Eurogam Event Builder Software

John Cresswell

September 1990

1 Introduction

The Eurogam PSC has defined a list of general actions for the Event Builder to process, together with suggestions of items it should not. It has been stated that it will be necessary to provide support for Compton suppression, New word formation, Event filtering and Simple arithmetic. It has also been stated that the Event Builder will not support operations that are not 100% reliable, such as Gain matching and Ballistic deficit correction.

The prime function of the Event Builder must be to build events containing the defined set of parameters in the defined format. There will need to be some consistency checking on the quality of events processed. For example notification of parameter words that are not expected. In this case the user has wrongly identified the set of parameters to be collected, or the hardware has generated corrupt information.

A second criterion for event builder processing would be to reduce the data volume for onward shipment. Since the event builder has to process 100% of a high data volume (by definition) then any processing has to be simple and effective.

A third criterion might be to prevent confusion for the physicist as to what processing should be done in the Event Builder or in the Sorter. This should be achievable by limiting the functionality of the Event Builder and making the user interface different.

2 Discussion

The list of operations asked for by the PSC above would probably be difficult to code without resorting to a compiled language as used in the sorter. The sort language itself is not really appropriate for describing these operations unless they were built in as extra functionality which would make the sort language bulky and Eurogam specific (for shared suppression).

The efficiency of computation required to cope with the level of processing necessary suggests that code will have to be written specifically for the Eurogam array. The known structure of the array and the input format of the data will be used to maximise the efficiency of the software.

In particular, the set of Ge and associated BGO detectors can be considered as a group of parameters. For a single Ge detector this group will have a maximum of 8 parameters. Each event will consist of several such groups preceeded by a trigger or system group. This ordering of parameters into associated groups should make the processing easier to accomplish. Several of the functions to be applied to each event will be group oriented. For example, the sum energy calculation will include the same words from each of the Ge groups. Also, a typical operation may be to filter on all Ge TDC words ... again the same operation performed on the same word from each group.

3 Solution

The program will require the definition of three sections ...

- Input parameter list
- Actions to be performed
- Output parameter list and format

The input parameter list is just the set of hardware devices that the user has defined. This information will be available elsewhere in the system. It should be available in a format suitable for entry into the Event Builder setup facility.

The output parameter list will be affected by the set of actions performed on each event. For example it could be assumed that the set of parameters output will be those in the input list together with any new words created and excluding any words specifically rejected by the actions section.

The output section will also have to define the format of the parameters to be output. The event format specification allows for a group format of associated parameters. This will save space depending on the group size.

The action section will typically consist of a list of functions to be applied to the event together with a resultant operation. Resultant operations will include the following ...

- Reject parameter(s)
- Reject group
- Reject event
- Increment variable (for later testing)

4 Input data format specification

This will require liaison with the people involved in designing the code to setup the acquisition hardware. It will consist of a list of names and associated addresses in a simple text format.

5 Output data format specification

This will essentially be the same as the input data format specification of the Sorter. At the moment it is expected to use the same data file section as entry for setting up both the Event Builder and the Sorter. See the *FORMATS section of the Sort Language manual (EDOC033).

6 Command specification

It is proposed to divide the processing into those functions that can be applied to an individual word, to an individual group and to those that can only be applied after the whole event has been read in.

Some functions will require information to be extracted during processing of the event, but resulting in a decision only when all the event has been read in.

The shared suppression algorithm will process hit pattern words as they are encountered during input of the event, but will not be able to take action immediately as the particular Ge word to be suppressed may occur later in the event. Similarly, a filter on the Ge TDC words could result in a count incremented on passing each filter test, but a filter on the count itself could only be applied at the end of event processing.

7 Commands

reject<device-name>

reject <group-name>

reject<event>

The commands above would only be specified after any functions using the parameter involved. They have the effect of omitting the defined word or words from the output event.

filter a,b <parameter-name> <result>

The filter command is similar to the filter command of the Sort Language. It passes or fails dependent on whether the value of the parameter lies within or outside the gate-pair a,b. The result field describes the action to be taken on pass or fail.

<simple-expression>

e.g. SUMENERGY = SUMENERGY + EBG0/2 + E2

The expression allows simple arithmetic to be applied to the defined parameter.

if <condition> { ... }

```
where <condition> is one of <device-name>
                            <group-name>
                            <device-name> = <value>
                            group = <value>

       <result> is specified as (success-result;fail-result)
                and each field may take one of the following values

                            keep-word    (default)
                            reject-word
                            reject-group
                            reject-event
                            increment <variable>

        e.g.   IF  GE_GROUP
                {
                SUMENER + EBGO/2 + E2
                FILTER 100,250  T1  (keep-word;reject-group)
                }
```

shared-suppression <data-file>

where <data-file> contains suppression information. The document entitled Eurogam Suppression provides an example which could provide the basis of the necessary data representation.

output { ... }

This command allows the execution of any commands enclosed after all the event has been read and before the event has been written to the output buffer.

We may need one or two more commands, but not many ...

8 Command section example

This is taken from the document of Beausang and Fallon entitled "A Mind Experiment with Eurogam". It is not intended to be a complete example, but to give a flavour.

```
if (group=255)
   {
   if (Mraw < 8) (;reject-event)
   }

if (group=GE)
   {
   SUMENER=SUMENER+(E2+EBGO)
```

```
   reject E2,EBGO
   shared-suppression <filename>
   reject HIT
   filter  E1  a,b    (;reject-event)
   filter  T1  c,d    (;increment FT1)
   reject T1
   }

output
   {
   filter SUMENER  e,f    (;reject-event)
   if (FT1 < 1)           (;reject-event)
   }
```

Named parameters (Mraw,GE,E2,EBGO,HIT,E1,T1) are defined in the input section. Note that "GE" is any Germanium group of parameters. This will be defined in the input section as any group in the range [1..70].

```
e.g.

name=GE  group=[1..70]  items=(E1,E2,T1,.....,HIT)
```

Other named parameters (SUMENER,FT1) are assumed by usage to be zeroed at start of each event.

9 Programming technique

We propose to use the same compiler technique as for the sorter. We have the source of BISON (GNU version of YACC) and FLEX (GNU version of LEX), and will enhance the sort language using these tools. By comparison with the Sort Language, these proposals will be relatively easy to implement. Some further detail may be found in the Event Builder and Sorter Controls document.

The command description above is not meant to specify the language exactly, but to give a general idea.

10 Diagnostics

Crate-wide hardware diagnostics will be incorporated to detect faulty components wherever possible. The main testing will occur during power-up initialisation. This will include memory pattern writing, register writing and reading, responses from all cpus in the crate, and so on. Hardware fault detection during normal operation has to be a bit more limited, but interrupt redirection can be used to spot faulty addressing at least. The full range of possible checking will require some coordination of ideas from all developers involved.

Software diagnostics will include some automatic statistics gathering during Event building. This will include the number of events input and output, the number of bad parameters, the number of data blocks output etc...

In the case of errors found in the input data stream it would not be sensible to send a message to the user every time. Several errors per event and many thousands of events per second would overwhelm the user. It would be possible to keep counts of some types of errors. It would be possible to keep a local record of the previous 100 errors, for example, which would be overwritten cyclically. For the present, I will leave this subject for comment.