The ST TELL1 Emulator

This page contains information about the ST TELL1 Emulator and how to use it within the Vetra environment (v7r0 and up). The current version of the emulator used is **ST/STTELL1Algorithms v3r4**.

The packages used by the Emulator are:

- **ST/STTELL1Algorithms**
- **Tell1/TELL1Engine**
- **DAQ/Tell1Kernel**

The Emulator is included in the Vetra release, thus there is no need to install these packages locally. We can run the Emulator within the Vetra environment.

STTELL1Algorithms contains the following 4 algorithms:

- Pedestal Subtraction (Including Pedestal Update and Header Correction)
- Common Mode Suppression
- Zero Suppression
- Monitoring

The first three algorithms function as a *wrapping* around the C-code written processes (supplied by Guido), which can be found in the TELL1Engine package. STTELL1Algorithms is the only package which is specific to the ST. Both TELL1Engine and Tell1Kernel are also used by the Velo.

Before running the Emulator the raw data must be decoded. This is done using the STFullDecoding algorithm (ST/STDAQ). To run the Emulator add the following lines to the options file:

```
ApplicationMgr.TopAlg = {"STFullDecoding","ProcessPhase/Emulator" };

Emulator.DetectorList += { "ST" };
EmulatorSTSeq.Members += {
    "STTELL1PedestalSubtractor"
    ,"STTELL1LCMS"
    ,"STTELL1ClusterMaker"
};
```

Another important algorithm is "RawBanktoSTClusterAlg" (also ST/STDAQ). With this algorithm clusters are decoded. To compare TELL1 clusters with Emulator clusters, this algorithm needs to be run twice. Once to decode the clusters in the ZS bank and once to decode the clusters created by the ClusterMaker of the Emulator. For the second, the rawEventLocation needs to be set to "Emu/RawEvent" and as output location you can set the clusterLocation to "Emu/IT/Clusters".

The monitoring algorithm dumps all information on clusters and/or nzs information in a large NTuple. This NTuple can contain two TTrees, one called 'nzs' and one called 'zs' (both can be switched on/off). If you choose also to monitor the nzs information, you should put the output in a /tmp location. Here follow the options which can be set.

Job Options

The jobOptions which can be set in the Emulator are:

STTELL1PedestalSubtractor	STTELL1LCMS	STTELL1ClusterM
InputDataLoc (STTELL1DataLocation::TTFull)	InputDataLoc (STTELL1DataLocation::TTPedSubADCs)	InputDataLoc (STTELL1DataLocation::TT

`OutputDataLoc`
`(STTELL1DataLocation::TTPedSubADCs)`
`SubPedsLoc`
`(STTELL1DataLocation::TTSubPeds)`
`TELL1PedestalBankLoc`
`(STTELL1DataLocation::TTPedestal)`
`DetType ("TT")`
`CondPath ("CondDB")`
`UseTELL1PedestalBank (false)`
	`OutputDataLoc`
	`(STTELL1DataLocation::TTLCMSADCs)`
	
	`DetType ("TT")`
	`CondPath ("CondDB")`
	
	`ConvergenceLimit (0)`
		
		`DetType ("TT")`
		`CondPath ("CondDB")`
		
		`ConvergenceLimit (0)`

The ConvergenceLimit is the number of events used to calculate a good pedestal. For these events only the Pedestal Subtraction algorithm will run. Only when the ConvergenceLimit is reached will the CMS and the ClusterMaker run. (Example: If a strip is off, i.e. it always has ADC 0 and we start with subtracting the default value of 128 (see next section), then it takes the Pedestal Update about 12000 events to get the pedestal to the right value of 0.)

For the monitoring: IncludeZS and IncludeNZS refer to the TTrees you'd like in the NTuple.

Configuration Parameters in the Conditions Database

This section explains which parameters are in the conditions database. For information on how to set these parameters, please read the section "Setting/Changing the conditions database parameters".

Each condition (or TELL1Board) contains the following parameters (specified per algorithm). **These are their names as they are in the conditions database.**

STTELL1PedestalSubtractor

`Header_correction_analog_link_00, size:4`
`Header_correction_analog_link_01, size:4`
`...`
`Header_correction_analog_link_95, size:4`
`Header_threshold, size:2`
`Pedestal_mask, size:3072`
`Pedestal_value, size:3072`
`Pedestal_enable`
`Update_enable`
`ZS_enable`
`Header_enable`

The Pedestal_mask is a vector of size 3072, which entries can be set to either 0 or 1. 1 meaning that a certain strip is masked (ADC set to 0). The Pedestal_value contains the initial pedestals to be subtracted per link. To enable the pedestal subtraction, both Pedestal_enable and ZS_enable must be set to 1. Pedestal Following is enabled by setting Update_enable to 1. Although Header_enable and Header_correction_analog_link_## are in the list, the header correction is currently being revised and not used at the moment.

STTELL1LCMS

`CMS_threshold, size:3072`
`CMS_enable`

To enable the CMS process set CMS_enable to 1. If the CMS is disabled, all data is directly copied and supplied to the ClusterMaker algorithm. The CMS Thresholds per strip are set in the CMS_threshold vector.

STTELL1ClusterMaker
`Hit_threshold, size:3072`
`Confirmation_threshold, size:48`
`SpillOver_threshold, size: 48`
`PP_max_clusters`
`ZS_enable`
`Disable_beetles_0_11, size:12`
`Disable_beetles_12_23, size:12`
`Disable_links_per_beetle_0_11, size:12`
`Disable_links_per_beetle_12_23, size:12`

To enable the ZS process set ZS_enable to 1. There are three thresholds used in the ZS process: The Hit_threshold is per individual strip and thus the charge of each strip in a cluster needs to exceed this threshold. The total charge of the cluster needs to exceed the Confirmation_threshold. When the total charge of the clusters is higher than the SpillOver_threshold the SpillOver bit is set. Both Confirmation_threshold as SpillOver_threshold can be set per processing channel (64 strips), but clusters are searched for per beetle (128 strips). The 'Disable_beetles' are simply set with either 1 or 0. However, the 'Disable_links_per_beetle' needs a number between 0 (no links disabled) and F (all links disabled).

Setting up the Conditions Database

In order to use the database, Vetra v7r0 needs to be installed: **getpack Tell1/Vetra v7r0**. (or there may even be a more recent version). There are 2 steps we need to take. First, we need to create our own file with configuration parameters. This is done by going to the *python* folder and running:

python write_it_xml_cond.py (or tt)

This is however a standard file without the latest settings in them. For a file with the latest settings contact me. After this the conditions database can be set up; go to the *scripts* folder and run:

source create_it_sqlite_file_from_xml.sh (or tt)

this creates a conditions database COND.db in the location: VetraCondDB/IT (or TT). This file works as a layer on top of the general database (which you don't have to install!) The program will now first look for conditions in this file before it searches in the general database. A more detailed description is given by Tomasz on the VetraHowTo pages. The new conditions path is default set to "CondDB" in the ST/STTELL1Algorithms and won't need changing (for both IT and TT).

To use this conditions database COND.db, we have to add the following lines to the jobOptions:

```
CondDBCnvSvc.CondDBReader="CondDBLayeringSvc";
CondDBLayeringSvc.Layers={'CondDBAccessSvc/COND', 'CondDBDispatcherSvc'};
COND.ConnectionString="sqlite_file:$VETRAROOT/VetraCondDB/IT/COND.db/COND";
```

For using the TT database, "IT" needs to be substituted to "TT" in the last line.

Setting/Changing the conditions database parameters

We can set all parameters without changing the conditions database. To set a parameter, add a line like the following to your jobOptions

ConditionsOverride += {"CondDB/TELL1Board10 := int Pedestal_enable = 1;"};

In order to change many parameters, for instance of all TELL1Boards or just the 3072 values of the Hit_threshold, we can use a python script. In the folder *Tell1/Vetra/v6r1/options* there are two files **runSTEmulator.opts** and **runSTEmulator.py** to demonstrate how to use python to add lines to the normal options file.

It is also possible to change line in the actual TELL1Cond.xml which is created by running write_it_xml_cond.py or write_tt_xml_cond.py (as explained above). But then you would need to recreate a new conditions database everytime, which means running the source file create_it_sqlite_file_from_xml.sh or create_tt_sqlite_file_from_xml.sh.
