The DRS DAQ Program for CTX

S. Commichau
<sebastian.commichau@phys.ethz.ch>

Institute for Particle Physics
ETH Zurich

November 2008

Contents

1 Introduction

This note briefly describes the installation and the main features of the DRS DAQ program [1].

2 The Domino Ring Sampler Readout

The Domino Ring Sampler (DRS) is an analog sampling chip fabricated in a 0.25 μm CMOS process [2, 3]. The chip provides 8(10) input channels with an input voltage limit of 2 V (0.1 V - 1.1 V linear input range), each of which is followed by an array of 1024 capacitive sampling cells (arranged in a ring-buffer). The capacitive sampling cells form a so-called switched capacitor array (SCA) and belong to the analog part of the chip. The digital part of the chip is responsible for digital control and multiplexing. The sampling frequency is generated on the chip itself (with a series of inverters) and ranges from 0.5 GHz to 3 GHz. Once an analog signal is sampled and a trigger occurs, the contents of the sampling cells are frozen and the SCA is read out at 40 MHz with an external 12 bit flash analog-to-digital (FADC) converter. The readout speed is limited by the fact that all 1024 cells have to be transferred from the SCA. For the next DRS generation (version 3/4) it is foreseen to reduce the readout time by selecting a region of interest before readout.
The DAQ system at the electronics workshop (Institute for Particle Physics, ETH Zurich) is based on a 32-channel VME board. The board houses two 16(20)-channel mezzanine cards, each of which contains two DRS chips (version 2) and supporting electronics. The VME board contains FPGAs for generating the readout sequence and the storage of the digitized data. Further information can be found elsewhere [2]. Results on performance measurements of the DRS chip (version 2) can be found in [4].

3 Requirements

With the DAQ program it is possible to modify the DRS registers, to transfer the data from the 32-channel VME board and to store it on a hard disk.

- The DAQ program requires DRS version 2 or 3 and version 3 of the DRS VME board [2, 3]. Furthermore, the program works with the Concurrent Technologies (CT) VME controller VP 315/022-RC [5] and the Struck VME controller SIS 3100. Both controllers have a VME-64 interface supporting
  A32/A24/A16/D64/D32/D16/D8(E0), MBLT. The SIS 3100 controller supports the 2eVME protocol, which is much faster than the other transfer protocols.

- The program has to be compiled for each controller. Depending on the controller different objects/libraries are required and linked to the executable. To select one of the above controllers the flag `VMECTRL` (in `Makefile.general`) must be set either to `-DCT_VME` or `-DSTRUCK_VME`.
  Note: in case of the CT VME controller the driver and library (`vme_rcc`) so far only supports the Tundra Universe PCI $\leftrightarrow$ VMEbus ASIC. As this chip does not support 2eVME & 2eSST there is no 2eXXX support in the library (`vme_rcc`) either [7].

- All shared libraries and drivers which are needed by the CT and the Struck controller are included in this version of the DAQ program (the directory `./VME/` contains everything).

- The Linux kernel version required for the CT VME driver is `2.6.9-67.0.7`.

- ...

4 Installation

The following steps are necessary to install the DRS DAQ program:

1. First of all make sure that the VME driver is properly installed. The CT driver was already installed, so I cannot comment on this, but in case of the Struck controller do the following (requires super user privileges!):
   - Go to the directory `$HOME/drsdaq/VME/struck/sis1100/V2.02/dev/pci/`
   - Enter `sudo ./load_module`
   - To check if the driver has been installed type `dmesg | grep sis1100`
   - Go to the directory `/tmp/` and enter `ln -s sis1100_00remote sis1100`
2. Extract the gzipped tar ball: `tar xvfz drsdaq_VN_DDMMYYYY.tgz`.
3. Go to the main directory: `cd drsdaq`.
4. Select the current VME controller by chosing the correct flag in `Makefile.general`: the flag `VMECTRL` must be set to `-DCT_VME` in case of the CT VME controller or to `-DSTRUCK_VME` in case of the Struck controller.
5. Type `make` in the main directory.
6. Edit the default configuration file `DRSDAQ.conf`: the variables `LogPath` and `RawDataPath` must be set properly.
7. Create the directory corresponding to the `RawDataPath`, as defined in `DRSDAQ.conf`.

5 Running the Program

- To run the program, type `./drsdaq`. If the VME driver is appropriate and a DRS VME board has been found you will get an output as shown in appendix B.
- The program automatically scans the entire VME crate for DRS boards, assuming geographic addressing. Geographic addressing allows the controller to identify the slot in which the DRS board is located.
- Loading a custom configuration file: a custom configuration file can be passed as command line argument to the DAQ program:

  ```
  ./drsdaq -c <ConfigFile>
  ```

  The configuration file is updated at program-exit time (see next section).
- Upon every startup a log file will be generated. It will be stored in the directory specified by `RawDataPath` (see appendix D).

5.1 Available Commands

```
<ARG> = mandatory argument
[ARG] = optional argument
```

- **ba(sh)**: start bash. Type exit to return to the DAQ program.
- **b(oard)**: Select CMC boards from VME crate:

```
board <i>, <i> [j], <all>
```

select CMC board `i`, boards `i-j`, `all` boards.

- **ca(libration)**: Before data acquisition an internal calibration is required: to do this, enter

  ```
  ca(lib) [target dir] <trigger frequency> <calibration frequency> [GHz].
  ```

  If no target directory (first argument) is provided the program will use the default directory (recommended). The program generates calibration files for each CMC Board. These files can only be used at the DRS sampling frequency which has been set for their generation.

- **(cl)ear**: clear screen.

- **cc(lient)**: The command

  ```
  cc(lient) <name>
  ```

  sets the machine name of the CC client to `name`. The DAQ configuration file is updated accordingly at program-exit time.

- **c(onfig)**: Print DAQ configuration (read from configuration file (`DRSDAQ.conf`)).

- **de(layed)**: Modify delayed start:

  ```
  de(layed) <0|1>,
  ```

  `0` disables and `1` enables delayed start.

- **d(isk)**: Print disk space in mega bytes.

- **f(requency)**: To set the DRS sampling frequency enter

  ```
  f(requency) <GHz> [1]
  ```

  The first argument denotes the sampling frequency in GHz, the second one is optional and specifies if frequency regulation should be used or not.

- **h(elp)**: To print all available commands (see appendix C) type

  ```
  h(elp)
  ```

- **inf(o)**: Show status of DAQ and connected DRS boards.

- **le(d)**: Turn LED on/off:

  ```
  le(d) <on|off>.
  ```

- **m(ode)**: Set DRS mode

  ```
  m(ode) <0|1>,
  ```

  to single shot (`0`) or continuous (`1`).

- **(p)ort**: The command

```
(p)ort <port>
```

changes the port to be used by the CC to `port`. The most recent value is written to the DAQ configuration file (`DRSDAQ.conf`).

- **q(uit)/e(xit)**: Exit the program.
- **r(amtest)**: RAM integrity and speed test.
- **rea(d)**: Read data from DRS board:

  ```
  r(ead) <i> <ch> [0|1],
  ```

  reads data from board `i`, channel `ch` with (`1`) or without calibration (`0`). The data will be printed to the standard output. If data is read without calibration the unit is ADC counts otherwise millivolts.

- **rm(ode)**: Set DRS readout mode:

  ```
  rm(ode) <0|1>.
  ```

  To start the readout from stop position set it to `0` and to start the readout from first bin choose `1`.

- **ro(ot)**: Start ROOT (if installed).
- **sc(an)**: Search for DRS boards in the VME crate.
- **se(rial)**: Set serial number of DRS boards (experts only):

  ```
  se(rial) <i> <n>,
  ```

  changes serial number of board `i` to `n`.

- **st(art)**: Start DRS wave.
- **sto(p)**: Stop DRS wave (issue software trigger).
- **t(ake)**: To acquire a pedestal, calibration (not yet implemented) or data run at the current sampling frequency, type

  ```
  t(ake) [P|C|D] [EVENTS] [RUNNUMBER] [SOURCE]
  ```

  If no argument is provided a data run is started. The first argument specifies the run type. The second argument (`EVENTS`) specifies the number of events to be taken, whereas the third denotes the run number. Both the number of events and the run number are read from the DAQ configuration file and automatically incremented for the next run. On program exit the current values are written to the configuration file.
  The raw files are stored in the directory

  ```
  RawDataPath/YYYYMMDD/
  ```

  The naming convention of the raw files follows to some extend the one used in the MAGIC experiment:

  ```
  YYYYMMDD_RUNNUMBER_SOURCE_TYPE.raw
  ```

The format of the raw data can be inferred from `RawDataCTX.cc/h` (see appendix F). The program `inspectrawfile` can be used to read the raw data. It may also serve as a template to develop a program for further analysis of raw data.
In case of a data and calibration run the program waits for an external (hardware) trigger. The external trigger must be provided as a TTL signal (input signal: $< 0.8\,\mathrm{V}$ = low level, $> 2.0\,\mathrm{V}$ = high level) to the USER connector front panel: pin 2 (left), GND; pin 1 (right), trigger input. The signal should be terminated with $50\,\Omega$ at the VPC input and it should last about 100 ns [3]. In case of a pedestal run the program issues software triggers to continuously stop and re-start the DRS.

- **te(st(2e)blt32|64)**: Do benchmark test, i.e. perform `N` `D32` or `D64` DMA read operations. Node: the 2eVME mode only works with the Struck VME controller.

  ```
  te(st(2e)blt32|64) [N]
  ```

  Some statistics are shown after completion.

- **ti(me)/da(te)**: Print current time and date.

- **tr(ig)**: Change between hardware and software trigger:

  ```
  tr(ig) <0|1>,
  ```

  `0` disables and `1` enables the (external) hardware trigger.

- **u(ptime)**: Print DAQ uptime (`h:m:s`).

- **w(mode)**: Set DRS wave mode:

  ```
  w(mode) <0|1>.
  ```

  To keep the DRS wave running during readout set it to `1` and to stop it during readout choose `1`.

- ...

All commands listed beforehand can also be sent through a TCP/IP connection to the DAQ program (see next section). The machine name of the client and the port to be used for the connection are defined in `DRSDAQ.conf`. The DAQ program continuously tries to connect to the CC client using the port which was either set by hand or read from `DRSDAQ.conf`.

5.2 Utility Programs

Two simple utility programs which are able to write a dummy file and to read raw data can be found in the subdirectory `utilities`. The program `writerawfile` writes a certain number of events (max. 500 events) to a file according to the current raw data format. Usage:

```
./writerawfile <outfile> [events]
```

To read the data - either raw data from the DAQ program or from the program `writerawfile` - one can use the program `inspectrawfile <rawfile>`. Usage:

```
./writerawfile <rawfile>
```

Another program, called `remotecontrol`, represents a simple command line interface to issue all commands listed in the previous section to the DAQ program via TCP/IP. Usage:

```
./remotecontrol <server name> <port>
```

For the server name and port configuration see e.g. `DRSDAQ.conf`.
Note: all executables mentioned beforehand are not compiled when the main DAQ program is compiled and linked, i.e. type `make` in the directory `utilities`.

6 Using ROOT to read Raw Data

The following steps are necessary to use ROOT for reading binary raw data produced by the DAQ program:

1. Make sure that ROOT [6] is installed. ROOT version 5.18/00 is not necessary but was used for all tests until now.

2. Assure that the DRS DAQ program is properly installed: a shared object called `RawDataCTX.so` (in `$HOME/drsdaq/DAQ/`) is generated, which is mandatory to read binary raw files with ROOT.

3. If the location of the library has changed go to the directory called `root`. Edit `rootlogon.C` such that ROOT can find the shared object, i.e. change the lines

   ```
   const TString path = "/cpp/drsdaq2/DAQ/";
   const TString lib  = "RawDataCTX.so";
   ```

   accrodingly.

4. A template macro called `readraw_template.C` is provided. The location of the header files

   ```
   #include "/home/scommichau/drsdaq/DAQ/RawDataCTX.h"
   #include "/home/scommichau/drsdaq/DAQ/CTXTypes.h"
   ```

   has to be changed accordingly to be able to run the macro. You also have to change the macro such that the raw data will be found. Once this is done the macro should work. Note: the macro must (!) be compiled, i.e. start ROOT and type

   ```
   .x readraw_template.C++
   ```

5. You may want to use ROOT to read raw data from everywhere: edit the file `copy_to_home_as.rootrc` (if needed) and copy it to `$HOME/.rootrc`

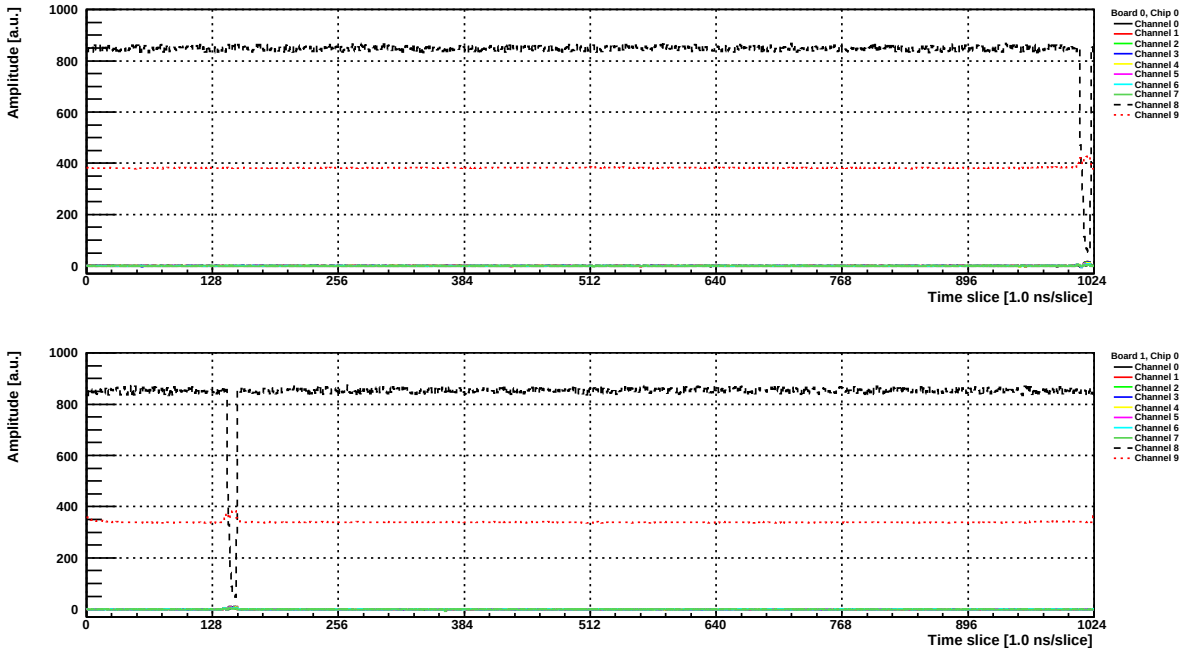


Figure 1: Digitised waveforms from the DRS 2 chip. The trigger signal in channel #8 is used as a reference to rotate the waveform.

7 What remains to be done...

⋆ Transferrate achieved with the CT VME controller: only $\sim$ 13.5 MBytes/s with D64 (MBLT) read and $\sim$ 7 MBytes/s with D32 read are achieved, which results in a trigger rate well below 100 Hz - if all four mezzanine cards are used.

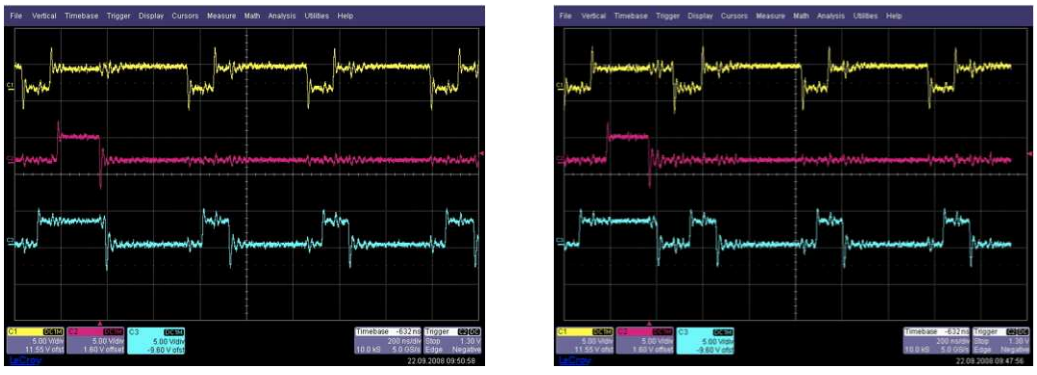
- Result from D32 read:

```
************************************************
 Mode: VMEbus A32/D32 DMA read [64 kB]
 Contiguous buffer:
  VMEbus address:   0X02A40000
  Physical address: 0X064B0000
  Virtual address:  0XB5E3F000
 1000 BLT(s) finished...
 0 errors... success!
 Duration: 8.725 s
 Rate: 7.163 MB/s
************************************************
```

- Result from D64 read:

```
************************************************
 Mode: VMEbus A32/D64 DMA read [64 kB]
 Contiguous buffer:
  VMEbus address:   0X02A40000
  Physical address: 0X18900000
  Virtual address:  0XB665F000
 1000 BLT(s) finished...
 0 errors... success!
 Duration: 4.611 s
 Rate: 13.556 MB/s
************************************************
```

Some of the VMEbus signals that are used during (DMA) read/write cycles are shown in figure 2. Yellow: `DTACK*` (data transfer acknowledge); pink: `AS*` (address strobe); turqoise: `DS*` (data strobe)). The star (*) indicates that the signal is active if low. The slave (DRS VME board) uses the `DTACK` signal for two purposes: directly after `AS` being inserted to acknowledge the address; after each assertion of `DS` to acknowledge data. It can be seen that the response of the slave is in the order of 400 ns, which limits the maximum data transfer rate to $\sim$ 13.5 MBytes/s, i.e. contrary to 2eXXX transfers D32/64 DMA transfers are handled inefficiently by the DRS VME board [3]. To speed up the transferrate we could use the Struck controller (requires changes in `DRS.cc/h`, see section F), which supports 2eXXX modes. The 2eXXX mode allows for a transferrate $\mathcal{O}$(250 MBytes/s).



(a) D32 DMA read.

(b) D64 DMA (MBLT) read.

Figure 2: Some of the VMEbus signals that are used during (DMA) read/write cycles: example for a D32 DMA read (left) and for a D64 DMA read operation (right). Yellow: `DTACK*` (data transfer acknowledge); pink: `AS*` (address strobe); turqoise: `DS*` (data strobe)).

⋆ The maximum transferrate achieved with the Struck VME controller is $\sim 80\,\mathrm{MB/s}$ (with the 2eVME protocol):

```
************************************************
 Mode: VMEbus A32/D64 2eVME read [64 kB]
 1000 BLT(s) finished...
 0 errors... success!
 Duration: 0.797 s
 Rate: 78.443 MB/s
************************************************
```

8 Troubleshooting

- You might get the following error when trying to run the DAQ program:

  ```
  [eth-vme02] > error: 0x20c => major: Error  12 in package 2 => VMEbus
  driver/library for the RCC: Error from file operation (open/close) VME
  open not successful.
  ```

  This error maybe due to a mismatch between kernel and VME driver version: the current VME driver requires kernel version `2.6.9-67.0.7`. Use the command `uname -r` to find out about the current kernel version and compare this with the version of the kernel modules located in (`/home/$USER/drsdaq/atlas/driver/`). Load the correct kernel version at startup, and, if necessary, edit `/etc/grub.conf` to automatically use the required kernel version. Check the directory `/boot` if the right kernel version is available.
  In order to disable the YUM automatic update (to avoid the problem of being automatically updated to another kernel version) do the following:

  ```
  [eth-vme02] > service yum-autoupdate stop
  [eth-vme02] > chkconfig --del yum-autoupdate
  ```

  In order to enable the automatic update system, use the following commands:

  ```
  [eth-vme02] > chkconfig --add yum-autoupdate
  [eth-vme02] > service yum-autoupdate start
  ```

- ...

A A Template Configuration File

```
# Configuration file for the DRS DAQ V1, 2008 10 06, 13:53:11

LogPath             /home/scommichau/cpp/drsdaq/log/

RawDataPath         /home/scommichau/data/

NumEventsDatRun     1
NumEventsCalRun     1000
NumEventsPedRun     100

RunNumber           613

RotateWave          0

FirstSample         0
LastSample          512

MinDiskSpaceMB      1000
MaxFileSizeB        1000000000

CCPort              3000
CCClient            ihp-pc29.ethz.ch

FirstVMESlot        1
LastVMESlot         7
```

B Sample Output

```
********************** DRS readout built Oct  3 2008, 16:51:21 **********************

Opening file: DRSDAQ.conf
VME connection opened
CMEM opened
DAQ> found mezz. board  0 on VME slot  3 upper, serial #122, firmware revision 5268
DAQ> found mezz. board  1 on VME slot  3 lower, serial #123, firmware revision 5268
DAQ> found mezz. board  2 on VME slot  5 upper, serial #306, firmware revision 5268
DAQ> found mezz. board  3 on VME slot  5 lower, serial #213, firmware revision 5268
DAQ|B0-3> initialization of board 0 done
DAQ|B0-3> initialization of board 1 done
DAQ|B0-3> initialization of board 2 done
DAQ|B0-3> initialization of board 3 done
DAQ|B0-3> f 1
DAQ|B0-3> setting frequency without regulation:
CHIP #0, iter  0: 0.80000(20971) 0.84128 +1040
CHIP #0, iter  1: 0.83968(22011) 0.99362   +42
CHIP #0, iter  2: 0.84128(22053) 0.99972
CHIP #1, iter  0: 0.80000(20971) 0.91116  +582
CHIP #1, iter  1: 0.82221(21553) 1.00151    -9
CHIP #1, iter  2: 0.82183(21544) 1.00009
DAQ|B0-3> domino wave of board 0 is running at 1.000 GHz
CHIP #0, iter  0: 0.80000(20971) 0.92417  +497
CHIP #0, iter  1: 0.81896(21468) 1.00267   -17
CHIP #0, iter  2: 0.81829(21451) 0.99995
CHIP #1, iter  0: 0.80000(20971) 0.83174 +1103
CHIP #1, iter  1: 0.84207(22074) 0.99281   +47
CHIP #1, iter  2: 0.84386(22121) 0.99965
DAQ|B0-3> domino wave of board 1 is running at 1.000 GHz
CHIP #0, iter  0: 0.80000(20971) 0.90164  +645
CHIP #0, iter  1: 0.82459(21616) 1.00002
CHIP #1, iter  0: 0.80000(20971) 0.89356  +698
CHIP #1, iter  1: 0.82661(21669) 0.99720   +18
CHIP #1, iter  2: 0.82731(21687) 0.99976
DAQ|B0-3> domino wave of board 2 is running at 1.000 GHz
CHIP #0, iter  0: 0.80000(20971) 0.91385  +565
CHIP #0, iter  1: 0.82154(21536) 1.00297   -18
CHIP #0, iter  2: 0.82079(21516) 0.99965
CHIP #1, iter  0: 0.80000(20971) 0.91513  +556
CHIP #1, iter  1: 0.82122(21527) 1.00534   -34
CHIP #1, iter  2: 0.81988(21492) 0.99972
DAQ|B0-3> domino wave of board 3 is running at 1.000 GHz
DAQ|B0-3>
```

C The Help Menu

```
  <ARG> = mandatory argument
  [ARG] = optional argument

****************************************** HELP ******************************************

 board <i>, <i> [j], <all>             Address board i, boards i-j, all boards
 calib [DIR] <T_FREQ> [C_FREQ]         Response calibration (frequencies in GHz)
 cclient <name>                        Set machine name of the CC client to <name>
 clear                                 Clear screen
 config                                Print DAQ configuration
 del <0|1>                             Switch delayed start on <1>|off <0>
 disk                                  Show disk space in MB
 freq <GHz>                            Set DRS sampling frequency
 help                                  Print help
 info                                  Show DAQ status information
 led <ON|OFF>                          Turn LED on|off
 mode <0|1>                            Set DRS mode: 0 = single shot, 1 = continuous
 port <port>                           Set port to be used by the CC to <port>
 quit                                  Exit program
 ramtest                               RAM integrity and speed test
 read <i> <ch> [0|1]                   Read data from board <i>, <ch>=0...9 [W/O|W] cal.
 rmode <0|1>                           Set DRS readout mode
 root                                  Start ROOT
 serial <i> <n>                        Set serial # of board <i> to <n> (experts only)
 scan                                  Scan for boards
 start                                 Start domino wave
 stop                                  Issue soft trigger and stop DAQ
 take [P|C|D] [EVENTS] [RUN#] [SOURCE] Start DAQ, take Data, Ped. or Cal. run
 test[2e]blt[32|64] [N]                Test VMEbus [2edge] BLT D32|D64 (read)
 time                                  Prints current date and time
 trig <0|1>                            Hardware trigger on <1>|off (0)
 wmode <0|1>                           Set DRS wave mode
 uptime                                Get DAQ uptime [h:m:s]

******************************************************************************************
```

D A Sample Log File

```
[2008:07:08:12:46:03] start logfile
DRS DAQ configuration:
LogPath: /home/scommichau/cpp/drsdaq/log/
RawDataPath: /home/scommichau/data/
NumEventsDatRun: 1
NumEventsCalRun: 1000
NumEventsPedRun: 100
RunNumber: 613
RotateWave: 0
FirstSample: 0
LastSample: 512
MinDiskSpaceMB: 1000
MaxFileSizeB: 1000000000
CCPort: 3000
CCClient: ihp-pc29.ethz.ch
FirstVMESlot: 1
LastVMESlot: 7
[2008:07:08:12:46:03] DAQ> found mezz. board  0 on VME slot  3 upper, serial #123, firmware revision 5268
[2008:07:08:12:46:03] DAQ> found mezz. board  1 on VME slot  3 lower, serial #122, firmware revision 5268
[2008:07:08:12:46:03] DAQ|B0-1> initialization of board 0 done
[2008:07:08:12:46:03] DAQ|B0-1> initialization of board 1 done
[2008:07:08:12:46:18] USER> info

************************************* DAQ STATUS *************************************

 DAQ: stopped
 Run number: 613
 Run type: data
 Event: 0
 Requested events per data run: 1
 Requested events per ped. run: 100
 Requested events per cal. run: 1000
 Storage directory: /home/scommichau/data/
 Disk space (/home/scommichau/data/) [MB]: 60104
 CC state: disconnected
 CC client: ihp-pc29.ethz.ch
 CC port: 3000
 Total number of CMC boards: 2
 Active CMC boards: all
 Frequency of board 0 set: yes
 Frequency of board 1 set: no

************************************* DRS STATUS *************************************

 Mezz. board index:    0
 Slot:                 3  upper
 Chip version:         DRS2
 Board version:        3
 Serial number:        123
 Firmware revision:    5268
 Temperature:          28.6 C
 Status reg.:          0X00000000
 Control reg.:         0X00020000
   DMODE circular
 Trigger bus:          0X00000000
 Domino wave stopped

 Mezz. board index:    1
 Slot:                 3  lower
 Chip version:         DRS2
 Board version:        3
 Serial number:        122
 Firmware revision:    5268
 Temperature:          30.7 C
 Status reg.:          0X00000000
 Control reg.:         0X00020000
   DMODE circular
 Trigger bus:          0X00000000
 Domino wave stopped

******************************************************************************************

[2008:07:08:12:46:28] CC> date
[2008:07:08:12:46:28] DAQ|B0-1> current date/time is: Tue Jul  8 12:46:28 2008
[2008:07:08:12:46:34] USER> q
[2008:07:08:12:46:35] end logfile
```

E The Raw Data Format

The raw data is written like `RunHeader`, `EventHeader 1`, `CMCData` 1_1, `CMCData` 2_1,... `CMCData` N_1, `EventHeader 2`, `CMCData` 1_2, `CMCData` 2_2,... `CMCData` N_2,...

- **Run Header**:

```
                                //                                size in bytes:
I8  Name[5];                    // name of the structure                      5
I8  DAQVersion[5];              //                                           10
I8  Source[16];                 //                                           26
I8  Type[2];                    // run type (P,C,D)                          28
I8  RunNumber[32];              //                                           60  60

U16 StartYear;                  //                                           62
U8  StartMonth;                 //                                           63
U8  StartDay;                   //                                           64
U8  StartHour;                  //                                           65
U8  StartMinute;                //                                           66
U8  StartSecond;                //                                           67   7

U16 EndYear;                    //                                           69
U8  EndMonth;                   //                                           70
U8  EndDay;                     //                                           71
U8  EndHour;                    //                                           72
U8  EndMinute;                  //                                           73
U8  EndSecond;                  //                                           74   7

F32 SourceRA;                   //                                           78
F32 SourceDEC;                  //                                           82
F32 TelescopeRA;                //                                           86
F32 TelescopeDEC;               //                                           90  16

U32 Events;                     // number of events in the run               94
U32 NCMCBoards;                 // number of used boards                     98
I32 Samples;                    // number of samples (out of 1024)          102
I32 Offset;                     // offset from first sample (<1024)         106  16

F32 NomFreq[MAX_NUM_CMCBOARDS];    // nominal sampling frequency [GHz]      146
F32 BoardTemp[MAX_NUM_CMCBOARDS];  // board temperature [C]                  186  80
```

- **Event Header**:

```
                                //                                size in bytes:
I8  Name[5];                    // name of the structure                      5

U32 EventNumber;                //                                            9
F32 TimeSec;                    // [s] event time stamp, ms precision        13

U16 TriggerType;                //                                           15
```

- **CMC Data**:

```
                                                                  //  size in bytes:
F32 WaveForm[MAX_NUM_CHIPS][MAX_NUM_CHANNELS][MAX_NUM_SAMPLES]; //  81920 = 4*2*10*1024
```

F List of Files

The following directories and files are included, among others:

Directory	File	Task/Content
drsdaq	drsdaq.cpp	Main DAQ program, does global initialization, starts threads for control and data acquisition.
drsdaq	DRSDAQ.conf	Default configuration file, loaded at program startup.
drsdaq	Makefile	Main makefile for the DRS DAQ.
drsdaq	Makefile.general	General definitions for all makefiles.
drsdaq	Makefile.rules	Rules for all makefiles.
DRS	DRS.cc/h	Main DRS class, library functions for the VME DRS board and CMC cards. Requires DRS version 2 or 3 and VME board version 3 [2, 3]. Note: the classes only work with the Concurrent Technologies VME single board computer VP 315 [5].
DRS	mxml.c/h	Midas XML library (S. Ritt).
DRS	strlcpy.c/h	Contains strlcpy and strlcat which are versions of strcpy and strcat, but which avoid buffer overflows (S. Ritt).
DRS	Makefile	Local makefile (follows global rules).
DAQ	DAQ.cc/h	Main DAQ class (thread).
DAQ	DAQReadout.cc/h	Interface between user and DAQ.
DAQ	DAQStatus.cc/h	Houses DAQ and DRS status information.
DAQ	DAQConfig.cc/h	Houses DAQ configuration.
DAQ	ConsoleCommand.cc/h	Handle user input (thread).
DAQ	CCCommand.cc/h	Handle Central Control input (thread).
DAQ	CheckDisk.cc/h	Checks continuously remaining disk space (thread).
DAQ	Log.cc/h	Logging utility.
DAQ	ReadCard.cc/h	Parse DAQ configuration file (F. Goebel).
DAQ	Utilities.cc/h	Utility routines.
DAQ	socklib.c/h	TCP socket interface.
DAQ	Makefile	Local makefile (follows global rules).
utilities	writerawfile.cpp	Test program writing raw files.
utilities	inspectrawfile.cpp	Test program reading raw files.
utilities	remotecontrol.cpp	Test program providing a simple network interface to the DAQ program.
utilities	Makefile	Local makefile.
VME	atlas	Shared libraries, drivers and debug tools [7] for the CT VME controller.
VME	struck	Shared libraries and drivers for the Struck VME controller.
root	...	ROOT [6] configuration files and template macro to read raw data.
calib	...	Default target directory for the response calibration files.
log	...	Target directory for the log files. It is specified in the default DAQ configuration file.
doc	manual.ps/pdf	This documentation.

References

[1] Latest version of the DRS DAQ program and documentation:
`http://ihp-pw1.ethz.ch/commichau/.CTX/daq/`

[2] DRS documentation web pages, `https://midas.psi.ch/drs/.`

[3] `stefan.ritt@psi.ch`, `boris.keil@psi.ch`

[4] M. Schneebeli, The drift chambers of the MEG experiment and measurement of the ρ-parameter in the Michel spectrum of the muon decay, DISS. ETH No. 17719, 2008.

[5] Concurrent Technologies web pages, `http://www.gocct.com/sheets/vp315022rc.htm.`

[6] ROOT web pages, `http://root.cern.ch.`

[7] VMEbus Application Program Interface, ATLAS Internal Note ATL-D-ES-0004, V 1.7, 14 June 2007.