NATIONAL
OPTICAL
ASTRONOMY
OBSERVATORY

SYSTEM INSTRUMENTATION GROUP
950 N. Cherry Ave.
P. O. Box 26732
Tucson, Arizona 85726-6732
(520) 318-8000 FAX: (520) 318-8303

`

TORRENT
Software User Manual
assimilate

NOAO Document TRNT-AD-08-0003

Revision: 1.2

Authored by:
Nick C. Buchholz

Please send comments:
nbuchholz@noao.edu

Revision History

Version	Date Approved	Sections Affected	Remarks
0.1	7/9/2009	All	Initial draft release - aro
1.0	20091111	All	Added Appendices, Introduction and rearranged sections - ncb
1.1		All, 5.1 & Appendices	Made typo corrections revised version number section, added four new signal types - ncb
1.1	20100318	Appendix I.ii & I.iv	Added **DFLTVAL** and **UNITS** fields - ncb
1.2	20111108	All	Revised document to reflect current version

Table of Contents

List of Figures

1.0 Introduction

This document describes the design of the Torrent assimilate program. the assimilate program reads the ".vhd" files that are used to create the Torrent FPGA firmware and creates a description of the attributes, assigns attribute information to locations in the module EEPROMs for each hardware module and creates a ".cfg" file description for each firmware module EEPROM and a template ".csv" file that can be read by the MONSOON pan processes to define how the DHE is controlled.

The collector program (***collector***) will use these files and connect to the DHE through a minimal panDaemon and determine if the DHE is a MONSOON "orange" or a Torrent system. If it's an "orange" system the collector will shut down and hand off execution to a standard MONSOON client. If the DHE is a Torrent system the collector will obtain the ".cfg" description files for the firmware modules and combine the ".cfg" information with the data stored in the EEPROM(s) on the DHE and create a Config.csv file for the system being started. This .csv file will be customized with the correct calibration constants for all the voltage and telemetry attributes and with the max and min voltages that can be obtained from the hardware. The ***collector*** program will then read the TSM (Transition Module) ".cfg" file and the TSM EEPROM to obtain the default, maximum, minimum values for this Dewar and focal plane. Using this information ***collector*** will create a *system*_DefaultSetup.mod file that will be used by the MONSOON client to initialize the detector.

2.0 Torrent Auto-configuration

2.1. assimilate

The ***assimilate*** program will read the FPGA project build file for the current firmware version and extract comment lines of a particular format. The program will interpret the lines and create a set of ".cfg" files, one for each module described in the comment lines. These ".cfg" files and a software ".cfg" are then used to create a template "Tmplt.csv" file. These ".cfg" and "Tmplt.csv" are unique to each version of the firmware running in a Torrent system. These files will be identified with the Version numbers of the VHDL files used to generate the file.

At the system startup the ".cfg files for the SFTW, LCB, PSM and CCD[1,2] or IRFE[1,2] and TSM modules, and the LCB, PSM, CCD[1,2] or IRFE[1,2] and TSM EEPROMS will be read and a ".csv" file containing the required attributes will created. The TSM.cfg file and TSM EEPROM will be used to create a *system*_DefaultSetup.mod file to be loaded at runtime to bring the detectors to their correct operating levels.

2.2. collector

At runtime a second program called ***collector*** will read these files and use them to obtain information from the EEPROMs associated with each module. This information will then be combined with the "Tmplt.csv" to create a systemName_config.csv that is read by the pan processes to configure and control the detector.

In addition a separate torrent_TSM.cfg file is created during the system configuration and detector optimization process. This file is used by ***collector*** to read the default, maximum and minimum values for every hardware function connected to the Dewar from the TSM module's EEPROM and create a *system*_DefaultSetup.mod file to be used by the initialization routines at system startup to set all attributes to optimum levels. See Section 3.0 for a detailed description of this process

3.0 Using *assimilate*

The ***assimilate*** program is started in Linux by typing a command line in an xterm. The ***assimilate*** command line takes the following optional arguments (defaults in parentheses) : Note the argument and its value are separated by an ASCII space.

```
-help                print help and exit
-version <verFPGA>   The version number of FPGA to use (210)
-sysName <sysName>   Create cfg & csv files for this system (basicCCD)
-fromCfg <fromCfg>   Start from cfg files for version not vhdl (False)
-startDir <startDir> The starting directory for FPGA search
                              (/MNSN/engr_Development/Torrent/Xilinx/SystemBuilds)
-stdout              print messages rather than display in label (True)
```

assimilate is a mostly automated program that gives the user a chance to verify the program's operation. To use ***assimilate*** at a linux prompt type: (a MS windows version may be provided at a later date.) If you are going to use version 210 (the default and the default FPGA code directory you can just type

```
assimilate
```

or if you are building for another version or from another directory type

```
assimilate -version ### -startdir diectoryName
```

If assimilate has been run previously you can short circuit part of the process by typing

```
assimilate -version ### -fromCfg True
```

This will use the previously produced .cfg files stored in ${MONSOON_CFG}/_common.

After the user presses return on the command line the main ***assimilate*** graphical user interface should appear looking like Figure 1

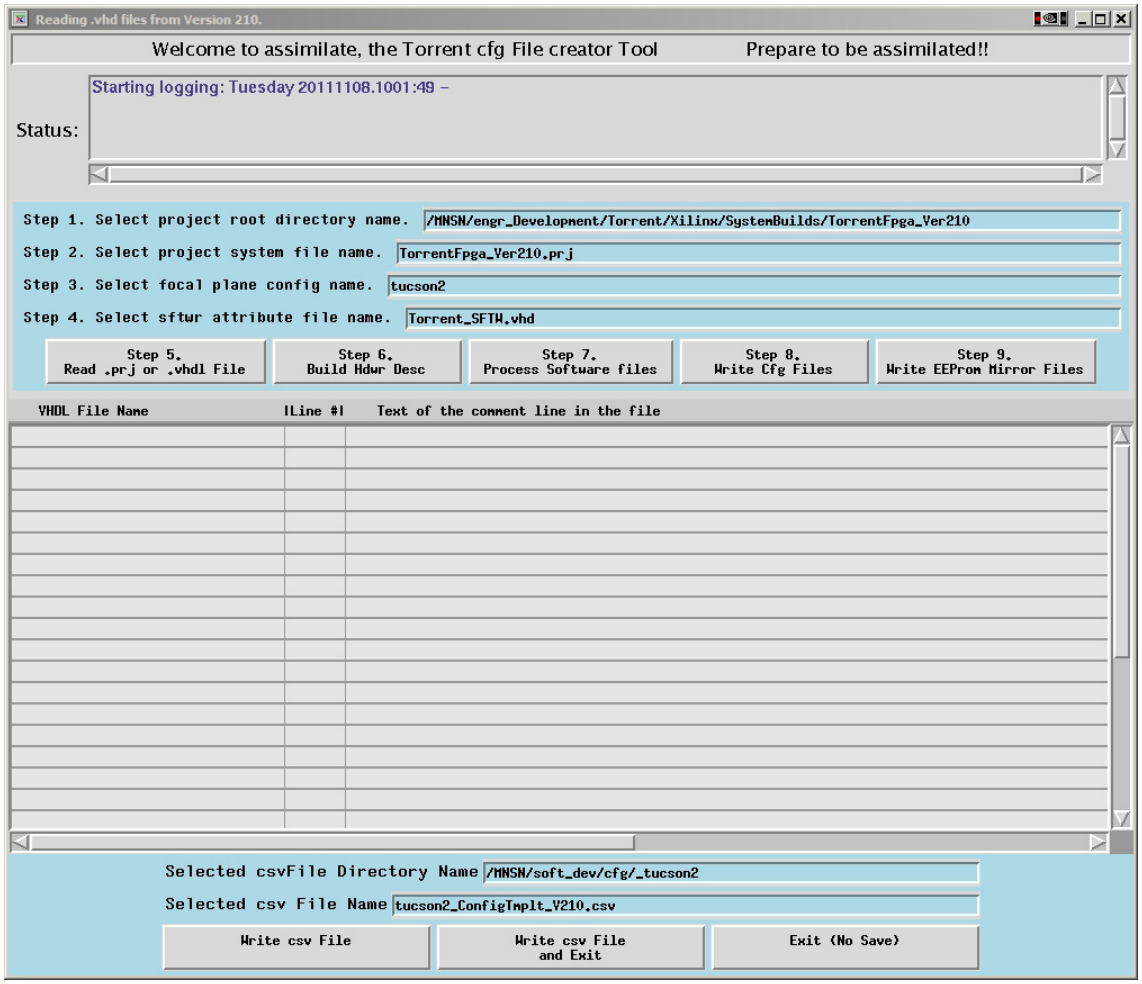

Figure 1. Main interaction screen

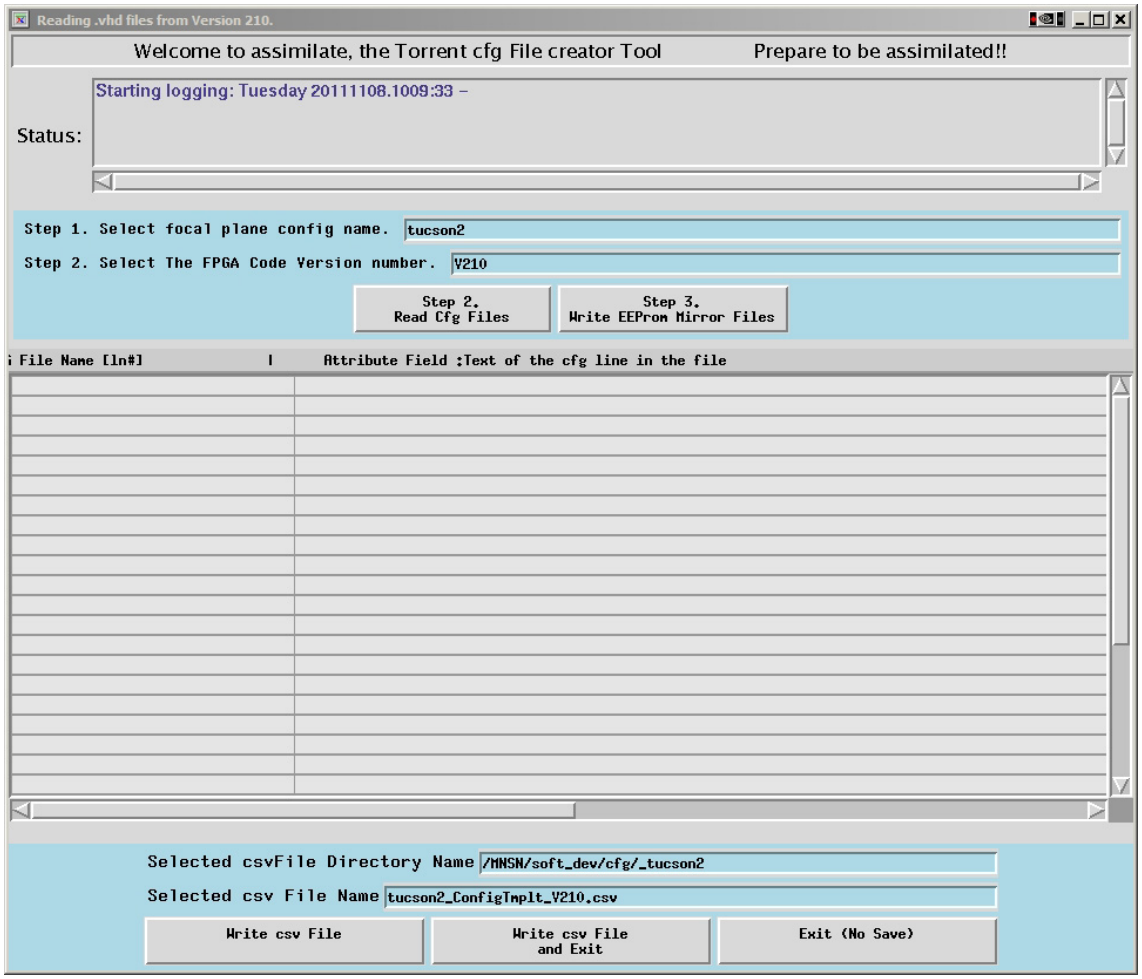

Figure 2. Main screen when using previously created .cfg files

3.1. Assimilate GUI interaction Areas.

The main interaction screen consists of several sections. The purpose and usage of each section is explained below. Note that closing the main screen without saving files will cause the loss of the current information. Since the information is automatically generated it can be easily recovered by re-running the ***assimilate*** program.

3.1.1. The Status area

The status area of the window will display all of the error, warning and informational messages produced by the assimilate program (see Figure 3). The information displayed here is also written to a log file in the ${MONSOON_CFG} directory as assimilateLog. A review of this file can aid in resolving error messages generated by the program. All messages sent to the status area are always also written to the log file.

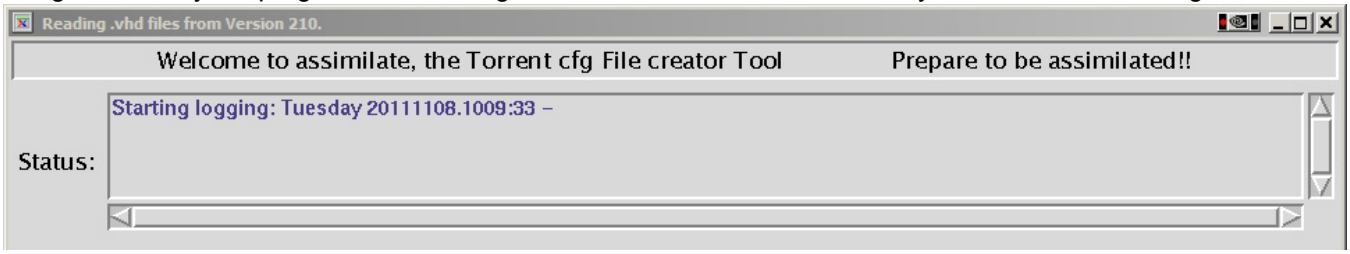

Figure 3. The Status area

3.1.2. The File Area/

The file selection area changes depending on how the program was started. In the normal start up using the FPGA code files to create the .cfg and other files the file area will look like Figure 4. When starting from the archived .cfg files the File Area will look like Figure 5

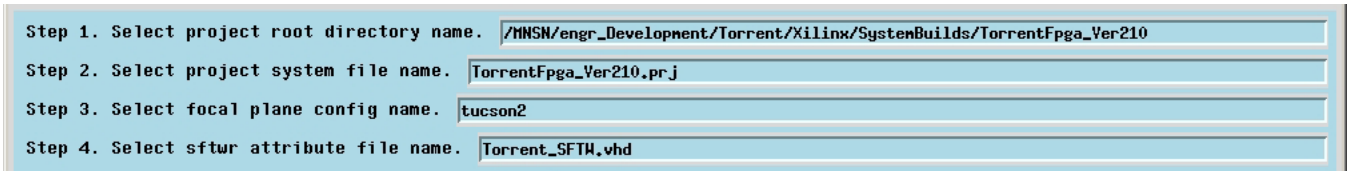

Figure 4. File Area when starting from FPGA code files (.vhd)

The steps listed in figure 4 should be reviewed in order. In general the default values will be correct.

Step 1. allows the user to choose the starting point for the assimilate process and the name (version) of the FPGA code to assimilate. Doing a left or right click with the mouse in the Step 1 Root Directory entry field will bring up a file dialog (see Figure V.) that allows you to select the directory containing the project file for the FPGA code.

Step 2. selects the name of the project file to use this should have the same version number as the start directory.

Step 3. chooses the name of the system (focal plane) directory that will contain the .configuration template (.csv) and eeprom mirror (.eep) Files. The system will create the directory if it does not exist. The directory will be created in ${MONSOON_CFG} and will have the name _sysName, where sysname is the value entered in step 3.

Step 4. Allows the user to speify an alternate software ,vhd file to use to create the software entries in the .cfg files the standard software file is found in ${MONSOON_CFG}/Torrent_SFTW.vhd.

The Files area is much simpler when starting from the .cfg file created by a previous run of the assimilate program. Step 3 above becomes step 1 and have the same meaning and result. A new step 2 is used to identify the versin of the archived .cfg files that will be used in creating the .csv and .eep files.

Figure 5. File Area when starting from the archived (.cfg) files.

Likewise doing a left or right click with the mouse in the Step 1 or Step 2 entry areas will bring up a file dialog that allows the user to select the directory or file without typeing in the name.

Note that the template ".csv" file created by the assimilate process is for review and testing only The ***collector*** program creates the .arr, .ini. .csv and .mod files at run time from information stored in .cfg files and EEPROMs in the DHE modules.

3.1.3. The Top Control button area

The top control button area when assimilating from the .vhd Code files are layed out as in Figure 6). These buttons provide the control for the assimilate process. The buttons need to be pressed in order and allow the user to review the results at each step. This review has been found to be useful in the case of format errors in the FPGA code comments that assimilate uses to create the .cfg files.

Figure 6. The Top Control Buttons processing FPGA code files

Step 5 causes the selected project (.prj) file to be read and a list of the files to be processed displayed (see Figure 8). Scroll through this list to be sure the correct file versions were found.

Step 6 builds the hardware description in memory from the FPGA code files.

Step 7 adds the software information in the two software .vhd files (Torrent_STFW.vhd and Torrent_TSM.vhd). These files contain no vhdl code but only include comments to be read by assimilate and converted into .cfg file lines. The SFTW fiel include descriptions of all the software attributes used by the PAN software. The TSM file includes the names of hardware attributes

located in other modules whose default runtime values will be stored in the TSM module attached to the system Dewar

Step 8 causes the memory structures created thus far to be written to the disk as .cfg files.

Step 9 causes the eeprom layout for each module to be written to the disk as .eep files in the system configuration directory. The values in these files are the approximate values calculated from the hardware design they will be replaced as needed during the board calibration and testing phase of the production process for new Torrent systems.

The Control buttons for the .cfg starting point include only a button to read the .cfg files and a button to write the eeproms to the system configuration directory.

Figure 7. The Top Control Buttons processing .cfg files

Figure 8. The ".vhd" File list

3.1.4. The Context Area

When the step 6 and step 7 buttons have been pushed, the comment lines in the FPGA code are displayed in the Context area as shown in Figure 9. The context area contains three types of lines at this time "blue" background lines are lines indicating the start of a new FPGA module indicated by a line starting with "#MD". Dark gray background lines are attribute description comment lines that are incomplete i.e. missing requied fields. Light gray background lines are attribute description comment lines that are complete.

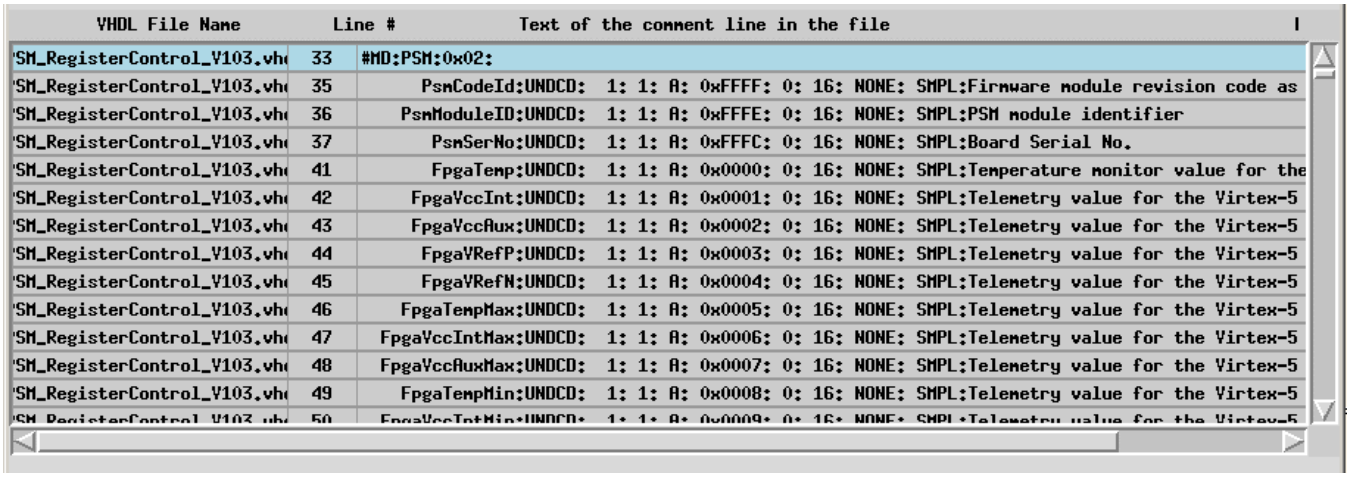

Figure 9. The Context Area

As a result of the build process there are likely to be messages in the status area indicating problems with the vhdl file comments. Warnings are shown in orange and usually indicate either a missing field in the comment or attributes with duplicate addresses in the module/address field. These warnings can be ignored as the missing fields are filled in with “NoEntry” or a blank and the duplicate addresses generalyy indicate read only or write only attributes with different names but the same address. However the engineer in charge of the non-compliant file should be informed so that the comment can be corrected or the duplicate addresses verified as not interfering with each other. Eventually the Torrent systems will be depending on complete and accurate information from these VHDL comments. (See Figure X below for an example of a warning message)

Figure 10. Example incomplete comment Warning

Figure 11. Example duplicate address Warning

The list of files processed and warning is contained in the assimilateLog file and can be used to inform the responsible engineer of comment errors.

Lines preceded by a green background line indicating the module being written are the l;ines that will apear in the .cfg files and include the EEPROM address in the appropriate EEPROM for each attribute

3.1.5. Bottom Controls and file entries

When all the steps of the creation process are complete the user can choose a directory and file name for a template .csv file using the file entry areas at the bottom of the screen (see Figure 12). This tmplt file is is usable for engineering tests of the system created. Note that this .csv file contains hardware function names and approximate conversion factors. This means that voltages are set to approximately correct values. Only after the calibration and test process is complete for all the boards and the collector program run will the sysName_Config.csv file contain the correct conversion factors.

Figure 12. Bottom Buttons and file controls

3.2. The *sysConfig, mborg* and *collector* programs

The files produced by the ***assimilate*** program are intended for use by the ***sysConfig, borg and collector*** programs.

sysConfig is a semi-automatic engineering tool used to describe and document a system, its wire connections and attribute usage and to store that information in the _Torrent_Systems documentation area.

borg uses the files in it’s the test routines to determine the location in the EEPROMS for the final calibration factors for each hardware attribute.

collector uses the module .cfg Files and information stored in the module EEPROMs and TSM Module EEPROM to create the files required by the PAN software to safely run a focal plane system. ***collector*** runs each time a system is started creating a new .csv file that incorporates the latest information stored in the

EEPROMS. This means that if a replacement board or DHE is attached to a Torrent system Dewar the system will automatically be able to set voltages to the proper levels based on the module EEPROMS.

sysConfig, mborg and ***collector*** are described in TRNT-AD-08-0006-R1-SYSCFG.doc, TRNT-AD-08-0002-R1-BORG.doc and TRNT-AD-08-0004.R0CLCTR.doc respectively.

Appendix I Auto Configuration Comment Lines

As explained above the automatic configuration process uses comments in the VHDL source files to determine which functions and registers are available in the DHE hardware. This appendix describes the format and content of those comments.

I.i Firmware Source Code (.vhd files -> ".cfg" Files

This section will describe the format of a set of the FPGA source code comments required to describe the hardware. The tool will read these comments, parse the lines and generate a text file that describes the hardware. Other tools will use these files and the data stored in the EEPROMS to create the .csv file that will be read by the panDaemon code to set up the attributes in the software.

The ***assimilate*** tool will generate a .cfg file for each module described in the firmware comments. Since the FPGA source code files will be read in an arbitrary order to distinguish which EEPROM/module a register belongs to, we will include a comment line of the form:

#MD:EPD:Address: comment

where:

"– #MD:" - indicates this is an EEPROM address description line

"EPD" - is a module name i.e. one of LCB, PSM, CCD#, CCD1, CCD2, IRFE#, IRFE1, IRFE2 the CCD# and IRFE# module names indicate registers that are common to all CCD or IR AFE boards

"Address" - is the address of the Module 0x01 – 0x80, one bit set.

"comment " - is a comment for the engineer's use

FPGA register descriptions following the MD description line are allocated to the "EPD" EEPROM described by the MD line. Each new register description (#AD line) will be added to this EEPROM until a new "— #MD" line is found.

I.ii ".vhd" file FPGA Register Description Comment lines

This section covers comment lines for LCB, PSM, CFG, PIX, AFE and SYS modules. Note that the .cfg files produced by assimilate have the **AttrName** and **EPAdr** fields reversed.

Each register description line will be a comment in the FPGA source code file of the form below. A comment line may be included in the file as a column header line. The column header comment line will look like this:

--##AD:EPAdr:AtrName:ArraySz:#Vals:TypCde:BdAdr:RegAdr:AdrIncr:Size:SetMthd:RdMthd:SigTyp:DspGrp:dfltVal:Cmnt

Normal register description lines will have only one '#' character after the '–' at the start of the line as in '--#AD … A normal #AD line will look like this: eg.

-- #AD:128:ColIncValue:1:4:B:0x02:0x1234:1:SMPL:SMPL:LG:3:2.4:Signed value added to image address counter for pixel transfers.

where:

"--##AD" is a register description comment line and

"-- #AD" is the tag indication for a normal register description line

1. **EEPAdr** – This field will contain the starting word address (0-4095) in the EEPROM of the configuration information for this attribute. Acceptable values are an address starting at 0xFFF down to 0x000 in hex or decimal format or the words "AUTO" ,"UNDCD" or NONE**.** The "AUTO" / "UNDCD" words will indicate to ***assimilate*** that the location of the configuration data block for this attribute should be assigned automatically. The NONE word indicates to ***assimilate*** that no EEPROM words should be assigned to this attribute.

2. **AttrName** – The name used/defined in the FPGA. This is the DHE hardware name of the register/function.

3. **ArraySz** – the number of Registers in an array of registers with a common name and purpose. This can always be 1 treating each register in an array as a single register. The PAN already always unwraps arrays of registers to single registers.

4. **#Vals** – gives the size in words of the values needed to describe the behavior of the register. The values occupy EEPAdr → EEPAdr+(#Values-1)) in the EEPROM. These words contain the values described by TypCde below.

5. **TypCde** – a letter designating the layout of the attribute description in the EEPROM as follows:

   A – A single integer value entry representing information about the EEPROM structure or provenance; or
   a single default FLOAT*1000 value stored as a 32 BIT integer value to be used in a .ini or .mod file created by the collector;

   B –Four integer values that describe a bit field register. Laid out as >Width, Mask, Max, Min< The Bit field is always assumed to be located in bits 0 → width. NOT USED

   C – Four Floating point values that describe a value register calibration. Laid out as >Slope,Intercept,Min,Max <

   D – Four Floating point values that describe a value register calibration. Laid out as >Slope,Intercept,Min,Max < Plus a default value and Min, Max stored in the TSM eeprom

   E – A Four word entry representing a checksum value least significant word first.

   F – A single floating point value used in a software constant

   G – Default, Min, Max values stored in the TSM eeprom.

   ~~H – A word (4 Character) string entry in the TSM module #Vals tells how many words the string spans a 00 character ends the string NOT USED~~

   Note: types A, B, C, D all create new attributes and reserve space in the TSM eeprom. Types E, F, G only reserve space in the TSM eeprom and may modifiy previous attribute Entries.

6. **MdlName** – the text name for the board/module containing the Register e.g., CCD1, LCB

7. **RegAddr:** the 16 bit address of the register in the module address space for the first register in the group. The PAN will treat BdAddr and RegAddress: fields as a single address by summing the two fields with DheRegAddress = ((BdAddr<<16)&RegAddr+AdrIncr()). The csv file will place a !BDNAME:0x04 line at the beginning of the file and describe the address as ,BDNAME:0234, in the register address field.

8. **AdrIncr** – A code indicating how addresses increment in arrays of registers. Three values are recognized at this time.

   0, 1 – indicate the addresses increment by 1 for each successive index i.e. 1 2 3 4 5, etc.

   2 - indicates the addresses increment by shifting the lower #inArray bits by one bit to the left. i.e. 0x1101, 0x1102, 0x1104, 0x1108, 0x1110, etc.

9. **Size** – The width in bits of the register, usually 32. Reasonable values are 1, 8, 12, 16, 24, and 32.

10. **SetMthd,** The method used to write the register. Three methods are known:

NONE – The attribute cannot be accessed for writing.

SMPL – The register is accessed with a simple write command.

RMSKW – The register is a bit field that requires a read, masking the bits that are unchanged and a simple write to set the register.

Any of the methods acceptable for a csv file line are also accepted here including RDMSKWRT and INTTIME etc.z

11. **RdMthd** – The method used to write/read the register. Three methods are known:

NONE – The attribute cannot be accessed for zreading.

SMPL – The register is accessed with a simple write or read command.

RMSKW – The register is a bit field that requires a read, masking the bits that are unchanged and a simple write to set the register. On read the value returned is the value of the bits requested.

Any of the methods acceptable for a csv file line are also accepted here including RDMSKWRT and INTTIME etc.

12. **SigTyp** – This gives the type of signal for connection purposes. The sysConfig routine will use this to limit the connection type of detector signals for this attribute. sysConfig will recognize the following signal types. Note: These types refer to the connections to the detector or Dewar and are not used for internal or logical function attributes. These attributes should be given the NONE, LGCL or SFTWR signal type.

DPWR – a detector power signal – a voltage usually connected to a fixed bias level.

DCLK – detector clock signal – a control clock for the detector. These break down into an UpperRail, a LowerRail and a logical level. Connecting any one of the three signals to a detector pin automatically connects the other two signals.

AGND – a detector analog ground signal –

OGND – other detector ground signals –

VOUT – a detector video output. This breaks out into the video signal and an offset voltage for that signal.

BIAS - a detector bias voltage.

DVLT – a detector voltage not otherwise classified (e.g. logical controls as in OTA's. or hardware gain controls).

NONE – an attribute that does not connect to the detector and for which no or only a write/read test exists.

CMND – an attribute that when written to results in the start of some process in the DHE, e.g. afe1Reset, sysReset

TELE – an attribute that reads back the value of some telemetry voltage.

CNFG – a configuration Attribute e.g. Afe1TpGrp2Cfg or CcdAmpCfg

CNST – attributes that contain constants e.g. CodeID, AFE1SerNum, etc.

LGCL - an attribute that is a logical connection to the detector (i.e. rowCount, Number of outputs, pipeline enables, etc.)

TEMP – a temperature control signal.

SFTWR – connects to Dewar only through other values

13. **DspGrp** – a number representing the display group that includes this attribute. Equivalent to the MEC or BORG page number to be used for this attribute's display.

14. **DfltVal** – the value assigned to the attribute register when the fpga is loaded on power-up or when a hard reset occurs.

15. **Units** – the name of the units for this field as in mSeconds, Volts, mAmps, °K, and so forth.

16. **Cmnt** – a comment describing the usage/purpose/function of this register.

I.iii ".vhd" file Description lines for the SFTW module

A special '.vhd' file containing only comment lines will be constructed to allow the software attributes to be added to the Torrent system '.cfg and '.csv' files. The torrent_SFTWR.vhd will contain lines similar to the lines in a standard '.vhd' file as described is Appendix I.ii above. This file will contain entries that go into the TSM EEPROM as well as attributes that use no space in any EEPROM

I.iv Connecting Registers to the Detector Functions.

Using the TSM .cfg File and the TSM EEPROM the software will create the .mod file and csv file for the system. The TSM EEPROM contains the default, maximum and minimum values allowed for each of the detector voltages, clocks and other attributes. Each Detector voltage/clock and so forth is described in the TSM configuration file and is associated with a hardware register described in one of the .cfg files described above. The integrating engineer will use the ***sysConfig*** tool to create a file that connects a hardware register function to a detector requirement.

The TSM.cfg files will be constructed by ***sysConfig*** and have the following format:

AttrName:EEPAdr: :#EepVals:TypCde:UserName:CReg: : : : : Units:DspGrp:DfltVal:HelpText

where:

1. **AttrName** – The name used/defined in the FPGA. This is the DHE hardware name of the register/function or the software name for the attribute. It must be the same as a name given in one of the cfg files above.

2. **EEPAdr** – This field will contain the starting word address (0-4095) in the TSM EEPROM of the configuration information for this attribute. Acceptable values are an address 0x000-0xFFF in hex or decimal format or "AUTO" The "AUTO" code will indicate to ***sysConfig*** that the location of the configuration data block for this attribute will be automatically assigned.

3. ***blank*** – a blank field place holder for consistency with other cfg Files

4. **#EepVals** – gives the size in words of the values needed to describe the behavior of the register. The values occupy EEPAdr → EEPAdr+(#Values-1)) in the EEPROM. These words contain the values described by TypCde below.

5. **TypCde** – a letter designating the layout of the attribute description in the EEPROM:

   A – A single integer value entry representing information about the EEPROM structure or provenance; or
   Not used - a single word engineering value stored as a 32 BIT binary value to be loaded automatically by the FPGA startup code;

   B – Not Used - Four integer values that describe a bit field register. Laid out as >Width, Mask, Max, Min< The Bit field is always assumed to be located in bits 0 → width

C – Four Floating point values that describe a value register calibration. Laid out as >Slope,Intercept,Max,Min<

D – Not Used - Four Floating point values that describe a value register calibration. Laid out as >Slope,Intercept,Max,Min< Plus a default value and tolerance stored in the TSM eeprom

E – A Four word entry representing a checksum value least significant word first.

F – A single floating point value used in a software constant

G – A default value and tolerance stored in the TSM eeprom

H – A word (4 Character) string entry in the TSM module #Vals tells how many words the string spans a 00 character ends the string

6. **UsrName** – is the function name of a detector pin, clk or bias voltage or a user name for other attributes like integration time(intTime).
7. **CReg** – the 32bit register whose bits determine how the user may manipulate the attribute the ***sysConfig*** Tool will provide a checkbox mechanism to construct this field. The value in this field will take precedence over any CREG defined in a hardware or software module cfg file.
8. **– 12.** 5 blank field place holders for consistency with other cfg Files
13. **DspGrp** – a number representing the display group that includes this attribute. Equivalent to the MEC or BORG page number to be used for this attribute's display.
14. **DfltVal** – the default value to be loaded into the EEPROM for this attribute
15. **Units** – the name of the units for this field as in mSeconds, Volts, mAmps, °K, and so forth.
16. **HelpText** – a comment intended to describe the purpose/function of the register from the user's point of view. Eg.:

    This register adjusts the Global Reset voltage for the Aladdin detector.

    This is the voltage applied to the first parallel clock phase for the site1k CCD.

I.v Additional assimilate directives

In addition to the information described above the VHDL comments can contain A "directives" section. These are instructions to the process in ***`assimilate`*** or ***`collector`*** that creates a .csv file. The directives are appended to the comment field and are stripped off in the .csv file.

To create a .csv file the configuration information is processed in three phases. In the first phase each module .cfg file is read and the various modules are merged into a single cfgLine list. In the second phase the combined list is read and the directives are processed and applied. ADD and SUB directives are processed and converted into MOD directives. In the third phase the .csv file lines are created from the cfgLines list.

The format of the directive section of the comments is as follows

^directive^{AttrName}^slope^intercept^minValue^maxValue

where:

directive – is an instruction to assimilate or collector that tells the program what needs to be done. These can be:

- ADD –this directive is used to create new attributes for a system using an existing attribute and giving it a new user attribute name. After this directive is it is converted to a MOD directive
- SUB- this directive is used to change the name of an attribute After this directive is it is converted to a MOD directive

- MOD- these directives are only processed in phase three of the .csv creation process. This directive creates a .csv line using the slope, intercept, minValue and MaxValue fields instead of the hardware BITS values. If there is an optional AttrName in the line, it is used as the user name in the .csv Line.

attrName – this is an optional value if it is not a blank it will be used to search the cfgLine list for an attribute to be used by the ADD or SUB directive or as the user name in the final .csv file line by a MOD directive

slope – the slope of the conversion equation that converts engineering units (volts, amps, etc.) to BIT values to be written to the DHE DACs etc. It is also used in the reverse equation that converts BITS to engineering units.

intercept – The intercept of the conversion equations

minValue – the minimum value permitted allowed for setting the attribute

maxValue - the maximum value permitted allowed for setting the attribute