Purpose

This document defines how to describe the configuration and hookup of the PSA128 array to the PSA256 correlator.

Reason

The PSA128 array and PSA256 correlator have many cables interconnecting components to make the overall signal path. Being able to represent these interconnections is important to ensure that the data are written properly.

Due to the large number of cables and interconnections, the chance for errors is significant. Having a clear understanding of the signal path helps when tracking down such errors.

Due to the large volume of data generated by the PSA256 correlator, applying corrections after the fact is prohibitive. It is far better to write the data correctly in the first place. Having a clear understanding of the signal path helps to ensure that the data writing software can properly conjugate the data and label it correctly.

Goals

The implementation of the hookup description should meet these goals:

1) Easy for humans to read and write
2) Easy for computers to parse and generate
3) Easy for version control systems (e.g. git) to track changes

Terminology

The following terms are used to describe the array configuration, including hookup information. The terms are defined specifically in the context of describing the array configuration.

Antenna

This is the antenna number. The PAPER convention is to number antennas from 0, whereas MIRIAD numbers antennas from 1. For the purposes of describing the configuration of the PAPER array, the PAPER convention of numbering from 0 is used.

Station

This is the name of an antenna location. Station names follow a convention started with the maximum redundancy configuration which had the antennas arranged in a grid aligned to the four compass points. The east-west rows of the grid are designated with a letter starting with A on the north. The north-south columns of the grid are designated with a number starting with 0 on the west. The current configuration includes outlier antennas that are not on the grid (though some are interspersed within the grid).

Receiver

This is the name of the dual-pol receiver channel. The 128 dual polarization receivers for the PSA128 array are located in 8 refrigerators lined up west to east just south of the PAPER hut. Each refrigerator contains 16 receivers on two shelves (8 receivers per shelf). The refrigerators are numbered 1 (westernmost) to 8 (easternmost). On the left and right side are 32 input and 32 outputs connectors (one side is input, the other output). Each pair of connectors is labeled A1 through A8 and B1 through B8.

Plate Feed-Through Connectors

The output of the receivers are connected to the feed-through connectors mounted on 6 plates attached to the bulkhead wall of the south side of the PAPER hut (i.e. container). The plates are arranged in three rows, two plates per row. They are numbered 1 to 6, from west to east, top to bottom. Plate 1 is the westernmost, uppermost; Plate 6 is the easternmost, lowest. Each plate has 48 feed-through connectors arranged in 6 rows of 8.

F Engine Input

Inside the paper container, the signals are fed to the inputs of the correlator's eight F engines. The F engines are numbered from 1 to 8. Each F engine has 32 inputs labeled A1 through H4.

Nomenclature

This section describes the naming convention used to refer to specific instances of the components described in the previous section. Names consist of letters and/or numbers that follow a pattern defined for the type of component being named. Names are case sensitive. The names described here are canonical for use in the context of array configuration, but they are not intended to be globally canonical. Included in th descriptions below are POSIX regular expressions that can be used to match names.

Prefix Letters

To minimize verbosity while at the same time maximizing readability and comprehensibility, each type of component has an associated prefix letter. Prefix letters are lowercase. Prefix letters are unique across all currently defined types, but future types may re-use an existing prefix letter. Any future reuse of an existing prefix letter must ensure that there is no ambiguity with the existing names.

Using a single letter as the prefix is the balance between terseness and disambiguation. For example, antenna 1 is named `a1`, which could be confused with station A1 even though station A1 is named `sA1`. To minimize the potential for confusion when reading a name out loud or to yourself, the prefix letter should be expanded to its component type or omitted altogether if the context makes it clear.

- `a` for Antenna
- `s` for Station
- `r` for Receiver
- `p` for Plate Feed-Through Connector
- `f` for F Engine Input

Antenna [`a`]

For the PSA128 array, the antennas are numbered from 0 to 127, inclusive, so the valid antenna names are `a0` through `a127`. Additionally, negative antenna numbers are allowed. Antenna number -1 is assumed to be at the reference location. No other negative antenna numbers are pre-defined. Note that leading zeros *must not* be used in the numeric part (even though the regular expression below will match leading zeros).

Antenna regular expression: `/^a-?\d+$/`

Station [`s`]

The PSA128 array has a main grid consisting of rows A (northernmost) to G (southernmost) and columns 0 (westernmost) to 15 (easternmost). The 16 outlier stations are labeled as if they were in row X and the array reference position (the south-east corner of the hut) is labeled as if it were in row Z column 0. The 129 station names for the PSA128 array are `sA0` to `sG15`, `sX0` to `sX15`, and `sZ0`. Note that leading zeros *must not* be used in the numeric part (even though the regular expression below will match leading zeros).

Station regular expression: `/^s[A-Z]\d+$/`

Receiver [`r`]

The dual-pol receiver channels for the PSA128 array are named:

```
r{rx_number}{shelf_name}{shelf_unit}
```

where rx_number ranges from 1 to 8, shelf_name is 'A' or 'B', and shelf_unit ranges from 1 to 8. The 128 dual-pol receiver channel names are `r1A1` to `r8B8`.

Receiver regular expression: `/^r[1-8][AB][1-8]$/`

Plate Feed-Through Connector [`p`]

The plate feed-through connectors for the PSA128 array are named:

```
p{plate_number}r{row_number}c{column_number}
```

where plate_number ranges from 1 to 6, row_number ranged from 1 to 6, and column_number ranges from 1 to 8. The 256 plate feed-through names are `p1r1c1` to `p6r2c8`. Feed-throughs `p6r3c1` to `p6r6c8` are for temperature monitoring signals and spares.

Plate feed-through regular expression: `/^p[1-6]r[1-6]c[1-8]$/`

F Engine Input [`f`]

The PAS256 correlator F engine inputs are name named:

```
f{f_engine}{adc_chip}{adc_channel}
```

where f_engine ranges from 1 to 8, adc_chip ranges from 'A' to 'H', and adc_channel ranges from 1 to 4. The 256 input names are `f1A1` to `f8H4`.

F engine input regular expression: `/^f[1-8][A-H][1-4]$/`

Polarizing Names

The antennas and receivers are dual polarization (X and Y). To refer to a specific polarization, append an 'X' or 'Y' to the antenna or receiver channel name. For example, the X polarization of `a1` (i.e. antenna 1) is named `a1X`.

Although stations represent positions rather than actual signals, station names can also be polarized using this technique. A polarized station name actually refers to the specified polarization of the antenna that is on the specified station. For example, if `a1` (antenna 1) is on `sA2` (station A2) then `sA2Y` is another name for `a1Y`.

When reading a polarized name, the type of the component is augmented with "polarization" or, in shortened form, "pol" (pronounced "pole"). For example, `a12X` would be read as "antenna polarization 12X" or "antpol 12X", `sG4X` would be read as "stationpol G4X", and `r2B2Y` would be read as "receiver polarization 2B2Y" or "rxpol 2B2Y".

Plate feed-through connectors and F engine inputs cannot be polarized in this manner.

Antenna polarization regular expression:	`/^a-?\d+[XY]$/`
Station polarization regular expression:	`/^s[A-Z]\d+[XY]$/`
Receiver polarization regular expression:	`/^r[1-8][AB][1-8][XY]$/`

Representation

The array configuration and hookup information can be described by a collection of mappings that relate instances of one component type to instances of another component type.

Signal Path

The signal path (aka hookup information) can be described by mapping antpols to rxpols, rxpols to plate feed-throughs, and plate feed-throughs to F engine inputs. For valid configurations, each of these mappings is invertible (i.e. the antpols-to-rxpols mapping can be inverted to create an rxpols-to-antpols mapping). Each mapping can be thought of as a collection of cables. In fact, a collection of cables is exactly what these mappings represent. Each member of a mapping is a cable (or part of a multi-conductor cable).

Array Configuration

The array configuration can be described by mapping antennas to stations and mapping stations to locations.

Implementation

Data Format

The mappings representing array configuration and hookup information can be expressed in a wide variety of formats (e.g. text, spreadsheet, image) and media (e.g. computer files, posters, back of a napkin). Text based computer files are arguably the best choice for meeting all of the goals given above. They perhaps fall short on providing humans with an overall understanding of the array configuration and hookup, but any format/media that can concisely convey that amount of complexity to humans will fall short on the other two goals. It is easier and less error prone for computers to transform text files into human-friendly form than the other way around.

Many existing text file formats can express mappings. Most, if not all, of these formats have readily available parsers. The format chosen for the PSA256 correlator back end software is based on YAML (http://yaml.org/). YAML meets all of the goals specified above:

1) YAML is easy for humans to read and write
2) YAML is easy for computers to parse and generate
3) YAML is easy for version control systems (e.g. git) to track changes

Many other text file formats would have worked, but YAML seemed like the simplest to implement and there was no time for a selection committee.

Mappings in YAML

Mappings in YAML are very straightforward. They are reminiscent of Python dictionaries and Ruby hashes. Basically, a mapping is a collection of (key, value) pairs. YAML has several ways of expressing a mapping. Perhaps the easiest way is one key/value pair per line. The format is simply the key followed by a colon and a space followed by the value. For example, mapping key `foo` to value `bar` would look like this:

```
foo: bar
```

There must be at least one space character after the colon. To make things more readable, extra space characters can be added before and/or after the colon. Do not use tab characters.

YAML also supports comments, which are introduced by an octothorpe and continue until the end of the line. These comments are not part of the mapping itself, but provide out-of-band info for humans.

Mappings can be inverted in software to create a new mapping that uses the keys and values of the original mapping as values and keys, respectively, of the new mapping. If multiple keys map to the same value in the original mapping, the behavior in the inverted mapping is unspecified. For the purpose of array configuration, valid mapping do not contain duplicate values.

Mappings are allowed to have an empty value, but not an empty key. When inverting a mapping, keys that have empty values will be dropped.

Mapping Conventions

For array configuration and hookup information, mappings simply map names of one type to names of another type. Sometimes some extra metadata is needed or useful for describing the mapping. For example, in the station to position mapping it is useful to know the units, origin, and frame of the positions. Metadata can be added to any mapping by adding a `metadata` key that maps to a mapping of metadata key-value pairs. This can be done with an indented mapping after the `metadata` key:

```
# Example showing metadata.
#
# This is the metadata key that maps to a three element mapping containing keys
# "units", "origin", and "frame".  Notice that the three element mapping is
# indented to make it the value of the "metadata" key instead of just more "top
# level" mappings.
metadata:
  units  : m
  origin : # TBD
  frame  : ENU

# Start mapping
sA0  : [540846.114, 6600986.731, 1048.200]
# etc...
```

Signal Path

Three mappings define the PAPER signal path:

- Antpol to Rxpol – This is set of a dual-conductor cables with F connectors on both ends.
- Rxpol to Plate Feed-Through Connector – This is a set of single conductor cables with SMA connectors on both ends.
- Plate Feed-Through Connector to F Engine Input – This is a set of single conductor cables soldered to the SMA bulkhead connectors at one end and SMB connectors on the other.

These are the "forward flow" mappings, but since each mapping is invertible it is possible to derive the "backward flow" mappings relatively easily.

The mappings are defined below in YAML using the names described in the Nomenclature section as both keys and values.

Antpol-to-Rxpol Mapping

```
# This is only an example!
# It is not intended to reflect actual cabling!
a0X   : r1A1X
a0Y   : r1A1Y
# ...
a127X : r8B8X
a127Y : r8B8Y
```

Rxpol-to-Plate-Feed-Through-Connector Mapping

```
# This is only an example!
# It is not intended to reflect actual cabling!
r1A1X : p1r1c8
r1A1Y : p1r1c7
# ...
r8B8X : p6r2c2
r8B8Y : p6r2c1
```

Plate Feed-Through-Connector-to F-Engine-Input Mapping

```
# This is only an example!
# It is not intended to reflect actual cabling!
p1r1c8 : f1a1
p1r1c7 : f1a2
# ...
p6r2c2 : f8h3
p6r2c1 : f8h4
```

Array Configuration

Antenna to Station

```
# This is only an example!
# It is not intended to reflect actual configuration!
a0   : sB3
a1   : sG12
# ...
a127 : sX15
a-1  : sZ0
```

Station to Antenna

This is just the inverse of the antenna-to-station mapping. As of this writing all 128 stations of the PSA128 array have been defined, but not all antenna numbers assigned, so this direction may be easier to start with. Entry `sB0` in this example has no value (just a TBD comment).

```
# This is only an example!
# It is not intended to reflect actual configuration!
sA0  : a127
sB0  : # TBD
sB8  : a1
sD12 : a2
sE9  : a0
sG15 : a3
```

Station to Position

Positions can be given in various units with various origins oriented to different reference frames. To describe the units, origin, and frame used in a given mapping, the keys `units`, `origin`, and `frame` can be given as keys of a `metadata` mapping. The value for units should be self apparent. Two common values are `m` for meters and `ns` for nanoseconds, but more esoteric units are possible (e.g. `1/150 MHz`). The value for `origin` should be any suitably descriptive string. The value for `frame` should be any suitably descriptive string, e.g. `geocentric`, `topocentric`, `ENU`, `ENZ`, etc. These extra keys are optional. If omitted they must be somehow communicated or inferred.

The values for position are given as a three (typically) element comma delimited sequence of floating point values surrounded by square brackets (which is known as a "sequence" in YAML).

```
# This is only an example!
# It is not intended to reflect actual configuration!
metadata :
  units  : m
  origin : # TBD
  frame  : ENU
# Station name to [east, north, up] mapping
# sA0-sG15 are main grid stations
# sXn are outlier stations
# sZ0 is reference "station" (SE corner of hut)
sA0  : [540846.114, 6600986.731, 1048.200]
sB0  : [540846.091, 6600982.773, 1048.256]
sC0  : [540846.110, 6600978.796, 1048.290]
```

```
# ...
sF15 : [541070.969, 6600966.853, 1048.509]
sG15 : [541070.955, 6600962.844, 1048.512]
sX0  : [     0.000,       0.000,    0.000] # TBD
# ...
sX15 : [     0.000,       0.000,    0.000] # TBD
sZ0  : [540976.442, 6601013.289, 1053.131]
```

Files

Hookup and configuration files for PAPER are maintained in a Git repository [maybe on GitHub if not too sensitive]. For any given array, the hookup and config files are stored in a subdirectory named after that array (in lowercase). For example, files for the PSA128 array and PSA256 correlator are stored in a `psa128` subdirectory.

File Naming

Each type of component has a short mnemonic that is used in file names. Mnemonics for polarized names simply add `pol` to the end of the non-polarized mnemonic. The mnemonics for the various component types are given here:

Component Type	Mnemonic
Antenna	`ant`
Antenna Polarization	`antpol`
Station	`sta`
Station Polarization	`stapol`
Receiver	`rx`
Receiver polarization	`rxpol`
Plate Feed-Through Connector	`plate`
F Engine Input	`fxin`
Position	`pos`

In general, a file describing a mapping from mnemonic `abc` to mnemonic `xyz` is named `abc_to_xyz.yml`. The `.yml` extension is typical for YAML files. For example, a file containing antenna-to-station mappings would be named `ant_to_sta.yml` whereas one containing the inverse mapping of station-to-antenna would be called `sta_to_ant.yml`.

Of course, files can be named anything the user wants, but array and correlator operation and observation software may expect or even require the use of the canonical names described above.

Validation

Software (or diligent humans) can validate the consistency of a group of files containing interrelated mappings. For example, verify whether `ant_to_sta.yml` is indeed the inverse of `sta_to_ant.yml`. Missing values may result in an outcome of "not inconsistent" rather than an outcome of "fully consistent".