apeNEXT: Monitoring of Temperature, Voltage and Current

Id: adcmon.tex,v 1.4 2007/05/01 12:41:02 ckuelker Exp

Abstract

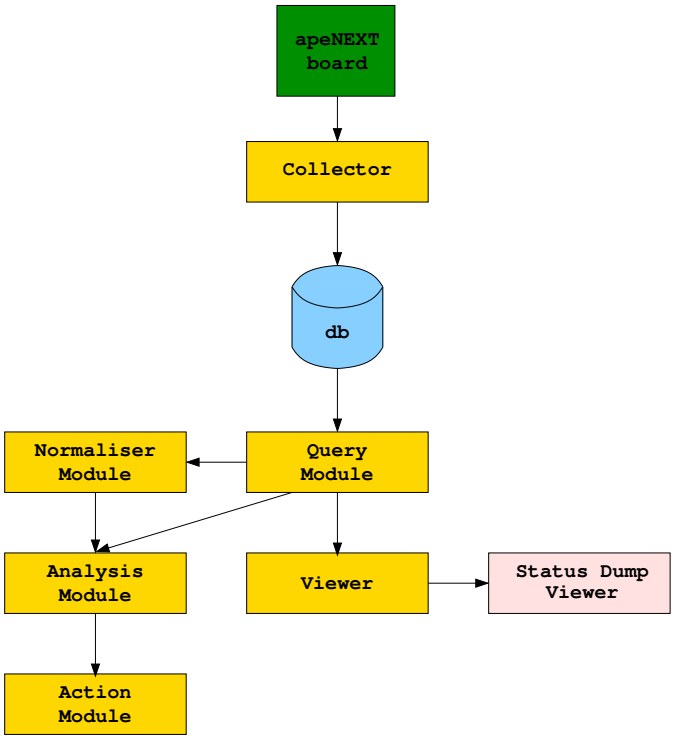
This document specifies a software infrastructure for monitoring the temperature, voltage and currents measured on the apeNEXT boards and documents the implementation.

Contents

1 Architecture

Each of the apeNEXT boards has 7 different Analog to Digital Converters (ADC) which allow to measure temperature, voltage and current on the boards. The 8-bit output of each ADC can be read via the ADC register (`ADCREG`) on the FPGA (one of the so-called `i2c` registers).

The infrastructure which monitors the values measured by the ADCs consists of the following components:



Collector : The collector reads the data periodically from the `ADCREG` on each of the boards and writes per measurement and board an entry into the database.

Query Module : The Query Module provides an interface to the database to query for data records and perform a set of data reduction operations, e.g. calculation of the average, maximum or minimum value for a given set of boards within a certain time window.

Normaliser Module : The Normaliser Module analyses the data for a given time period and partition and provides configuration data which is used by the Analysis Module to normalise the data.

Analysis Module : The Analysis Module reads the collected data periodically via the Query Module and provides qualified status information to the Action Module.

Action Module : Based on the status information provided by the Analysis Module the action module performs a particular action, e.g. notification of the administrators.

Viewer : Command line interface to the Query Module. Reads data records from database and provides differents presentations of the information.

2 Conventions

For argument options of any module the following conventions apply:

board *dddd* : A board is is addressed using 4 decimal numbers. The numbers are interpreted according to there position (0 refers to right-most position):

- 0 : Number of the board within the current unit ($0 \leq d \leq 3$).
- 1 : Number of the unit within the current crate ($0 \leq d \leq 3$).
- 3..2 : Number of the crates ($0 \leq d \leq 99$).

Any leading zeros may be ommitted, e.g. "0001" → "1".

unit *ddd* : Similar to board, but board index is ommitted.

crate *dd* : Similar to unit, but unit index is ommitted.

date : A date may either be defined as a date string
[YYYY-]MM-DD[+hh[:mm[:ss]]]
or number of seconds since the Epoch.

3 Mcollector

The Mcollector uses the Collector to read the `ADCREG` of all boards within a partition that is specified by adcmon.conf. The program has to be periodically invoked via a `crontab` entry. In the remaining document we will refer to the time interval between two consecutive invocations of this program as ΔT_{col}.

The information written to the database by the Collector is defined in the section 5.

3.0.1 mcollector

Multiple Collect apeNEXT ADC register

SYNOPSIS

mcollector –cfg <ADCMON.CFG>

OPTIONS

–collector :

This option can be given to use a different adcmon-collector.

```
-- collector /opt/adcmon-0.7/bin/collector
```

–cfg <FILE> :

To specify a configuration file. The configuration file is inevitable to provide the data for the query of crate, boards and units. This file should not be world readable. And it also should not be on a global readable network volume or similar. If the configuration file is not provided on the command line $NROOT/etc/adcmon.conf will be used. If $NROOT is not defined /nroot/etc/adcmon.conf will be used.

-d :

Same as option –domain.

–debug :

The debug option is mostly for development purpose and can be used to get more informations what the program creates or collects.

–domain :

If there is a FQDN defined on the system, use not only the host part of the name also use the domain name for searching the entry in adcmon.conf. Default is to search only for the hostname. Searching for the hostname only is considered to be more portable.

-f :

Same as option –cfg.

–help :

Print a brief help message and exits.

–man :

Prints the manual page and exits.

-v :

Same as option –verbose

–verbose :

Prints some basic information. To see more use the –debug option.

EXAMPLES

```
Example (1):

 define B<mcollector> hash in /etc/adcmon.cfg

 %mcollector = (
             apemaster5 => {
                     DNOSE => ["/nroot/opt/dnose-1.6/"],
                     dcfg  => ["/nroot/etc/dnose.conf"],
                     crate => [ 5 ]
                            }
            )

 Then run the command:

 B<mcollector> --cfg /etc/adcmon.cfg

 This will query ADC register of all boards of crate 5
 (if you run that on host apemaster5)

Example (2):

 define B<mcollector> hash in /etc/adcmon.cfg

 %mcollector = (
             apemaster5 => {
                     DNOSE => ["/nroot/opt/dnose-1.6/"],
                     dcfg  => ["/nroot/etc/dnose.conf"],
                     crate => [ 5, 6 ]
                            }
            )

 Then run the command:

 B<mcollector> --cfg /etc/adcmon.cfg"

 This will query ADC register of all board of
 crate 5 and crate 6
```

DESCRIPTION

The **mcollector** is intended for starting multiple collectors from crond. It function as a wrapper to collector. The **mcollector** can use the command line –collector switch to access a different collector. Per default **mcollector** uses the $NROOT/sbin/adcmon-collector-__version__ for collecting.

To perform this task the **mcollector** uses a hash called "%mcollector" inside adcmon.conf. If the host name of the current system matches an entry in %mcollector the query will be performed depending the entires of %host in **mcollector**.

Be aware that the host name entry might be with or without a domain. On default no domain is used. But if you would like to distinguish domains use the –domain command line option and define the host entry inside %mcollector with domain.

```
Without domain:      With domain:
apemaster5           apemaster5.ifh.de
```

Configuration File :

The add-on for **mcollector** in adcmon.conf is the definition of the hash %mcollector. This hash can be defined as follows:

```
level0      | level1    | level2       | level3
varname     | hashvalue | hashvalue    | arrayvalues
============+===========+==============+==================
%mcollector | host      | cli option   | cli option value
            |           | env variable | env variable value

cli = command line interface
env = environment
```

Example:

```
host                = apemaster5
cli option          = --dcfg
env variable        = DNOSE
cli option value    = /nroot/etc/dnose.conf
env variable value  = /nroot/opt/dnose-1.6/
```

Every CLI option of collector can be used. But not every option make sense, like h, help, man.

You can define one or more values. If you define more then one value, it should match the number of modalities. A modality can be crate, unit or board. If the number to not match, every not existing value will be taken from the first element.

Example:

```
If you query 2 crates on one host you can define [A] one value
for both crates or [B] 2 values for each crate.

 Example [A]:

     DNOSE =>[qw(/dnose-1.6)],
     crate => [ 4, 6,],

   Then the following crates with their DNSO  directory:

     crate   DNOSE
      4       /dnose-1.6
      6       /dnose-1.6

 Example [B]:

     DNOSE => [qw(/dnose-1.4 /dnose-1.5 /dnose-1.6)],
     crate => [ 4, 5, 6, 7],
```

```
The following DNOSE environment directory will be used:

  crate   DNOSE
   4       /dnose-1.4
   5       /dnose-1.5
   6       /dnose-1.6
   7       /dnose-1.4
```

Configuration File Syntax :

The syntax for the configuration file is Perl. The following examples are equal:

```
%mcollector = (
            apemaster5 => {
                    DNOSE => ["/nroot/opt/dnose-1.6/"],
                    dcfg  => ["/nroot/etc/dnose.conf"],
                    crate => [ 5, 6 ]
                              }
          );

%mcollector = (
            apemaster5 => {
                    DNOSE => [ qw(/nroot/opt/dnose-1.6/) ],
                    dcfg  => [ qw(/nroot/etc/dnose.conf) ],
                    crate => [ 5, 6 ]
                           }
              );
```

The next examples defines different DNOSE dirs for crate 5 and 6:

```
%mcollector = (
  apemaster5 => {
    DNOSE => [ qw(/nroot/opt/dnose-1.4/ /nroot/opt/dnose-1.6/) ],
    dcfg  => [ qw(/nroot/etc/dnose.conf) ],
    crate => [ 5, 6 ]
                }
              );
```

The next example defines a host entry with domain:

```
%mcollector = (
  "apemaster5.ifh.de" => {
     DNOSE => [ qw(/nroot/opt/dnose-1.4/ /nroot/opt/dnose-1.6/) ],
     dcfg  => [ qw(/nroot/etc/dnose.conf) ],
     crate => [ 5, 6 ]
                          }
              );
```

In abstract this means:

```
%mcollector = (
    HOST1 => {
               OPT1  => [qw( VAL1 VAL2 ... VALn)],
               OPT2  => [qw( VAL1 VAL2 ... VALn)],
               ...
               OPTm  => [qw( VAL1 VAL2 ... VALn)],
               crate => [ 1, 2, ..., n ]
             }
    HOST2 => {
```

```
                OPT1  => [qw( VAL1 VAL2 ... VALn)],
                OPT2  => [qw( VAL1 VAL2 ... VALn)],
                ...
                OPTm  => [qw( VAL1 VAL2 ... VALn)],
                crate => [ 1, 2, ..., n ]
              }
    ...
    HOSTi => {
                OPT1  => [qw( VAL1 VAL2 ... VALn)],
                OPT2  => [qw( VAL1 VAL2 ... VALn)],
                ...
                OPTm  => [qw( VAL1 VAL2 ... VALn)],
                crate => [ 1, 2, ..., n ]
              }

  );
```

The ";" at the end is important.

FUNCTIONS

The following functions are implemented in collector for internal use:

cli_param Compute a command line double:

```
VAR=VALUE
-Single_Letter_Option VALUE
--Multi_Letter_Option VALUE
-Single_Letter_Option
--Multi_Letter_Option
```

And give back a reference to an arry, where the cli option-value pairs are stored.

```
$cli_double_aref =
       cli_param($option, $value, $pre_init_array_ref);
```

query_obj

```
$command_aref = query_obj($host,$objectype);
```

debug

```
&debug($message . "\n");
```

```
&debug($format. "\n",$value);
```

SEE ALSO

See the Online Documentation of ADCmon.

AUTHOR

Christian Kuelker <christian.kuelker@desy.de >

4 Collector

The Collector reads the `ADCREG` of all boards within a partition that is specified by the command line arguments. The program has to be periodically invoked via a `crontab` entry. If no time is given on the command line the time interval between two consecutive invocations of this program as can be seen as ΔT_{col}.

The information written to the database is defined in the section 5.

The Collector is a simplified version of `dnosem`. It connects directly to `dnoses` to read information from the `ADCREG`.

4.1 Options

`--board` *N* Read `ADCREG` on board *N*.

`--cfg` *file* Use *file* as configuration instead of default configuration file.

`--crate` *N* Read `ADCREG` on all boards of crate *N*.

`--unit` *N* Read `ADCREG` on all boards of unit *N*.

Other configuration options are defined in the configuration file (see 11).

4.2 Error handling

The Collector does not provide any mechanism to warn the administrators. The Analysis Modules may detect situations where no data is written to the database.

4.3 Manual Page

4.3.1 collector

Collect apeNEXT ADC register part

SYNOPSIS

DNOSE=/path/to/dnose **collector** [options] -dcfg /path/to/dnose.conf

```
Mandatory Options (choose only one Option!):
 -c <CID> | --crate <CID>         crate number from 0 ... 127
 -u <UID> | --unit  <UID>         unit number from 0 ... 3
 -b <BID> | --board <BID>         board ID from 0 ... 3

Other Options:
 -D        | --debug            prints debuging information
 -d <FILE> | --dcfg <FILE>      Dnose configuration file
 -f <FILE> | --cfg <FILE>       config file
 --help                         brief help message
 --man                          full documentation\
 -m        | --mask             define adc regsiter mask
                                to be stored in database
```

```
 -n        | --noquery          do anything else but do
                                not query the temperature
 -v        | --verbose          prints ADC register in
                                hexadecimal format
 -e        | --epoch            use this as a time string
                                for DB storage

Examples:
 (1) B<collector> --board 833
 (2) DNOSE=/nroot/opt/dnose-1.4/ B<collector> \
     --cfg /etc/adcmon.cfg \
     --dcfg /nroot/etc/dnose.conf --crate 5
 (3) export DNOSE=/nroot/opt/dnose-1.4/
     B<collector> --cfg /etc/adcmon.cfg \
     --dcfg /nroot/etc/dnose.conf \
     --unit 11 --verbose
```

OPTIONS

-b <BID> :

Same as option –board.

–board <BID> :

To read the temperature, voltage and current of a given board one has to specify the board number (BID) of the board to read the ADC register. The last digit of a board ID is the board number inside a unit. The range is from 0 to 3. The second digit from the right is the unit number inside a crate and it is in the range from 0 to 3. The first 1 to 3 digits from the left (3rd to 5th from the right) are the crate number and can be in the range from 0 to 127.

Example:

(1) The last board of for example crate 5 has for the BID: 533.

(2) The 2nd board of the last unit of crate 4 has the BID: 431.

-c <CID> :

Same as option –crate.

–cfg <FILE> :

To specify a configuration file. The configuration file is inevitable to provide the data for the database connection. This file should not be world readable. And it also should not be on a global readable network volume or similar. If the configuration file is not provided on the command line $NROOT/etc/adcmon.conf will be used. If $NROOT is not defined /nroot/etc/adcmon.conf will be used.

–crate <CID> :

To read the temperature, voltage and current of a given crate one has to specify the crate number (CID) of the crate to read the ADC register of every board in the crate. The crate number (CID) can consist of one up to tree digits. The range of the crate number is from 0 to 127.

-D :

Same as option –debug.

-d :

Same as option –dcfg.

–dcfg <FILE> :

To specify a configuration file. The configuration file is inevitable for dnose configuration to the process which collect the data via I2C and HIB. If the option –dcfg is not given on command line $NROOT/etc/dnose.conf will be used. If $NROOT is not defined /nroot/etc/dnose.conf will be used.

–debug :

The debug option is mostly for development purpose and can be used to get more infomations what the programm creates or collects.

–help :

Print a brief help message and exits.

-e :

Same as option –epoch.

–epoch :

If this option is present, this epoch time string will be used to store in the database. This is imortant for the mcollector, because then it is possible to ave one query time for several objects. For example if you query 3 units of the same crate, then this unit values are stored under the same epoch time.

–man :

Prints the manual page and exits.

-n :

Same as option –nothing.

–noquery :

This is a dry run for debugging purpose. Excecute the script except the temperature query.

-u <UID> :

Same as option –unit.

–unit <UID> :

To read the temperature, voltage and current of a given unit one has to specify the unit number (UID) of a given unit in a crate to read the ADC register of every board inside that unit. The last digit of a unit number is the relative unit inside a crate. The range is from 0 to 3. The other 1 up to 3 digits (2nd to 4th from the right) are the crate number. The range is from 0 to 127.

-v :

Same as option –verbose

–verbose> :

The verbose option prints out the ADC register in HEX value. If you would like to see more use the –debug option.

DESCRIPTION

This program will query for ADCREG infomation, like temperature, current, voltage insite a given range of modality. It can query one crate or one unit or one board. If it queries for one unit it will submit 4 subqueries on every board of that unit. If it will query a crate it will submit 16 subqueries on every board of that crate. If it queries a board it will of course make only one subquery for that board.

The programm will store the releveant part of the recieved ADCREG data inside a database. The database might be configured by a configuration file, witch can be supplied by a command line option. If no configuration file is supplied it will search for $HOME/.adcmon.conf. If that can not be found it will search for $NROOT/etc/adcmon.conf.

EXAMPLES

```
(1)
B<collector> --dcfg $ENV{NROOT}/etc/dnose.conf --board 833

(2)
DNOSE=/nroot/opt/dnose-1.4/ B<collector> \
 --cfg /etc/adcmon.cfg \
 --dcfg /nroot/etc/dnose.conf --crate 6

(3)
export DNOSE=/nroot/opt/dnose-1.4/
B<collector> --cfg /etc/adcmon.cfg \
--dcfg /nroot/etc/dnose.conf \
--unit 22 --verbose
```

FUNCTIONS

The following finctions are implemented in **collector** for internal use:

query_boards Query a given range of board numbers and write this to a database. This boards must be query-able by on HIB. This means normaly that this boards are a unit or a subset of a unit, like a single board.

```
$success = query_boards(\@board_numbers);
```

success = 1 if OK success = 0 on failure

init_apanext

```
$success = init_apenext($config,\@slice);
```

success = 1 if OK success = 0 on failure, or quits

start_communication Starts communication for apeNEXT

```
start_communication(\@slice);
```

read_adcreg Reads a given slice from a board. Normally this shoild be one node.

```
$return= &read_adcreg(\@slice);

$return (0,$datetime,undef)          if noquery option
$return (1,$datetime,$data_buffer)   if query   option
```

term_communication Quits a apeNEXT communiction.

```
term_communication();
```

quit

```
quit()

quit($exit_code);
```

catchint

```
catchint
```

debug

```
debug($message . "\n");
```

SEE ALSO

See the Online Documentation of ADCmon.

AUTHOR

Christian Kuelker <christian.kuelker@desy.de >

5 Database

5.1 ADCREG decoding

The 56 least significant bits of the ADCREG contains the information of 7 ADC:

0x0000 0000 0000 00ff	ADC1	Current
0x0000 0000 0000 ff00	ADC2	Voltage
0x0000 0000 00ff 0000	ADC3	Temperature at node N000 (front)
0x0000 0000 ff00 0000	ADC4	Temperature at node N001
0x0000 00ff 0000 0000	ADC5	Temperature at node N010
0x0000 ff00 0000 0000	ADC6	Temperature at node N011 (back)
0x00ff 0000 0000 0000	ADC7	0V if jumper J25 is set

The following encoding of the 8-bit ADC values applies:

Current : ADC1 contains $I/5.0$, where I is measured in Ampere (A).

Voltage : ADC2 contains $\frac{5600+180}{180}\,U$, where U is measured in Volt (V).

Temperature : ADC3 ... ADC6 contain $0.744 + 0.0119T$, where T is measured in degree Celsius.

5.2 Database records

The following information is stored in the database:

idx : Each record in the database has a unique index that is automatically assigned by the RDBMS.

epoch : Contains the time when the measurement has been performed. The time is defined as number of seconds since the Epoch.

board : Contains the position of the board following the conventions described in chapter 2.

value : Contains the content of the 56 least significant bits of the ADCREG.

mask : The mask defines which of the ADCs values are considered as not reliable.

The size of each record is of the order of 24 bytes. Assuming the ADCREG of each board of a crate being read every 5 minutes, 0.1 MBytes of data has to stored per crate and day, or equivalently: 300 MBytes per 8 crates and year.

```
+-------+------------+------+-----+---------+----------------+
| Field | Type       | Null | Key | Default | Extra          |
+-------+------------+------+-----+---------+----------------+
| idx   | int(20)    |      | PRI | NULL    | auto_increment |
| epoch | int(11)    |      |     | 0       |                |
| board | int(5)     |      |     | 0       |                |
| value | bigint(20) |      |     | 0       |                |
| mask  | bigint(20) | YES  |     | 0       |                |
+-------+------------+------+-----+---------+----------------+
```

6 Query Module

The Query Module is a library module which provides and interface to the database and allows to perform various data reduction operations. The functions provided by this library are specified in the following sections.

6.1 Record object type

The Query Module operates on records which are hashes with the following keys:

`beginTime`	time of first measurement
`endTime`	time of last measurement
`beginBoard`	index of first board (encoding defined in section 5.2)
`endBoard`	index of last board
`i`	current in Ampere
`u`	voltage in Volt
`t000`	temperature at node N000 in degree Celsius
`t001`	temperature at node N001 in degree Celsius
`t010`	temperature at node N010 in degree Celsius
`t011`	temperature at node N011 in degree Celsius

6.2 `select($rec, $beginTime, $endTime, $beginBoard, $endBoard)`

Function queries database to retrieve a list of records matching given search criteria.

6.2.1 Function arguments

rec : Pointer to an array of records as defined in section 6.1. If query is successful the array will contain the matching records. The function clears the array before appending any matching records.

beginTime : The field `epoch` of all matching records will be equal or larger than `beginTime`. The time is defined as number of seconds since the Epoch.

endTime : The field `epoch` of all matching records will be equal or smaller than `endTime`. The time is defined as number of seconds since the Epoch. A negative value will be interpreted as wild card.

beginBoard : The field `board` of all matching records will be equal or larger than `beginBoard`.

endBoard : The field `board` of all matching records will be equal or smaller than `endBoard`.

6.2.2 Return value

The function returns a non-zero value in case the query failed.

6.3 reduce<Op><Dim>($out, $in)

This set of functions will take an array of records on input and perform the reduction operation <Op> over all records where the parameters for a given dimension <Dim> are the same. For example, the average over all records with equal board range is calculated.

The following reduction operations <Op> are foreseen:

Average : Average over all records

Minimum : Minimum value from all records

Maximum : Maximum value from all records

Reductions can be performed for the following dimensions <Dim>:

Time : All records with equal time range (beginTime,endTime)

Board : All records with equal board range (beginBoard,endBoard)

6.3.1 Function arguments

out : Pointer to resulting array of records.

in : Pointer to input array of records.

6.3.2 Return value

The function returns a non-zero value in case the query failed.

7 Normaliser Module

The Normaliser Module is executed to generate configuration data which the Analyses Module uses to gauge the data which it periodically reads from the Query Module.

7.1 Options

`--board|unit|crate` *N* Generate norm data for all sensors in the given partition *N*.

`--begin` *date* Select all records created at given date or later.

`--end` *date* Select all records created until the given data.

7.2 Output

On output the Normaliser Module generates the following data structures:

`$normAverage[$bid]->[$adc]` : This array contains for each board `$bid` and ACD `$adc` (where `$adc` = `i`, `u`, `t000`, `t001`, `t010`, `t011`) the average value.

`$normMax[$bid]->[$adc]` : Similar as `@normAverage` but storing the maximum value.

`$normMin[$bid]->[$adc]` : Similar as `@normAverage` but storing the minimum value.

7.3 Manual Page

7.3.1 normalizer

Generates config for apeNEXT ADC register analyser

SYNOPSIS

normalizer [options] -f adcmon.conf -[c|u|b] ID -B <TIME> -E <TIME> -n norm.dat

OPTIONS

```
Mandatory Options:
 -f <FILE> | --cfg   <FILE>    adcmon configuration file
 -n <FILE> | --norm  <FILE>    normalized  data file
 -B <TIME> | --begin <TIME>    start time of normalized period
 -E <TIME> | --end   <TIME>    end time of nomralized period

Mandatory Options (choose only one option!):
 -c <CID> | --crate <CID>      crate number from 0 ... 127
 -u <UID> | --unit  <UID>      unit number from 0 ... 3
 -b <BID> | --board <BID>      board ID from 0 ... 3

Other Options:
  --merge                      Merge existing file, should
                               be used with --overwrite
  --overwrite                  Write over existing file
                               without asking
```

```
Auxiliary Options:
 -D | --debug                  Prints internal messages
 -h | --help                   Prints brief help message
    | --man                    Prints complete manual page
```

EXAMPLE

The execution of

```
B<normalizer> -f adcmon.conf.am8 -u 10 -B 01-15+00:50 \
           -E 01-15+01:00 -n norm.dat
```

will generate the file "norm.dat" for unit 10 under the path which is specified at the -n switch. In this case it is the current directory.

The content of norm.dat will be:

```
# norm.dat (This file was auto-generated by the ADCmon)
# Generator: (B<normalizer>) ./B<normalizer>
# Date: 2007-2-23T20:16:29
# Use the B<normalizer> again to regenerate it.
# Command: ./normalizer -f adcmon.conf.am8 -n norm.dat
# -B 01-15+00:50 -E 01-15+01:00 -u 10

%norm_conf = (
         average => {
                 '10' => {
                      '--norm' => 'norm.dat',
                      '--board' => '',
                      '--cfg' => 'adcmon.conf.am8',
                      '--end' => '01-15+01:00',
                      '--unit' => 10,
                      '--crate' => '',
                      '--begin' => '01-15+00:50'
                      },
                 },
         min      => {
                 '10' => {
                      '--norm' => 'norm.dat',
                      '--board' => '',
                      '--cfg' => 'adcmon.conf.am8',
                      '--end' => '01-15+01:00',
                      '--unit' => 10,
                      '--crate' => '',
                      '--begin' => '01-15+00:50'
                      },
                 },
         max => {
                 '10' => {
                      '--norm' => 'norm.dat',
                      '--board' => '',
                      '--cfg' => 'adcmon.conf.am8',
                      '--end' => '01-15+01:00',
                      '--unit' => 10,
                      '--crate' => '',
                      '--begin' => '01-15+00:50'
                      },
                 },
```

```
        );

%norm = (
     average => {
                  '10' => {
                        '103' => [
                              '48.53',
                              '1.50',
                              '28.20',
                              '40.44',
                              '42.21',
                              '34.71'
                            ],
                        '102' => [
                              '48.43',
                              '1.43',
                              '30.23',
                              '44.60',
                              '43.03',
                              '36.18'
                            ],
                        '101' => [
                              '48.64',
                              '1.65',
                              '32.19',
                              '45.32',
                              '42.55',
                              '39.61'
                            ],
                        '100' => [
                              '48.98',
                              '1.55',
                              '34.27',
                              '37.28',
                              '36.60',
                              '34.71'
                            ],
                     },
                },
     min     => {
                  '10' => {
                    '103' => [
                              '47.41',
                              '1.48',
                              '23.48',
                              '32.02',
                              '32.67',
                              '28.08'
                           ],
                    '102' => [
                              '47.16',
                              '1.41',
                              '24.80',
                              '34.64',
                              '33.33',
                              '29.39'
```

```
                                ],
                              '101' => [
                                        '47.41',
                                        '1.64',
                                        '26.11',
                                        '34.64',
                                        '32.67',
                                        '32.02'
                                ],
                              '100' => [
                                        '47.66',
                                        '1.52',
                                        '28.73',
                                        '29.39',
                                        '29.39',
                                        '28.73'
                                ]
                               },
                        },
          max     => {
                       '10' => {
                         '103' => [
                                '49.42',
                                '1.56',
                                '32.02',
                                '45.80',
                                '47.12',
                                '41.21'
                                ],
                              '102' => [
                                '48.92',
                                '1.48',
                                '33.99',
                                '49.74',
                                '47.77',
                                '41.21'
                                ],
                              '101' => [
                                '49.42',
                                '1.72',
                                '35.96',
                                '49.74',
                                '47.12',
                                '45.80'
                                ],
                              '100' => [
                                '49.92',
                                '1.60',
                                '37.93',
                                '43.18',
                                '41.86',
                                '41.21'
                                ],
                               },
                    },
}
```

COMMAND LINE DESCRIPTION

-B TIMESTRING :

Same as option –begin.

–begin TIMESTRING :

Read the temperature, voltage and current after a given time period. If no end is specified, then the last entry will be used. The format of TIMESTRING can be one of the following:

```
[YYYY]-MM-DD[+hh:[mm:[ss]]]
[YYYY]-MM-DD[Thh:[mm:[ss]]] (This is Europa norm en28601)

YYYY = 4 digits year string
MM   = 2 digits month string
DD   = 2 digits day string
+    = delimiter
T    = delimiter
hh   = 2 digits hour string
mm   = 2 digits minute string
ss   = 2 digits second string
```

If the year is not given, the current year is taken. If the other values are not given then 00 is taken.

-b <BID> :

Same as option –board.

–board <BID> :

To read the temperature, voltage and current of a given board one has to specify the board number (BID) of the board to read the ADC register. The last digit of a board ID is the board number inside a unit. The range is from 0 to 3. The second digit from the right is the unit number inside a crate and it is in the range from 0 to 3. The first 1 to 3 digits from the left (3rd to 5th from the right) are the crate number and can be in the range from 0 to 127.

Example:

(1) The last board of for example crate 5 has for the BID: 533.

(2) The 2nd board of the last unit of crate 4 has the BID: 431.

-c <CID> :

Same as option –crate.

–cfg <FILE> :

To specify a configuration file. The configuration file is inevitable to provide the data for the database connection. This file should not be world readable. If no option is given a file in $HOME/.adcmon.cfg is searched.

–crate <CID> :

To read the temperature, voltage and current of a given crate one has to specify the crate number (CID) of the crate to read the ADC register of every board in the crate. The crate number (CID) can consist of one up to tree digits. The range of the crate number is from 0 to 127.

-D :

Same as option –debug.

–debug :

The debug option is mostly for development purpose and can be used to get more information what the program creates or collects.

-E <TIMESTRING> :

Same as option –end.

–end <TIMESTRING> :

If this option is given with the appropriate TIMESTRING then the query will be made up till the specified time. This helps to reduce data. The format of TIMESTRING can be one of the following:

```
[YYYY]-MM-DD[+hh:[mm:[ss]]]
[YYYY]-MM-DD[Thh:[mm:[ss]]] (This is Europa norm en28601)

YYYY = 4 digits year string
MM   = 2 digits month string
DD   = 2 digits day string
+    = delimiter
T    = delimiter
hh   = 2 digits hour string
mm   = 2 digits minute string
ss   = 2 digits second string
```

If the year is not given, the current year is taken. If the other values are not given then 59 is taken.

-f :

Same as option –cfg.

–help :

Print a brief help message and exits.

–i-only :

This prints only the current of the ADC registers. Of course this can not be combined with –u-only or –t-only. But it can be combined with some summary function (–summary) or reduction function (–position-*, –time-*).

–man :

Prints the manual page and exits.

–merge :

This option joins an existing norm.dat with actual generated data. If the partition exist, it will be over written without warning. –overwrite is probably needed

Example:

Unit 10 and 11 was queried in norm.dat. And now a new **normalizer** run will query unit 12 with –merge option then the so generated norm.dat will include 10, 11 and 12. If another **normalizer** run will query unit 11, the norm.dat will stay with unit 10, 11 and 12 but have included updated unit 11 data.

-n <OUTFILE> :

Same as option –norm.

–norm <OUTFILE> :

The –norm option requires a file to be given. It writes the content of the hash %norm{average}, %norm{min} and %norm{max} to that file. This can be sourced in by the analyser. The OUTFILE must be writable by the **normalizer**.

–overwrite :

If the output File for the %narm and %narm_conf hash exist, it will be over written without asking.

-u <UID> :

Same as option –unit.

–unit <UID> :

To read the temperature, voltage and current of a given unit one has to specify the unit number (UID) of a given unit in a crate to read the ADC register of every board inside that unit. The last digit of a unit number is the relative unit inside a crate. The range is from 0 to 3. The other 1 up to 3 digits (2nd to 4th from the right) are the crate number. The range is from 0 to 127.

DESCRIPTION

The **normalizer** program can be executed to obtain configuration data concerning average, maximum and minimum of the ADC register values. This configuration data is used as a comparison by the analyser to judge on the new queried data from the database. On this basis the action handler might be triggered.

FUNCTIONS

Internal functions to **normalizer**

debug

Synopsis: :

```
debug($format,$value);
debug($format,"$value\n");
debug($format,[$value_1, $value_2, .., $value_n]);
```

Description: :

Prints a debug message. The format string is the same as printf.

SEE ALSO

See the Online Documentation of ADCmon.

AUTHOR

Christian Kuelker <christian.kuelker@desy.de >

8 Analysis Module

The Analysis Module performs on request of the Action Module the following operations:

1. Define start time for check processing as time stamp
2. Read norm data
3. Apply analysis rules and submit necessary requests to Query Module
4. Update warn level and return the warn level and a list of messages the Action Module
5. Store warn level and time stamp for given partition

The Analysis Module returns:

1. The current warn level.
2. A list of messages where each message contains (human readable) the following information:
   - Rule *i* failed for board *dddd*.
   - Rule *i* could not be applied for board *dddd*.

8.1 Analysis rules

8.1.1 Average values

For each board of a given partition and all ADCs on this board check that the average value over all records for that board collected during the last T_{av} seconds does not exceed the norm value multiplied by a factor ρ_{av}.

8.1.2 Maximum values

For each board in given partition and all ADCs on this board check that maximum value of all records for this particular ADC collected since the last check does not exceed the norm value multiplied by a factor ρ_{max}.

8.2 Warn level

The warn level for a particular partition is changed after the completion of one pass according to the following rules:

1. If any of the analysis rules fails the warn level for this pass is incremented by 1.
2. If none of the analysis rules fails and the warn level is larger than zero the warn level for this pass is decremented by 1.
3. If the application of any analysis rule fails the warn level for this pass is incremented by 1.
4. At startup the warn level is 0.

5. For any pass the warn level changes only by -1, 0, +1.

8.3 Options

`--board|unit|crate` N Analyse data for given partition N.

`--cfg` Configuration file.

8.4 Parameters

T_{av} When evaluating rule 8.1.1 a reduction is performed over all records in time range $[t - T_{\mathrm{av}}, t]$, where t refers to the time the processing of a check request started.

ρ_{av} The norm value for the ADC times this parameter controls the limit when applying rule 8.1.1.

ρ_{max} The norm value for the ADC times this parameter controls the limit when applying rule 8.1.2.

9 Action Module

The Action Module regularly uses the Analysis Module to check the current action level for a particular partition. For each action level `$l` a particular action `$action{$l}` may be defined. If no action is defined for the corresponding action level, the action for the next lower action level is executed for which `$action{$ll}` is defined (`$ll` $<$ `$l`). `$action{0}` is set to `nop` unless the parameter file defines it differently.

9.1 Actions

The following actions can be defined:

`nop` Do nothing.

`warn` Send an email including the messages returned by the Analysis Module.

`kill` Kill jobs running on the given partition and sends a mail.

9.2 Options

`--board|unit|crate` *N* Analyse data for given partition *N*. This option is not mandatory. If it is used the list of partitions defined in the configuration file is overwritten.

`--cfg` Configuration file.

9.3 Parameters

`@partition_list` List of partitions for which the warn level is checked.

`$action[$l]` Action to be performed at given `$l`.

9.4 Configuration

The configuration file `action.conf` contains valid Perl syntax. It defines the hash %action.

```
SYNTAX:

%action = (
            level => 'ID:Subject:Mail Address',
          );

WHEREAS:

level        = (0..n)

ID           = qw(nop warn kill shutdown halt)
               nop      = no operation
               warn     = send a warning, nothing else
               kill     = kill the jobs in the queue
               shutdown = switch off blade server
               halt     = switch off apeNEXT
```

```
Subject      = Some String

Mail Address = RFC 822 comliant internet mail address
               (example: abc@desy.de)
```

9.5 Manual Page

9.5.1 action

Takes **action** upon analysis of apeNEXT ADC register data

SYNOPSIS

action [options] -f adcmon.conf

OPTION OVERVIEW

```
Mandatory Options:
 -f <FILE> | --cfg      <FILE>    adcmon configuration file
```

EXAMPLE

The execution of

```
B<action>
```

OPTIONS

-f adcmon.conf :

–test : run in testmode (no mail) :

-D level : prints debug messages :

debug

```
&debug($message . "\n");

&debug($format. "\n",$value);
```

10 Viewer

The Viewer reads for a given partition and a optionally defined time range all matching records from the database and provides different representations of the data.

10.1 Options

Note that additional configuration options are defined in the configuration file (see 11).

10.1.1 General options

`--cfg` *file* Use *file* as configuration instead of default configuration file.

10.1.2 Record selection options

The options `--board`, `--crate`, `--unit` are mutually exclusive.

`--begin` *date* Select all records created generate at given date or later.

`--end` *date* Select all records created until the given data.

`--board` *N* Select all records for board *N*.

`--crate` *N* Select all records for all boards of crate *N*.

`--unit` *N* Select all records for all boards of unit *N*.

10.1.3 Reduction operation options

The options `--position*` are mutually exclusive. The same applies to `--time*` and `--tch*`.[1]

`--position-av` Perform data reduction by calculating average values for all records where the time stamp is the same but the board position possibly different.

`--position-max` Perform data reduction by selecting maximum values from all records where the time stamp is the same but the board position possibly different.

`--position-min` Perform data reduction by selecting minimum values from all records where the time stamp is the same but the board position possibly different.

`--time-av` Perform data reduction by calculating average values for all records where the board position is the same but the time stamp possibly different.

`--time-max` Perform data reduction by selecting maximum values from all records where the board position is the same but the time stamp possibly different.

`--time-min` Perform data reduction by selecting minimum values from all records

[1] However, one position, one time and one channel reduction operation may be combined, for instance: `--position-max --time-max --tch-max`.

where the board position is the same but the time stamp possibly different.

`--tch=N` Perform data reduction by selecting results from the temperature sensor $N = 0..3$. The temperature sensors numbers are ordered from front to rear, i.e. N=0 (N=3) corresponds which is closest to the front-panel (backplane).

`--tch-av` Perform data reduction by calculating average values for all four temperature sensors.

`--tch-max` Perform data reduction by selecting the maximum value from all four temperature sensors.

`--tch-min` Perform data reduction by selecting the minimum value from all four temperature sensors.

10.1.4 Select options

The following options are mutually exclusive:

`--t-only` Show results for temperatures only.

`--u-only` Show results for voltages only.

`--i-only` Show results for currents only.

10.2 Manual Page

10.2.1 viewer

Views apeNEXT ADC register data

SYNOPSIS

viewer [options] –cfg /path/to/adcmon.conf [-c|-u|-b] [–position-*|–time-*]

OPTIONS OVERVIEW

Mandatory Options:

```
-f <FILE> | --cfg <FILE>      adcmon configuration file
```

Mandatory Options (choose only one option!):

```
-c <CID> | --crate <CID>       crate number from 0 ... 127
-u <UID> | --unit  <UID>       unit number from 0 ... 3
-b <BID> | --board <BID>       board ID from 0 ... 3
```

Reduction options: (–position-* or –time-* should be given)

```
--position-av      reduction of time, prints average
--position-max     reduction of time, prints maximum
--position-min     reduction of time, prints minimum
--time-av          reduction of position, prints average
--time-max         reduction of position, prints maximum
--time-min         reduction of position, prints minimum
--tch N            red. of temp. channel, prints only one
--tch-av           print temp. average over 4 channels
--tch-man          print temp. maximum over 4 channels
```

```
--tch-min          print temp. minimum over 4 channels
--i-only           prints only current
--t-only           prints only temperature
--u-only           prints only voltage
```

Summary options

```
-s        | --summary      print some statistic summary
--tch-av                   s.o.
--tch-max                  s.o.
--tch-min                  s.o.
```

Other Options:

```
-D        | --debug        prints debugging information
--help                     brief help message
--man                      full documentation\
-r        | --raw          print stored data without
                           processing (for debug)
-v        | --verbose      prints ADC register in
                           hexadecimal format
-w        | --warn         prints warnings about corrupt
                           data (if available)
```

EXAMPLE

Example (1):

```
viewer --cfg adcmon.conf --board 533
(no output printed)
```

Example (2):

```
viewer --cfg adcmon.conf --board 100 --time -av

#:   position:    Vav:   Aav:  Cav:   Cav:   Cav:  Cav:
 ----------------------------------------
00   board 100    48.98  1.55  34.27  37.28  36.60  34.71
```

OPTIONS

-B TIMESTRING :

Same as option –begin.

–begin TIMESTRING :

Read the temperature, voltage and current after a given time period. If no end is specified, then the last entry will be used. The format of TIMESTRING can be one of the following:

```
[YYYY]-MM-DD[+hh:[mm:[ss]]]
[YYYY]-MM-DD[Thh:[mm:[ss]]] (This is Europa norm en28601)

YYYY = 4 digits year string
MM   = 2 digits month string
DD   = 2 digits day string
+    = delimiter
T    = delimiter
hh   = 2 digits hour string
```

```
mm   = 2 digits minute string
ss   = 2 digits second string
```

If the year is not given, the current year is taken. If the other values are not given then 00 is taken.

-b <BID> :

Same as option –board.

–board <BID> :

To read the temperature, voltage and current of a given board one has to specify the board number (BID) of the board to read the ADC register. The last digit of a board ID is the board number inside a unit. The range is from 0 to 3. The second digit from the right is the unit number inside a crate and it is in the range from 0 to 3. The first 1 to 3 digits from the left (3rd to 5th from the right) are the crate number and can be in the range from 0 to 127.

Example:

(1) The last board of for example crate 5 has for the BID: 533.

(2) The 2nd board of the last unit of crate 4 has the BID: 431.

-c <CID> :

Same as option –crate.

–cfg <FILE> :

To specify a configuration file. The configuration file is inevitable to provide the data for the database connection. This file should not be world readable. If no option is given a file in $HOME/.adcmon.cfg is searched.

–crate <CID> :

To read the temperature, voltage and current of a given crate one has to specify the crate number (CID) of the crate to read the ADC register of every board in the crate. The crate number (CID) can consist of one up to tree digits. The range of the crate number is from 0 to 127.

-D :

Same as option –debug.

–debug :

The debug option is mostly for development purpose and can be used to get more information what the program creates or collects.

-E <TIMESTRING> :

Same as option –end.

–end <TIMESTRING> :

If this option is given with the appropriate TIMESTRING then the query will be made up till the specified time. This helps to reduce data. The format of TIMESTRING can be one of the following:

```
[YYYY]-MM-DD[+hh:[mm:[ss]]]
[YYYY]-MM-DD[Thh:[mm:[ss]]] (This is Europa norm en28601)
```

```
YYYY = 4 digits year string
MM   = 2 digits month string
DD   = 2 digits day string
+    = delimiter
T    = delimiter
hh   = 2 digits hour string
mm   = 2 digits minute string
ss   = 2 digits second string
```

If the year is not given, the current year is taken. If the other values are not given then 59 is taken.

-f :

Same as option –cfg.

–help :

Print a brief help message and exits.

–i-only :

This prints only the current of the ADC registers. Of course this can not be combined with –u-only or –t-only. But it can be combined with some summary function (–summary) or reduction function (–position-*, –time-*).

–man :

Prints the manual page and exits.

–position-av :

This reduction-function give only the average of the ADC values for a fixed time where the position differ. If you want only one table as output you should not use any –time-* reduction function, but you can select more as one –position-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --position-av
```

```
#:   time:        Vav:   Aav:  Cav:   Cav:   Cav:   Cav:
 ---------------------------------------
00   1168822201   49.11  1.55  30.05  39.57  38.42  34.64
01   1168822501   48.61  1.53  30.21  39.73  38.75  35.30
```

–position-max :

This reduction-function give only the maximum of the ADC values for a fixed time where the position differ. If you want only one table as output you should not use any –time-* reduction function, but you can select more as one –position-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --position-max
```

```
#:   time:        Vmax:  Amax: Cmax:  Cmax:  Cmax:  Cmax:
 -------------------------------------------------
00   1168822201   49.42  1.68  32.67  42.52  39.89  37.93
01   1168822501   48.92  1.64  33.33  42.52  40.55  37.93
```

–position-min :

This reduction-function give only the minimum of the ADC values for a fixed time where the position differ. If you want only one table as output you should not use any –time-* reduction function, but you can select more as one –position-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --position-min

#:   time:        Vmin:  Amin: Cmin:  Cmin:  Cmin:  Cmin:
 ------------------------------------------------
00   1168822201   48.92  1.45  27.42  35.30  34.64  33.33
01   1168822501   48.42  1.41  27.42  35.96  34.64  33.99
```

-r :

Same as option –raw.

–raw :

The raw option can be used to make an output pars-able for other programs. It will not print header or lines. The values will be separated by a single TAB character.

-s :

Same as option –summary.

–summary :

The summary option adds a summary line to the output. The kind of summary depends on other option. If you choose for example the maximum for reducing option –position-max you will get the maximum of all values for a register. Basically the summary is a column summary. The –tch-* option, which is by it self a summary, will force to be –summary switch present. This is due to the fact that the result of –tch-* option can otherwise not assigned.

Example:

```
 This will force --summary no be present, you have not
 to specify it:

./viewer --cfg $CFG -u 10 -B 01-15+00:50 \
         -E 01-15+01:00 --position-min --tch-min

#:  time:       Vmin: Amin:Cmin: Cmin: Cmin: Cmin: 0-3Cmin:
 ------------------------------------------------------------
00  1168822201  48.92 1.45 27.42 35.30 34.64 33.33
01  1168822501  48.42 1.41 27.42 35.96 34.64 33.99
 ============================================================
Summary:        48.42 1.41 27.42 35.30 34.64 33.33 27.42

 If you are not using --tch-* you must specify --summary

#:  time:   Vmin: Amin: Cmin: Cmin: Cmin: Cmin:
 ------------------------------------------------
00  1168822201  48.92 1.45 27.42 35.30 34.64 33.33
01  1168822501  48.42 1.41 27.42 35.96 34.64 33.99
 ================================================
Summary:        48.42 1.41 27.42 35.30 34.64 33.33
```

–t-only :

This prints only the temperature of the ADC registers. Of course this can not be combined with –u-only or –i-only. But it can be combined with some summary function (–summary) or reduction function (–position-*, –time-*). It can also be combined with the –tch=N or one of –tch-min, –tch-max, –tch-av.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50 \
         -E 01-15+01:00 --position-min \
         --t-only --tch=0

#:      time:           Cmin:
-----------------------------------------------------
00      1168822201      27.42
01      1168822501      27.42
```

–tch N :

This option specify a the channel (ch) of temperature (t) on which a reduction, should be made. Only this channel will be printed. Of course you can not specify a –tch-av, –tch-max or –tch-min option, because this must have all channels to be present to calculate the average, maximum or minimum. But it can be used together with the –summary function.

The channel number N can be in the range from 0 to 3.

Example:

```
Calculate the average, minimum and maximum of
temperature channel 0
./viewer -f $CFG -u 10 -B 01-15+00:50 \
-E 01-15+01:00 --position-av --position-min \
--position-max --t-only --tch 0

#:      time:           Cav:    Cmin:   Cmax:
---------------------------------
00      1168822201      30.05   27.42   32.67
01      1168822501      30.21   27.42   33.33
```

–tch-av :

If this option is present, the average of all temperature registers will be calculated and printed. This option must preceded by –time-av or –position-av. The option –tch-av, –tch-min and –tch-max are mutually exclusive.

–tch-max :

If this option is present, the maximum of all temperature registers will be calculated and printed. This option must preceded by –time-max or –position-max. The option –tch-av, –tch-min and –tch-max are mutually exclusive.

–tch-min :

If this option is present, the minimum of all temperature registers will be calculated and printed. This option must preceded by –time-min or –position-min. The option –tch-av, –tch-min and –tch-max are mutually exclusive.

–time-av :

This reduction-function give only the average of the ADC values for a fixed position where the time differ. If you want only one table as output you should not use any –position-* reduction function, but you can select more as one –time-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --time-av
```

```
#:   position:    Vav:   Aav:  Cav:  Cav:  Cav:  Cav:
 ------------------------------------------------------
00   board 100    49.17  1.56  33.00  35.63  34.64  33.66
01   board 101    48.92  1.66  30.70  42.52  39.57  37.93
02   board 102    48.67  1.43  29.39  41.86  40.22  34.64
03   board 103    48.67  1.52  27.42  38.58  39.89  33.66
```

–time-max :

This reduction-function give only the maximum of the ADC values for a fixed position where the time differ. If you want only one table as output you should not use any –position-* reduction function, but you can select more as one –time-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --time-max
```

```
#:   position:    Vmax:  Amax: Cmax:  Cmax:  Cmax:  Cmax:
 ------------------------------------------------------
00   board 100    49.42  1.56  33.33  35.96  34.64  33.99
01   board 101    49.17  1.68  30.70  42.52  39.89  37.93
02   board 102    48.92  1.45  29.39  41.86  40.55  35.30
03   board 103    48.92  1.52  27.42  38.58  39.89  33.99
```

–time-min :

This reduction-function give only the minimum of the ADC values for a fixed position where the time differ. If you want only one table as output you should not use any –position-* reduction function, but you can select more as one –time-* option.

Example:

```
./viewer --cfg $CFG -u 10 -B 01-15+00:50:00 \
         -E 01-15+01:00:00 --time-min
```

```
#:   position:    Vmin:  Amin: Cmin:  Cmin:  Cmin:  Cmin:
 ------------------------------------------------------
00   board 100    48.92  1.56  32.67  35.30  34.64  33.33
01   board 101    48.67  1.64  30.70  42.52  39.24  37.93
02   board 102    48.42  1.41  29.39  41.86  39.89  33.99
03   board 103    48.42  1.52  27.42  38.58  39.89  33.33
```

-u <UID> :

Same as option –unit.

–unit <UID> :

To read the temperature, voltage and current of a given unit one has to specify the unit number (UID) of a given unit in a crate to read the ADC register of every board inside that unit. The last digit of a unit number is the relative unit inside a crate. The range is from 0 to 3. The other 1 up to 3 digits (2nd to 4th from the right) are the crate number. The range is from 0 to 127.

–u-only :

This prints only the voltage of the ADC registers. Of course this can not be combined with –i-only or –t-only. But it can be combined with some summary function (–summary) or reduction function (–position-*, –time-*).

-v :

Same as option –verbose

–verbose :

The verbose option prints out the ADC register value. If you would like to see more use the –debug option.

DESCRIPTION

This program will query a database for ADCREG information, like temperature, current, voltage inside a given range of modality. It can query one crate or one unit or one board.

The program will query the ADCREG data from a given database. Since now MySQL is supported. The database might be configured by a configuration file, witch can be supplied by a command line option. If no configuration file is supplied it will search for $HOME/.adcmon.conf. If that can not be found it will search for $NROOT/etc/adcmon.conf.

FUNCTIONS

Internal function to **viewer**.

print_view()

```
print_view(reduction, column, av, min, max, index_href);
```

This function mainly prints the header if no –raw option is present and call print_all_entries.

print_all_entries()

```
print_all_entries( type, type_href, width );
```

This function mainly prints the entry number and call print_entry. If the *–summary* switch is present, it calculates and prints a summary.

print_entry()

```
print_entry( type, type_href, object );
```

This function mainly calculates and prints one line of adcreg depending on the reduction.

SEE ALSO

See the Online Documentation of ADCmon.

AUTHOR

Christian Kuelker <christian.kuelker@desy.de >

11 Configuration files

A configuration allows to define additional run-time options. The default location of the configuration file is `$NROOT/etc/adcmon.conf` for adcmon.conf or adcmon.conf.template and a local file system for adcmon_db.conf. For Zeuthen this is /apeshare/aperun and for Bielefeld this is /root/etc for every apemaster. adcmon_db.conf should not be placed on an AFS volume.

The following syntax rules apply:

- Lines starting with a '#' (hash) are considered as comments.
- Lines containing white-space only are ignored.
- Lines defining a variable should match rules for Perl syntax.

11.1 adcmon.conf

To have a initial adcmon.conf the adcmon.conf.template can be copied.

11.1.1 Configuration parameters

$institution : Possible values: MOON, Roma, Bielefeld, Zeuthen.
Default value: 'MOON' (the test institution).
SYNTAX: $institution = 'MOON';

%mcollector : The %mcollector variable is defined as follows (example):

```
%mcollector = "host" => {
                DNOSE => [ "path/to/dnose/dir"],
                dcfg  => [qw(/nroot/etc/dnose.conf)],
                crate => [ 6, 7 ],
                         };
```

%action : This hash defines the action that will be performed acording the given action level.

```
SYNTAX:

%action = (
            level => [
                       'ID',
                       'Subject',
                       'Address',
                       'Script|noexec'
                     ],
          );

WHEREAS:

level   = (0..n)

ID      = nop|warn|kill|shutdown|halt
```

```
          nop      = do nothing
          warn     = send a warning, nothing else
          kill     = kill the jobs in the queue, send mail
          shutdown = switch off blade server, send mail (*)
          halt     = switch off apeNEXT, send mail      (*)

Subject = Some String (without ":")

Address = RFC 822 comliant internet mail address
               (example: abc@desy.de)
Script  = script code| script name | noexec
          If 'script name', exists tries top execute,
          else execute 'script code'.

(*) not implemented jet. Implementation not sheduled.
    instaed warn will be execued.

EXAMPLE:

   %action = (
       0 => [
              'nop',
              'MOON: AdcMon is working',
              'apeadm@desy.de',
              'noexec'
            ],
       1 => [
              'warn',
              'MOON: AdcMon first warning',
              'christian.kuelker@desy.de',
              'noexec',
            ],
       2 => [
              'warn',
              'MOON: AdcMon second warning',
              'dirk.pleiter@desy.de',
              'noexec',
            ],
       3 => [
              'warn',
              'MOON: AdcMon third warning',
              'apeadm@desy.de',
              'noexec',
            ],
       4 => [
              'kill',
              'MOON: AdcMon kill all running jobs',
              'apeadm@desy.de',
              'noexec',
            ],
       5 => [
              'shutdown',
              'MOON: AdcMon shutting down blade server',
              'apeadm@desy.de',
              'noexec',
```

```
            ],
        6 => [
               'halt',
               'MOON: AdcMon halting apNEXT',
               'capeadm@desy.de',
               'noexec',
            ],
    );
```

%partition : Defines a list of enabled partition entities for which the action script queries the analyses module and take action if necessary.

```
Example:
%partition = (
    board => [ 112 ],
    unit  => [ 20, 21, 22 ],
    crate => [ 1, 3, 4, 6, 7 ],
);
```

$normalizer_dat_file : If you do not want to define the location of norm.conf on the command line, you can define it here. But if you define it on the command line the command line has precedence.

Default value: ”

```
$normalizer_dat_file = '/nroot/etc/adcmon-zeuthen-norm.dat';
```

%action_modality : Defines a list of modalities for which the action should be analysed and eventually executed.

```
Possible values for modality are: crate, unit or board

The value to that modalities can be:
   0 no action
   1 action

Default value:
  %action_modality = (
      board => 1,
      unit  => 1,
      crate => 1,
  );
```

%analyzer : Values like threshold minimum, maximum and interval for the Analyzer.

```
interval      = comparison interval of seconds
threshold_av  = multiplication factor for norm
                in average modus
threshold_max = multiplication factor for norm
                in maximal modus

   %analyzer = (
       interval      => 600,
       threshold_av  => 1.2,
       threshold_max => 1.1,
               );
```

$adcmon_status_file : Status file for Action Module to save the current action level.

```
$adcmon_status_file='/apeshare/aperun/adcmon_status.dat';
```

$adcmon_db* : DB credentials should not be saved in this file or under a AFS volume. You can specify here a location for the DB credentials. This directory should be accessible by the user of crond.

```
$adcmon_db_read_access_conf
    = "/apeshare/aperun/adcmon_db_test.conf";
$adcmon_db_write_access_conf
    = "/apeshare/aperun/adcmon_db_test.conf";
$adcmon_db_create_access_conf
    = "/apeshare/aperun/adcmon_db_test.conf";
```

11.2 adcmon_db.conf

This file stores only the database credentials in the variable adcmon_db_access.

11.2.1 Configuration parameters

adcmon_db_access :

```
%adcmon_db_access = (
                DB_USER          => 'apeuser',
                DB_DRIVER        => 'mysql',
                DB_PASSWORD      => 'secret',
                DB_HOST          => 'localhost',
                DB_PORT          => '3306',
                DB               => 'apenext_adcmon',
                DB_TABLE         => 'ADCmon',
                        );
```