JIVE Uniboard Correlator Memo 10: JUC configuration output for j2ms2

Des Small

October 26, 2012

1 Introduction

The current proposal for the Jive Uniboard Correlator (JUC) is that it will correlate datastreams which correspond to integer station identifiers from 0 to 31 inclusive and subbands (actually just input buffers) from 0 to 3 inclusive, where each subband can be specified to contain one or two polarizations.

The JUC will not itself have any knowledge of how these "stations" and "subbands" map to physical stations or frequencies – it is quite possible that we will actually correlate sixteen physical stations and eight subbands, with each physical station's subbands being divided over two virtual stations on the JUC.

To be able to post-process the data into measurement sets the program `j2ms2` will need to be supplied with information about the configuration, including the map between real and virtual stations. It is intended that this data will be recorded in json, in the format specified in this document.

2 File format

2.1 Simple attributes

The top level of the json file is an object. The `"setup"` key maps to a sub-object specifying

- the integration time (float) in seconds;
- the number (integer) of spectral points;
- the complete set of polarization pairs whose products should be included, encoded as a list of two letter strings (the order of this list will define the order in the measurement set);
- the correlator configuration as a decimal integer (json does not support hex literals);
- the start time of the correlation; and
- the version number of the firmware for that correlator configuration (integer).

```
    "setup" : {
        "integrationtime" : 0.25,
        "spectralpoints" : 1024,
        "polarizations" : ["ll", "rr"],
        "correlatorconfig" : 8234234,
        "correlatorversion" : 0,
        "starttime" : "1999-12-16T00:30"
    }
```

Listing 1: The setup section

See Listing 2.1 for an example.

The `"datafiles"` key simply specifies the datafiles that contain the data; entries beginning with a leading dot, as shown in Listing 6, refer to data files relative to the location of the json file itself – in other words, they assume the configuration file has been written to a directory where the data also is.

2.2 Compound attributes

2.2.1 A grammar

A Backus-Naur grammar of the compound attributes is shown in Listing 2.2.1. Since javascript lacks a tuple type distinct from lists we use square brackets to designate fixed-length lists intended as tuples, and use `list(…)` for homogenous lists of arbitrary length. Another, related, quirk of Javascript is that only strings can serve as the key of an object (the language's dictionary or mapping type). As a result we often find ourself needing to compose a compound key as a comma-separated string.

⟨*ce_sb*⟩ ::= `dict(`⟨*ce_sb_key*⟩ `:` ⟨*virtual_subband*⟩`)`

⟨*ce_sb_key*⟩ ::= `"`⟨*uniboard*⟩`,`⟨*node*⟩`,`⟨*correlation_engine*⟩`"`

⟨*input_mapping*⟩ ::= `dict(`⟨*input_mapping_key*⟩`:[`⟨*virtual_uniboard*⟩`,` ⟨*virtual_station*⟩`,` ⟨*virtual_subband*⟩`])`

⟨*input_mapping_key*⟩ ::= `"`⟨*station*⟩`,`⟨*input_subband*⟩`"`

⟨*enabled_products*⟩ ::= `list(`⟨*product*⟩`,` ⟨*station0*⟩`,` ⟨*pol0*⟩`,` ⟨*station1*⟩`,` ⟨*pol1*⟩`,` ⟨*enabled_product_engines*⟩`)`

⟨*enabled_product_engines*⟩ ::= `list( "`⟨*uniboard*⟩`,`⟨*node*⟩`,`⟨*correlation_engine*⟩`")`

Listing 2: Grammar for compound attributes

2.2.2 Enabled products

The `"enabled_products"` key contains a list of

```
"enabled_products" :  [
    [0, 0, "r", 1, "l", ["0,4,0", "0,5,0"]],
]
```

Listing 3: Example of the enabled products attribute

- product number (integer) from routing table;
- JUC virtual station number (integer < 32) for first station in correlation
- polarization (`"l"` or `"r"`) for first station
- JUC virtual station number for second station
- polarization for second station
- a list of strings encoding triples of uniboard number, backnode number (an integer from 4 to 7 inclusive) and correlation engine (integer).

The last of these requires several remarks.

- In principle, each correlation engine has a separate list of products to output, but we invert the list and make product numbers the key for brevity;
- the backnodes are numbered from 4 to 7 (for the current firmware) since numbers are global to each Uniboard and the front nodes occupy 0 to 3 (again, in the current firmware);
- the triples are encoded as strings as usual for Javascript even though they are not used as keys here (the strings are however precisely the keys to the `ce_sb` dictionary).
- the routing table is a property of the JUC firmware; products calculated by back nodes are identified in the correlator output only by product number, so to reconstruct even the virtual station numbers used by the front nodes we need this table;
- there may be more than one correlation engine on a given back node in the future

2.2.3 Input mapping

```
"input_mapping" : {
    "Jb,0" : [0,0,0],
    "Jb,1" : [0,0,1],
    "Ef,0" : [0,7,0]
}
```

Listing 4: Example of the input mapping attribute

The `"input_mapping"` key maps to an object mapping pairs of two-letter station names and physical subband numbers to pairs encoded as lists of virtual station number (0 to 32) and subband (0 to 4). See Listing 2.2.3 for an example. It is important to note that the virtual subband corresponds to three less than the backnode number: this is the only way that virtual subband is encoded in the output, and each backnode has an identical routing table.

2.2.4 Correlation engine to subband map

```
    "ce_sb" : {
        "0,4,0" : 0,
        "0,5,0" : 1,
        "0,6,0" : 2,
        "0,7,0" : 3
    }
```

Listing 5: An example of the correlation engine to subband map attribute

The `"ce_sb"` key describes the map from backnode and correlator engine to virtual subband as shown in Listing 2.2.1. It maps a compound key composed of uniboard number (the backplane number 0..31 – the highest five bits of the 8-bit hardware node number), node and correlation engine number to the virtual subband processed.

3 A complete example

A complete example configuration file is given in Listing 6.

```
{
     "setup" : {
         "integrationtime" : 0.25,
         "spectralpoints" : 1024,
         "polarizations" : ["ll", "rr"],
         "correlatorconfig" : 8234234,
         "correlatorversion" : 0,
         "starttime" : "1999y334d12h14m00s"
     },
     "DATAFILES" : ["./file1.unb", "./file2.unb"],
     "enabled_products" :  [
         [0, 0, "r", 1, "l", ["0,4,0", "0,5,0", "0,7,0"]],
     ],
     "input_mapping" : {
         "Jb,0" : [0,0],
         "Jb,1" : [0,1],
         "Ef,0" : [7,0]
     }
    "ce_sb" : {
         "0,4,0" : 0,
         "0,5,0" : 1,
         "0,6,0" : 2,
         "0,7,0" : 3
    }
}
```

Listing 6: A complete example j2ms2 configuration for the JUC