EDOC312

EUROBALL DATA ACQUISITION SYSTEM

Event-by-Event Data Output Format

Edition 2.0
May 1997

Nuclear Physics Support Group
Central Laboratory of the Research Councils, Daresbury Laboratory
&
INFN,Legnaro

Euroball Event-by-Event data output format

The **Euroball** event-by-event data tape files consist of a number of data blocks. Each data block in the file has the same fixed length. Individual events have a variable length and are packed into the data blocks with a minima amount of padding at the end of each block.

Data Block Header

The data block header identifies the type of data block (ie normal data, configuration data, comment) and also contains other information such as source of the data.

The data block header is 32 bytes in length.

data block header format

bytes 0-7	block type
bytes 8-11	sequence number
bytes 12-13	source (host)
bytes 14-15	source (id)
bytes 16-17	physical tape device (host)
bytes 18-19	physical tape device (port)
bytes 20-21	format of data
bytes 22-23	reserved
bytes 24-27	reserved
bytes 28-31	actual length of data in the block

block type

The block type identifies the format of data which follows in the block. The following block types are currently defined.

EBEVENTD	normal Euroball formatted event data
EBCONFIG	configuration data block
EBINFODA	information (comment) data block

Format of data for blocks of type EBEVENTD

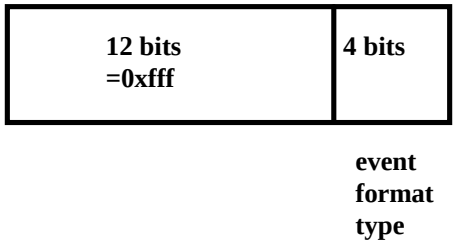
Programs designed to process Euroball event-by-event data tapes **MUST** check the block type field to ensure that the block type is **EBEVENTD.**

Each event starts with an **Event Header** which is either 4,6,8 or 10 bytes in length. All data elements within the event structure are defined in units of 16

bit words and hence the **Event Header** is aligned on a 16 bit boundary with respect to the start of the data block.
The data format is defined only in terms of 16 bit data words and thus data alignment is only on 16 bit boundaries.
Throughout this document bit 0 refers to the most significant bit of the 16 bit data words and bit 15 refers to the least significant bit of the data word.

Start Event Token

The **Start Event Token** is a single 16 bit data word which marks the start of the event and defines the format of the initial fixed part of the event.



The most significant 12 bits of the **Start Event Token** are set to 1 bits.

The **Event Format Type** is a 4 bit field which identifies the format of the event which follows. Currently only the following values are defined:

0 event number field and error pattern fields are NOT present
1 event number field is present but error pattern field is NOT present
2 event number field is NOT present but error pattern field is present
3 event number field and error pattern fields are both present

Other event format types are reserved for possible use in the future if new detector types make the current formats inefficient.

Event Length

The **Event Length** is the length (in bytes) of the whole event **INCLUDING** the **Start Event Token** and itself. Thus it is possible to step from one event to the next in the data block by adding the contents of the **Event Length** to the address of the start of the current event (the address of the current **Start Event Token**).

Event Length

Note - The **Start Event Token** most significant 12 bits are all one bits. However this is not unique since the **Event Number** following (if present) will also from time to time have the most significant 12 bits set. However in practice this is the only possible source of confusion.

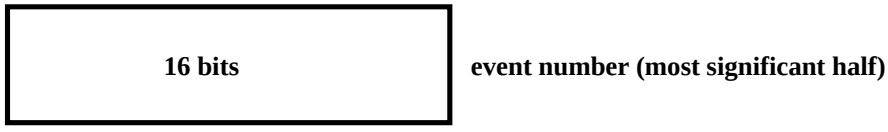
Error Pattern

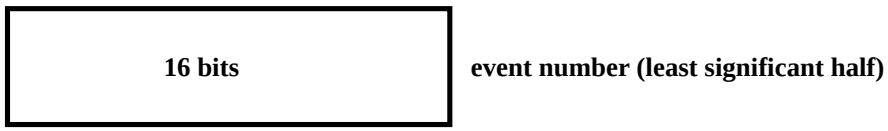
This is an optional item which is only present if the **Event Format Type** = 2 or 3.
It contains a code generated by the event processing software which indicates the status of this event. The code will be zero for an event in which no abnormal data has been detected. Non zero codes will be used to mark events which may contain abnormal information.(These error codes are to be supplied).
The default event processing option is that all potentially faulty events are suppressed and hence the **Error Pattern** item is only useful if event processing options are selected which enable the output of events which may contain faulty data.

Error Pattern

Event Number

This is an optional item which is only present if the **Event Format Type** = 1 or 3. It contains the event number as generated by the readout electronics and used by the merging software to construct the event from the data fragments received from the various data sources. This is a 32 bit number which is however stored in the event as 2 16 bit words with the most significant 16 bits stored first and the least significant 16 bits stored last.





After the **Event Header** follows a variable number of **Detector Data Items**. There will be one **Detector Data Item** for each detector in the event.

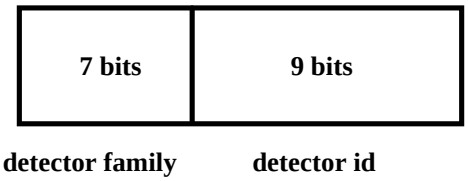
Only defined at present is the format of **Detector Data Items** for **Event Format Types** 0 to 3.

Each **Detector Data Item** starts with a **Detector Data Item Header** followed by **Detector Data** specified by the format supplied for the **Detector Family**. The **Detector Data Item** consists only of 16 bit data words and is aligned on a 16 bit boundary.

It is essential that the detector formats used to process and format the raw data are available in order to fully decode the detector data. These formats will eventually form part of the configuration data block written to each tape file.

Each **Detector Data Item** starts with a **Detector Data Item Header** which can be either 2,4,6 or 8 bytes in length. This contains a detector identification field followed optionally by length field and hit pattern field.

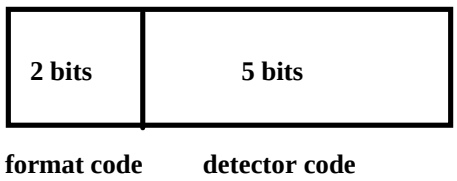
Detector Data Item Header



The **Detector Data Item Header** always starts with a 16 bit data word (**Detector Specifier**) which specifies the detector for this **Detector Data Item**. It is initially composed of two fields which specify the **Detector Family** (that is the class or type of detector) and the **Detector Id**.

Detector Family

The **Detector Family** is the most significant 7 bits of the **Detector Specifier**. The **Detector Family** is defined using the following bit field allocation.



Format code
The 2 most significant bits are a **Format Code** for the **Detector Family** and is the total number of 16 bit data words in the following part of the **Detector Data Item Header**.

The **Format Field** has the following values

= 0 **NO Hit Pattern** or **Data Fragment Length** follows; the total length of the **Detector Data Item Header** is 2 bytes.

= 1 A **Data Fragment Length** follows but there is **NO Hit Pattern**; the total length of the **Detector Data Item Header** is 4 bytes.

= 2 A **Data Fragment Length** follows and there is 1 16 bit **Hit Pattern** word; the total length of the **Detector Data Item Header** is 6 bytes.

= 3 A **Data Fragment Length** follows and there are 2 16 bit **Hit Pattern** words; the total length of the **Detector Data Item Header** is 8 bytes.

Detector Code

The next 5 bits is the **Detector Code** for this **Detector Family**. Each Detector type has a family code and the family code defines the format type for the **Detector Data Item Header**.

The following **Detector Codes** are currently defined

header	= 0
Cluster detectors	= 1
Clover detectors	= 2
Tapered detectors (Eurogam I)	= 3
Cluster detectors (full)	= 4
Ancillary (VXI)	= 5
Ancillary (fera)	= 6
Master Trigger	= 7
VRE modules (VXI ReadOut)	= 8

and the corresponding **Format Codes** are

header	= 1
Cluster detectors	= 2
Clover detectors	= 2
Tapered detectors (Eurogam I)	= 0
Cluster detectors (full)	= 3
Ancillary (VXI)	= 0
Ancillary (fera)	= 0
Master Trigger	= 0
VRE modules (VXI ReadOut)	= 0

Hence the **Detector Family** fields are

header	= 0x20
Cluster detectors	= 0x41
Clover detectors	= 0x42
Tapered detectors (Eurogam I)	= 0x03
Cluster detectors (full)	= 0x64

Ancillary (VXI)	= 0x05
Ancillary (fera)	= 0x06
Master Trigger	= 0x07
VRE modules (VXI ReadOut)	= 0x08

Future detectors will be assigned a unique **Detector Code** and one of the 4 **Format Codes**.

[**Note in explanation**: Cluster detectors will use Detector Family = 0x64 when the detector data includes individual BGO crystal information and thus requires 2 hit pattern words (full) but will use Detector Family = 0x41 when only the BGO sum channel information is required and thus only 1 hit pattern word is needed]

The **Detector Code** 31 will NOT be assigned which will ensure that the **Detector Data Item Header** cannot be confused for a **Start Event Token**.

For event decodeing purposes the **Detector Family** can be taken and a switch made to specific code for each detector family. However this will require additional code added for each new detector family. An alternative would be to use only the **Format Code** since all new detectors will use one of these.

The width of the **Detector Code** field allows for a maximum of 31 detector types.

Detector Id

The least significant 9 bits of the **Detector Specifier** is the **Detector Id** for this **Detector Data Item**. It is the index of the particular detector in its family and will be the same as the digit component of the detector name.

The width of this field allows for a maximum of 512 detectors in each family.

For example the detector Cluster8 will have Detector Family = 0x41 (Detector Code = 1 and Format Code = 2) and Detector Id = 8.

Data Fragment Length

Data Fragment Length

The **Data Fragment Length** is the length (in bytes) of this **Detector Data Item INCLUDING** the **Detector Data Item Header**. Thus it is possible to step from one data fragment (the data from one detector) to the next within the

current event by adding the contents of the **Data Fragment Length** to the address of the start of the current data fragment (the address of the current **Detector Data Item Header**). The **Data Fragment Length** is omitted if the **Format Code** = 0. In this case the length of the data fragment is determined by the format supplied for the **Detector Family** and is thus fixed for all data fragments for this **Detector Family**.

Hit Pattern

Hit Pattern

There may be 0, 1 or 2 16 bit **Hit Pattern** words. The number of **Hit Pattern** words is determined by the value of the **Format Code** and thus is determined by the **Detector Family**. Only detector families which are defined by a format which uses the subdetector feature (that is normally segmented detectors) require a **Hit Pattern**.

Tapered detectors (Eurogam I)	= 0 words
Clover detectors	= 1 word (16 bits)
Cluster detectors(full)	= 2 words (32 bits)
Cluster detectors	= 1 word (16 bits)

The bits of the **Hit Pattern** are used to indicate the presence of data from the subdetectors defined within the detector. The order of subdetector data in the **Detector Data** is the order of bits set in the **Hit Pattern** starting from the **LEAST** significant end of the **Hit Pattern**. Each subdetector generates one or more 16 bit data words as defined by the format for the subdetector.

Following the **Detector Data Item Header** are a variable number of 16 bit data words as defined by the **Data Fragment Length** (unless the **Data Fragment Length** is omitted in which case a fixed number of 16 bit data words follow). The order of data words is determined by the order in the format which defines the detector. In the case of data items for which there is a **Hit Pattern** an item in the supplied format is only present in the **Detector Data** if its corresponding bit is set in the **Hit Pattern.**

The **Detector Data Item** structure is repeated as many times as is required for the event.

As stated before the contents of the **Detector Data** is defined by the formats (NEO++ statements) used to define the detectors and subdetectors. The

examples which follow are realistic and are used to show how detector data and the detector data item hit pattern are built. However any change in the formats will result in a change to the formatted data generated.

Cluster Detector Data

The Cluster detector family (detector family 0x41) will have a detector format of the form:

```
Detector<Cluster,CLUSTER> cluster;
```

The Cluster detector will have a format of the form

```
format (Cluster)
        ClusterGe           ge[7];
end_format
```

if only data from the Ge crystals is required; and will have a format of the form

```
format (Cluster)
        ClusterGe           ge[7];
        ClusterBGOSum       bgosum[1];
end_format
```

if data from both the Ge crystals and the BGO shield sum channel is required.

The subdetector ClusterGe will have a format of the form

```
subdetector (ClusterGe)
        item_q              e20Mev;
        item_q              e4Mev;
        item                ft;
end_subdetector
```

The subdetector ClusterBGOSum will have a format of the form

```
subdetector (ClusterBGOSum)
        item                energy;
        item                timing;
        item                patternmsh;
        item                patternlsh;
end_subdetector
```

In both cases the bits of the hit pattern are allocated with the least significant bit (bit 15) being used to indicate the presence of detector data from Ge subdetector A through to bit 9 being used to indicate the presence of detector data from Ge subdetector G (that is in total 7 bits).
If the ClusterBGOSum subdetector is included in the Cluster detector format then bit 8 will be used to indicate the presence of detector data from the BGO Sum channel.

Cluster Detector Data (full)

The Cluster (full) detector family (detector family 0x64) will have a detector format of the form:

```
Detector<Cluster_full,CLUSTER_FULL> cluster_full;
```

The Cluster_full detector will have a format of the form

```
format (Cluster_full)
        ClusterGe           ge[7];
        ClusterBGOSum       bgosum[1];
        ClusterBGO          bgo[19];
end_format
```

The subdetector ClusterBGO will have a format of the form

```
subdetector (ClusterBGO)
        item                energy;
        item                timing;
end_subdetector
```

This detector family requires 2 hit pattern words.
The hit pattern bits are allocated with the least significant bit (bit 15) of the first hit pattern word being used to indicate the presence of detector data from Ge subdetector A through to bit 9 being used to indicate the presence of detector data from Ge subdetector G (that is in total 7 bits).
Bit 8 of the first hit pattern word will be used to indicate the presence of detector data from the BGO Sum channel.
Bit 7 of the first hit pattern word will be used to indicate the presence of detector data from BGO subdetector A through to bit 0 (the most significant bit) of the first hit pattern word being used to indicate the presence of detector data from BGO subdetector H. The least significant bit (bit 15) of the second hit pattern word will be used to indicate the presence of detector data from BGO subdetector I through to bit 6 being used to indicate the presence of detector data from BGO subdetector R.

Clover Detector Data

The Clover detector family (detector family 0x42) will have a detector format of the form:

```
Detector<Clover,CLOVER> clover;
```

The Clover detector will have a format of the form

```
format (Clover)
        dummy_item      unused[4];
        dummy_item      unused[3];
        dummy_item      unused;
        CloverGe        ge[4];
end_format
```

if only data from the Ge crystals is required; and will have a format of the form

```
format (Clover)
        dummy_item      unused[4];
        CloverBGO       bgo[1];
        dummy_item      unused;
        CloverGe        ge[4];
end_format
```

if data from both the Ge crystals and the BGO shield sum channel is required.

The subdetector CloverGe will have a format of the form

```
subdetector (CloverGe)
        item_q          e20Mev;
        item_q          e4Mev;
        item            ft;
        item            co;
end_subdetector
```

The subdetector CloverBGO will have a format of the form

```
subdetector (CloverBGO)
        item            energy;
        item            timing;
        item            pattern;
end_subdetector
```

If there is **NO** BGO data then the bits of the hit pattern are allocated with the least significant bit (bit 15) being used to indicate the presence of detector data from Ge subdetector A through to bit 12 being used to indicate the presence of detector data from Ge subdetector D (that is in total 4 bits).
If the CloverBGO subdetector is included in the Clover detector format then the least significant bit (bit 15) will be used to indicate the presence of detector data from the BGO channel and bit 1 will be used to indicate the presence of detector data from Ge subdetector A through to bit 11 being used to indicate the presence of detector data from Ge subdetector D.

Tapered Detector Data

The Tapered detector family (detector family 0x03) will have a detector format of the form:

```
Detector<Tapered,TAPERED> tapered;
```

The Tapered detector will have a format of the form

```
format (Tapered)
        item_q          e20Mev;
        item_q          e4Mev;
        item            ft;
        item            co;
        item            energy;
        item            timing;
        item            pattern;
end_format
```

This format does not use subdetectors and the **Detector Data Item** does not contain a **Hit Pattern**. All items defined by the detector format (7 in the above example) appear in the detector data.

Note: VXI data readout errors generate raw data that unless suppressed will produce a formatted event 0xfff0 0x0008 0x0a5c 0xffff. Thus it appears to be ancillary VXI detector channel 92. This data is defined by the detector format

```
Detector<Dead,ANCILLARY_VXI> dead;
```

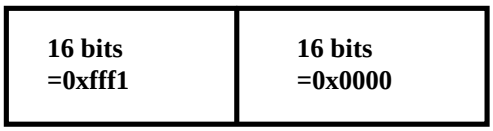
(This may be changed in the future).

There is no **End Event Token** since the total length of the event is available in the **Start Event Token**.

Events are packed into the data block until there is insufficient room for the next event to fit totally into the current data block. Events are complete within a data block and will not span data blocks.

The end of events in a data block is marked by a special **Event Header** having the format

End Data Block Token



This is equivalent to a **Event Header** for an event of length = 0 and no **Event Number.**