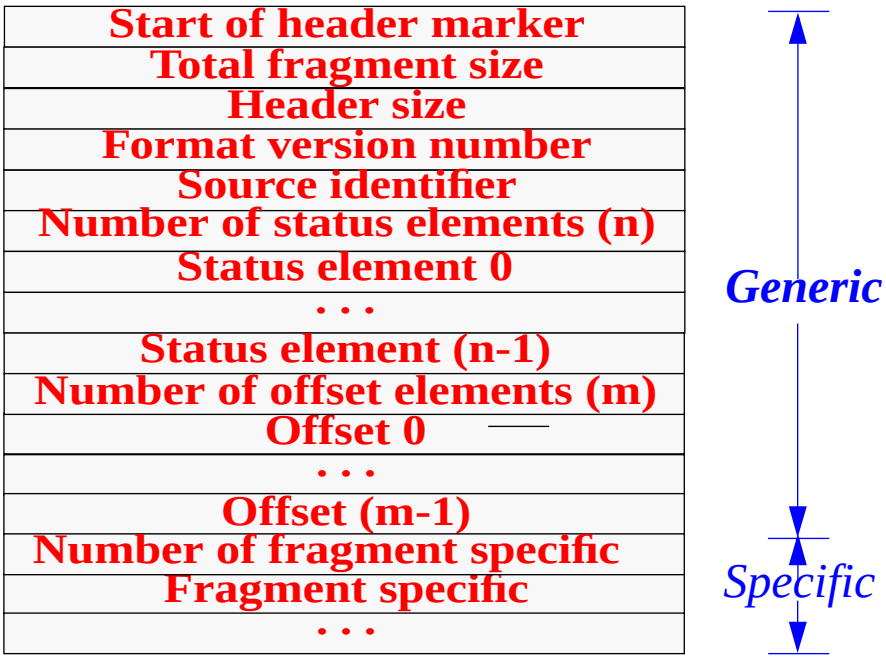


3 Header formats

3.1 The Header

The object Header depicted in Figure 2. is an aggregation of Generic and Specific parts. This is shown in Figure 3. The Generic part is the same for all event fragments. While the Specific part allows fragment specific information to be included in the header *e.g.* information generated specific to a sub-detector.

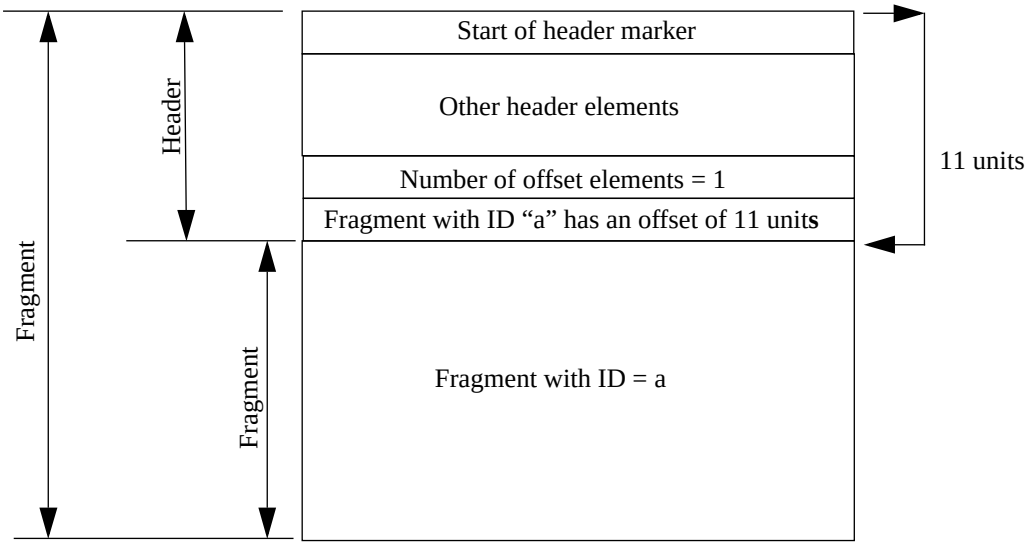
Figure 3. The fragment header.



The Generic part consists of the following elements:

1. *Start of header marker.* This marker indicates the start of a fragment header and is itself part of the header. Hence, it is the first word of a fragment. The value of this element will be unique for each type of fragment, but the structure shall be identical.
2. *Total fragment size*. This element indicates the total size of the fragment, including the Header.
3. *Header size*. The element indicates the total size of the Header.
4. *Format version number.* This element gives the format version of the fragment. For example, if this is a ROB fragment, it defines the format version of the ROB fragment. It allows the format of a ROB fragment to change independently of, for example, a sub-detector fragment. This element does not refer to the version of the detector data.
5. *Source identifier.* This element identifies the fragment. It consists of a sub-detector ID, a Read-Out Crate ID, an I/O module type and ID. The combination of these fields should allow the Source identifier to be unique across the whole of Atlas.The I/O Module type and ID refer to the I/O Module which builds and adds the header to the event fragment.
6. *Number of status elements.* This element gives the number of elements (Status words) following this element.
7. *Status element*. This element contains information about the status of the data within the fragment. The structure of this element is specific to the I/O module which builds the header and will therefore be defined during the detailed design of the DataFlow I/O modules.
8. *Number of offset elements*. This element indicates how many elements (Offset elements) follow this element.
9. *Offset*. This element contains an identifier and offset to a fragment contained within this fragment. The offset is relative to the start of this fragment. The use of this element is shown in Figure 4.

Figure 4. The use of the Offset element.



Following the Generic part of the header there is a fragment Specific section consisting of:

1. *Number of fragment specific*. An element which gives the number of elements following this element

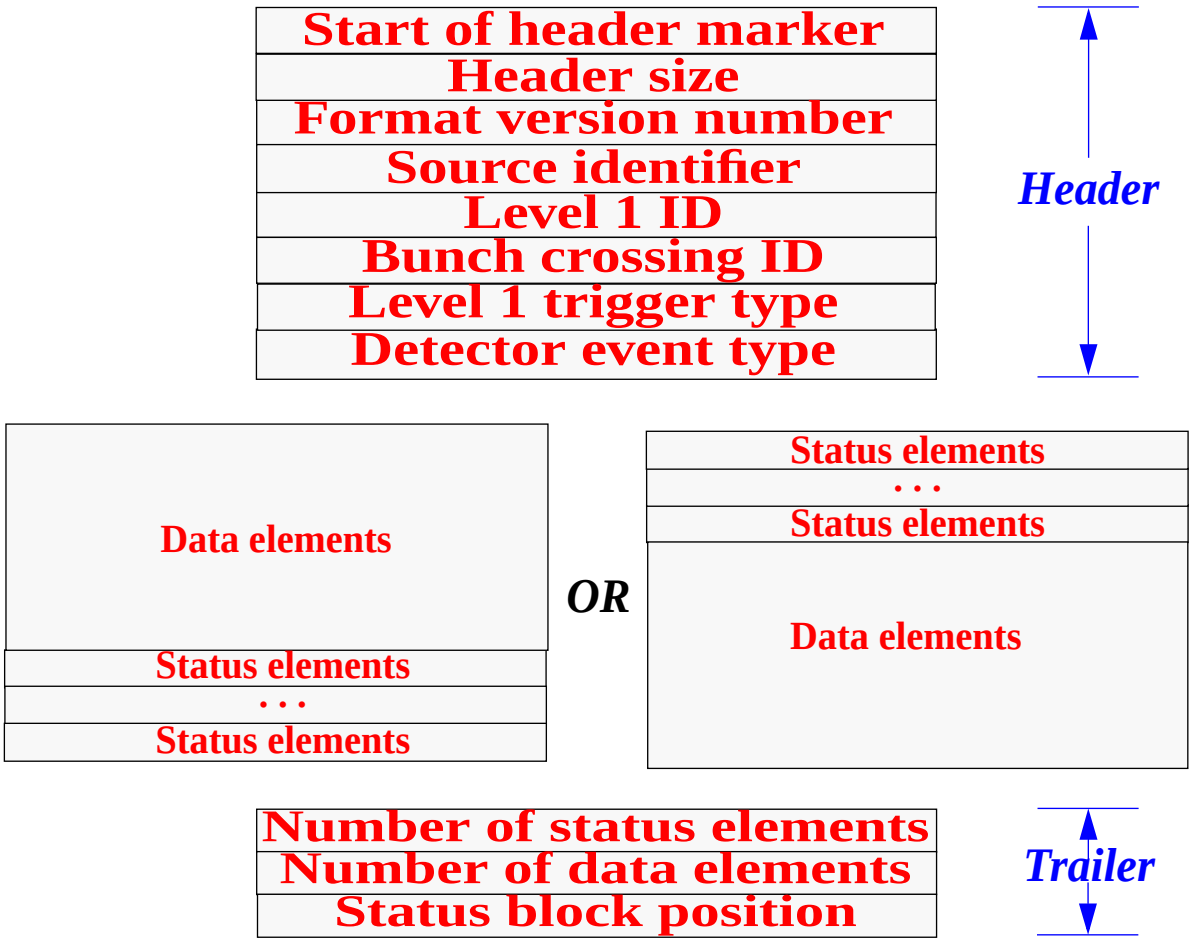
2. Zero or more elements, as specified in the previous element, which are specific to the type of fragment (ROB fragment, crate fragment etc.).

4 ROD data format

The definition of the format of the data transferred between the ROD and ROB applications must take into account factors such as: the data is formatted in hardware and not necessarily by programmable devices; the information within the header may influence component cost and ROD performance; the actual current understanding of ROD designs (a ROD may not buffer data).

Current designs of the ROD indicate that the data transferred from a ROD to a ROB should have both a Header and a Trailer as shown in Figure 5. The proposed header is derived from that presented in section 3.1 and the elements have the same meaning. The Trailer contains the Number of data elements, Number of Status elements and zero or more status elements. Each of these elements, again, have the same meaning as described in section 3.1. Some detector groups have voiced a preference for having the Status elements proceeding the Data elements. Instead of imposing an order, an additional element, *Status block position*, has been added to the trailer. This element solely defines the relative order of the Data and Status elements. A value of zero for this element indicates that the status block precedes the data block and a value of one indicates that the status block follows the data block. These two cases are shown in Figure 5. for reasons of clarity.

Figure 5. The ROD data format.



Within the header four elements are explicitly defined, these are:

- **Level 1 ID**: The event identifier generated by the Level 1 trigger system.
- **Bunch Crossing ID**: The bunch crossing identifier generated by the level 1 trigger system.
- **Level 1 Trigger Type**: The event type transmitted by the level 1 system.
- **Detector event type**: This element identifies an event which may have been generated by a sub-detector, independent of other sub-detectors and the ATLAS trigger systems.

In addition, the first status word must indicate the global status of the fragment. A non-zero value indicating that the fragment is corrupted (*e.g.* missing data, bit errors). The exact details of this element are still to be defined.

5 Initial implementation

This section presents an initial implementation of the event format described in the previous sections. This initial implementation is aimed at the requirements of DAQ/EF -1. It defines the Start of Header Markers, the Fragment IDs, the sub-detector IDs and the elements specific to the different types of fragments.

The ROD and ROB header are built by the ROD and ROB respectively. With respect to DAQ/EF prototype -1, the ROC header is built by the EBIF and the Sub-detector and Event Header are built by the SFI. It is not excluded that some information from the ROD header and trailer be copied into the ROB header and the ROD header and trailer be discarded.

This initial implementation takes the following points into account:

- *Bit fields* allow efficient use of memory and matching to hardware-defined data structures. However, they are likely to be non-portable. In this implementation of the event format bit fields are restricted to multiples of bytes and are avoided when possible. Efficient use of memory at this stage is not a primary issue.
- *Floating point types* are not used in this implementation, again due to portability issues.
- *Byte ordering*. Within the time-scale of DAQ/EF prototype -1 processors implementing different byte ordering schemes (*i.e.* Big or Little-endian) can be envisaged. The selection of endian ordering is only deemed necessary for the interpretation of the event format. The choice of endian ordering when exchanging data between processors is an issue to be addressed by the ROD designers, DataFlow and Event Filter systems. For the presentation of the initial implementation of the event format Big-endian ordering has been chosen, as it is used in established network header formats.
- *Alignment restrictions* are not deemed to be an issue for the event format. This issue is normally addressed by the compilers *i.e.* it ensures alignment on byte boundaries.

5.1 Start of Header Markers

Each fragment header begins with a Start of Header Marker. These markers fulfil requirements 7, 8 and 9 as described in section 2.1. The markers at each level of the event format are given in Table 2.

Table 2. Start of Header Markers.

Fragment Type	Header Marker
ROD	0xeeeeeeee
ROB	0xdddddddd
ROC	0xcccccccc
Sub-Detector	0xbbbbbbbb
Full Event	0xaaaaaaaa

5.2 Sub-Detector IDs

A proposal for sub-detector IDs is given in Table 3. Each major sub-detector (TRT, Pixel, etc.) has a unique major-id with a maximum of 16 possible minor-ids with which it can be sub-divided.

Table 3. Sub-detector IDs.

Detector		ID
Pixel		*0x10*
SCT	*Barrel*	*0x20*
	Left endcap	*0x21*
	Right endcap	*0x22*
TRT	*Barrel*	*0x30*
	Left endcap	*0x31*
	Right endcap	*0x32*
EM	*Barrel*	*0x40*
	Left endcap	*0x41*
	Right endcap	*0x42*
	Left forward	*0x43*
	Right forward	*0x44*
Hadronic Calo-rimeter	*Barrel*	*0x50*
	Left endcap	*0x51*
	Right endcap	*0x52*
	Left extended	*0x53*
	Right extended	*0x54*

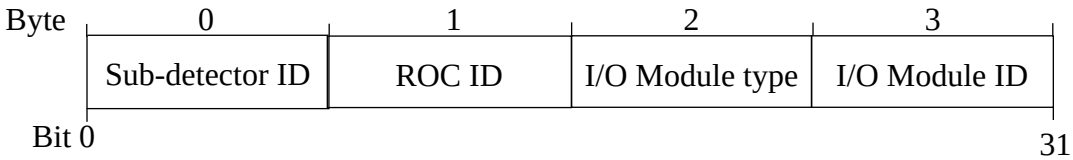
Detector		ID
Muon	*barrel trigger*	*0x60*
	barrel precision	*0x61*
	Left endcap trigger	*0x62*
	Right endcap trigger	*0x63*
	Left endcap precision	*0x64*
	Right endcap precision	*0x65*
	Left endcap CSC	*0x66*
	Right endcap CSC	*0x67*
Level 1	*Calorimeter front-end*	*0x70*
	Calorimeter e/γ	*0x71*
	Calorimeter jet & missing E_T	*0x72*
	CTP	*0x73*
	Muon Interface	*0x74*
Level 2		*0x80*
Event Filter		*0x90*

5.3 Total fragment and Header size

These elements are each 32-bit integers and their values give the total size of the fragment and the size of the fragment header in units of bytes.

5.4 Source IDs

The structure of the Source ID, as shown below, consists of four byte fields. The combination



of these four fields allows the Source ID to be unique across all sub-detectors. The possible values of the Sub-detector ID are defined in section 5.2. The values that may be assigned to the Crate ID, Module type and ID will be defined during the implementation of the DataFlow.

Note that this implementation restricts the range of values allowed. On the time-scale of DAQ/ EF -1 this is not an issue, however, the implementation of this element will have to be reviewed to remove this restriction.

5.5 Format Version Number

This element is a 32-bit integer. The value is the header version number for this type of fragment. More specifically, if this is a fragment of type ROB it defines the version of the ROB fragment header. It does not give the version number of the detector data.

5.6 Number of Status elements

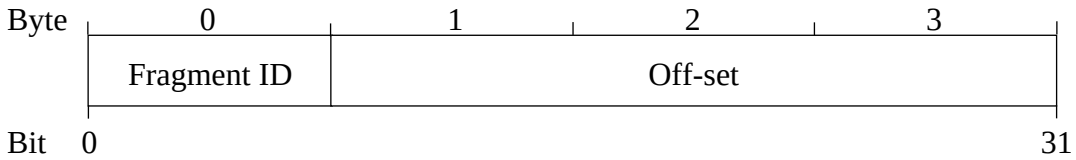
For the initial implementation there should be at least one status element, see below. Therefore this element must have a value greater than or equal to one.

5.7 Status element

This element is 32-bits wide. There must be at least one Status element. The structure and values of the element are free to be defined by the designers of the I/O modules which build the specific fragment header. There is one exception, a non-zero value of the first status element indicates that the fragment is corrupted (*e.g.* truncated, bit error etc.).

5.8 Offset element

This element is 32-bits wide and has the structure shown below.



Bytes 1 to 3 are the offset to the fragment identified by the value in byte zero and is in units of 32-bit words. Byte zero identifies the fragment found at the offset. For example: if this element is part of an event header then Fragment ID takes the value of a Sub-detector ID; If this element is part of a Sub-detector header then Fragment ID takes the value of a crate ID; If this element is part of a ROC Fragment header then Fragment ID takes the value of an I/O Module ID (*i.e.* the identifier of a ROB within the crate).

5.9 Fragment specific elements

5.9.1 Full Event Specific elements

The Full Event specific elements are defined in Table 4. Each element is a 32-bit integer. The table also presents the required order of the specific elements.

Table 4. Fragment Specific Header for the full event.

Event Header Words	Definition
Date & Time	`0xssssssss`
Run Number	`0xrrrrrrrr`
Global ID	*to be defined*
Level 1 ID	`0x00nnnnnn`
Level 1 Trigger Type	`0x000000tt`
Level 1 Trigger Info	*to be defined*
Level 2 Trigger Info	*to be defined*

- *Date & Time*: This element encodes the data and time as the number of seconds elapsed since 00.00.00 on 1st January 1997 (this defines the maximum lifetime of ATLAS as 136 years!). It provides a constantly increasing number with which one may use to time-order events.
- *Run Number*: The value of this element is unique throughout the lifetime of the experiment. Note that the precise definition of a 'run' within ATLAS is still to be defined.
- *Global ID*: The value of this element will be provided by the Event Building sub-system of DAQ/EF -1[1]. The value will be unique within a run. The format is to be defined.
- *Level 1 ID*: The event identifier generated by the Level 1 trigger system. As the latter only generates a 24-bit value, the upper byte in this element is un-used.
- *Level 1 Trigger Type*: The 8-bit word transmitted by the level 1 system. The remaining 24-bits are un-used.
- *Level 1 Trigger Info*: Trigger characteristics. Possibly multi-word data from the Level 1 trigger, containing summary information on the level 1 decision. To be defined.
- *Level 2 Trigger Info*: Trigger characteristics. Possibly multi-word data from the Level 2 trigger, containing summary information on the level 2 decision. To be defined.

5.9.2 Sub-Detector Specific Header

The fragment specific elements for a Sub-detector header are defined in Table 5. Each element is a 32-bit integer.

Table 5. Fragment Specific Header for a sub-detector.

Sub-Detector specific	Definition
Level 1 Trigger Type	`0x000000tt`

1. The value of this element must be unique for the duration of a run. If ATLAS defines a run to be 3 minutes or less then the 24-bit Level 1 ID may be used.

- *Level 1 Trigger type*: see section 5.9.1.

5.9.3 Crate Specific Header

The elements specific to a Crate are shown below. There are currently four elements defined. The Run number, Level 1 ID and Global ID are as defined in section 5.9.1. The Bunch crossing ID is as defined in section 4. Only the lower 12-bits of this element are used, the upper 20-bits are set to zero.

Table 6. Fragment Specific Header for a Crate.

Crate specific	Definition
Run number	`0xrrrrrrrr`
Bunch Crossing ID	`0x000000xx`
Level 1 ID	`0x00nnnnnn`
Global ID	*to be defined*

5.9.4 ROB Specific Header

The elements specific to a ROB fragment are shown below. There are currently four elements defined. The Level 1 ID, the Bunch crossing ID, the Level 1 Trigger Type and the Detector specific Type, see section 5.9.1, section 5.9.3 and section 4.

Table 7. Fragment Specific Header for a ROB.

Crate specific	Definition
Level 1 ID	`0x00nnnnnn`
Bunch Crossing ID	`0x000000xx`
Level 1 Trigger Type	`0x000000tt`
Detector Event type	*to be defined*

5.10 ROD Header and trailer

The initial implementation of the ROD header and trailer has been given in section 4. These elements are 32-bit integers. *e.g.* The Level 1 trigger type is an 8-bit value, therefore the remaining 24-bits are un-used.

5.11 Framing

In transmitting an event or an event fragment between elements of the DataFlow, between a ROD and a ROB or between the DataFlow and the Event Filter, the fragment must be framed by technology specific information. This framing information is not part of the event format, it is specific to the link technology used and will be removed by a receiving element. Any status information contained in the framing information will, of course, be preserved.

Current ROLs are an implementation of the S-LINK specification. This section spells out the use of S-LINK control words for framing the ROD fragment and also the detection of errors during transmission over an S-LINK ROL.

Each ROD fragment is preceded and terminated by a single S-LINK control word. These control words are referred to as the Beginning of Fragment and End of Fragment. The words are 32-bits long and take the values shown in Table 8.

Table 8. Beginning and End of fragment control words.

Crate specific	Definition
Beginning of Fragment	`0xb0f0rrrr`
End of Fragment	`0xe0f0rrrr`

The lower 16 bits of the Beginning of Fragment and End of Fragment control words are used by S-LINK to report transmission errors and are therefore subsequently reserved. All bits should be set to zero prior to transmission of the control words. Only bits [1..0] are currently used. The description and meaning of these bits is shown in Table 9.

Table 9. The meaning of the reserved bits in the S-LINK control words.

	Meaning when:	
	value is 0	**value is 1**
Bit 0	*Previous data fragment ok*	*Transmission error in previous data fragment*
Bit 1	*Control word ok*	*Transmission error in control word*

References

[1]The ATLAS DAQ and Event Filter Prototype -1 Project. http://atddoc.cern.ch/Atlas/Conferences/CHEP/ID388/ID388-1.html.

[2]Definitions, acronyms and abbreviations in ATLAS DAQ/EF prototype -1. http://atddoc.cern.ch/Atlas/Notes/046/Note046-1.html.

[3]C. Mazza et. al., Software Engineering Standards. Prentice Hall. ISBN 0-13-106568-8.