
Subject: Some UCESB zero-suppressed multi hit documentation

Posted by [Ralf Plag](#) on Thu, 28 Jan 2016 13:05:12 GMT

[View Forum Message](#) <> [Reply to Message](#)

For electronics that produces multi hit data, here is how to handle it through ucesb:

In the ucesb .spec file:

Declaration (here for 16 energy channels):

```
MEMBER(DATA32 E[16] ZERO_SUPPRESS_MULTI(10));
```

Filling (as usual):

```
ENCODE(E[channel_id],(value=data));
```

Mapping:

It is necessary to declare the output variable as multi hit type as well:

```
SIGNAL(ZERO_SUPPRESS_MULTI(10): raw_level_output_variable);
```

The actual mapping then goes as usual.

Example: Mapping febex data to the PSPX01:

```
// Declaration of output variable PSPX01:
```

```
SIGNAL(ZERO_SUPPRESS_MULTI(10): PSPX01_01);
```

```
// Mapping of channels
```

```
SIGNAL(PSPX01_01, psp_febex.febex2.E[14], DATA32);
```

```
SIGNAL(PSPX01_02, psp_febex.febex2.E[15], DATA32);
```

```
SIGNAL(PSPX01_03, psp_febex.febex2.E[12], DATA32);
```

```
[...]
```

Reading the data

In the ucesb output, either via root TTree or struct_writer you'll find the following variables (sticking to the PSPX01 example):

PSPX1: size of array PSPX1v

PSPX1v: array of size PSPX1 holding the actual data (here energy values)

The order in the v-array is: all hits of first channel with data, all hits of second channel with data, ...

PSPX1M: number of channels that have data

PSPX1Ml: array of size PSPX1M holding the channel numbers of each channel with data

PSPX1ME: array of size PSPX1M providing the index in array PSPX1v of the first element of the next channel (or more generally: last element of current channel + 1)

--

Example data:

Channel 4 has two hits: E=430 and E=485

Channel 9 has three hits: E=938, E=923, E=967

PSPX1 = 5

PSPX1v = 430, 485, 938, 923, 967

PSPX1M = 2

PSPX1MI= 4, 9

PSPX1ME= 2, 5

--

When reading the channels, the following pseudo code should work:

```
numChannels = PSPX1.M;
```

```
curChannelStart=0; // index in v for first item of current channel
```

```
for (i=0; i<numChannels; i++)
```

```
{  
    channel=PSPX1.MI[i]; // channel number: 1..max_channel
```

```
    // get index in v for first item of next channel:
```

```
    nextChannelStart=PSPX1.ME[i];
```

```
    // read all hits of channel 'channel'
```

```
    for (j=curChannelStart; j < nextChannelStart; j++)
```

```
    {  
        energy=PSPX1.v[j];  
        // do something with energy  
    }
```

```
    curChannelStart=nextChannelStart;
```

```
}
```