MAGIC
Major Atmospheric
Gamma Imaging
Cerenkov Telescope

MAGIC-TDAS 00-09/3
000808/JCortina

Description of the DAQ raw data format Version 2.0

F. Goebel, J. Cortina and J. A. Coarasa

10th February 2004

Abstract

This document describes the raw data format which the DAQ writes to disk. First a brief overview of the FADC and the readout setup is given and then every variable of the raw data format is described in detail. The document is based on a previous version which is replaced by this new version.

Contents

1 Readout setup and formatting process

The setup of the FADC system and computer PCI readout has been described in detail elsewhere ([1, 2]. See DAQ web page in Munich for both documents). Let us go over some issues relevant to this discussion.

At arriving to the electronics room each pixel signal is split into two branches, one of which is further amplified (in the following *High Gain Branch*), delayed and **inserted** right after the original signal coming through the other branch (so called *Low Gain Branch*) **only** when it exceeds a certain level. This "insertion" is done with a fast GaAs switch. Each channel switches to low gain or not independently. The complete signal enters a FADC channel. (The electronic chain is illustrated at the end of the section). Each FADC crate contains 4 boards and each board contains 8 channels. There are a total of 19 crates with 74 FADC boards + 1 board with digital modules. The mapping from camera pixel to FADC channel is (seemingly) arbitrary, so a table containing the spiral pixel-channel assignment is essential for any further data processing. This table is stored in each run header.

In its present form the electronics does not supply the DAQ with any mask indicating the pixels for which there is a low gain signal. It does not even supply the DAQ with a 0/1 flag indicating if at least one channel has switched to low gain. However, two variables are allocated for this mask and this flag in the raw data structure. This is just in case that the hardware design changes or a fast algorithm can be implemented in the DAQ to find and reject online channels where no low gain signal is present.

Each board delivers a pre-specified number of samples (most probably ~ 30) for each of the 8 ADCs along with some additional variables once for all 8 channels: the clock tick (currently running with 9.375 MHz) at the time of the trigger, an event number, the board number and the so called AB-Flags for each of the 8 channels, which allow to resolve the 2 slice timing uncertainty inherent to the FADC design. These variables are stored together with the data for synchronization and test purposes.

There is one special board equipped with digital modules which are used to record extra digital information together with each event coming from the trigger system, the calibration sytem and a precise absolute GPS clock.

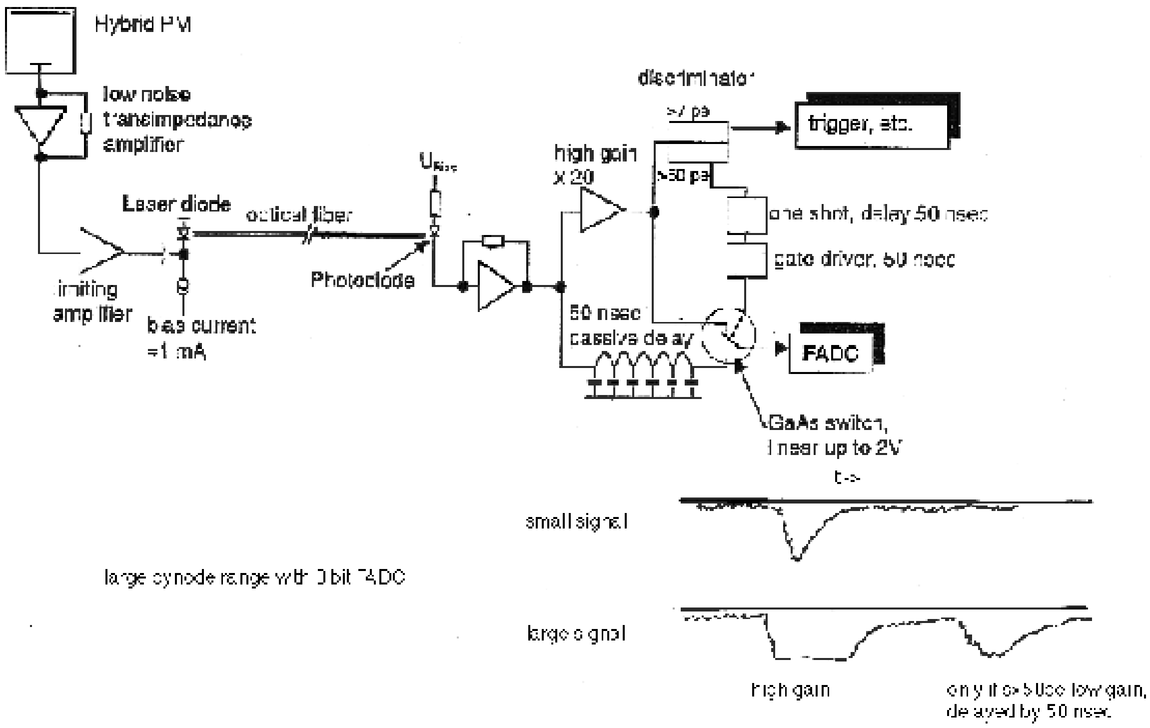
All boards (including the digital one) are read by one PCI cards into a DAQ computer. The DAQ is a multiprocessor PC running Linux and reads event after event and saves them to a file in what is called *DAQ raw data format*. This is the format that will be describe in the following sections.

During the day (or already during the data-taking night with some delay) these DAQ raw files are saved to LTO 200 tapes (along with the central control data files containing currents, scalers, weather, lidar, status... information) and deleted to make space for new data. This is done using a tape robot.

In the future a *preprocessing* ("rootification" or "merpping") may be applied before saving the different files to disk which would synchronize the central control different streams with the events and write them in ROOT format (which we may call *preprocess format*), possibly in one single file per run.

Let us point out that the DAQ has to write the raw data in this "custom" format and not in a rather more comfortable ROOT format because (i) it takes a non-negligible time to convert binary data into ROOT format [3] and (ii) any ROOT format would always consume more disk space than an optimized custom format. In the event that the computer were eventually left with always enough CPU time (due e.g. to a system upgrade), we would switch to a more standard ROOT format. The same question can be raised for the *preprocessing* itself. If it turned out to be impossible for it to perform all the above-mentioned tasks with an acceptable rate, we would be left only with the possibility to save the DAQ raw format to DLT and leave any rootification or synchronization process for the analysis program itself.

In any case the DAQ raw format contains already all the input variables that the DAQ readout system will provide the analysis program with. Hence any possible ROOT format will no doubt be very similar to the format described here. It is left now for the experts on ROOT to define the appropriate classes involved in the *preprocess format*.



2 Summary of variable names

Brief summary of variable names for further easy reference. The variables are defined in next section.

In the following the following conventions for C variable types: "i8, i16, i32" mean 8, 16... bits long signed integers, "u8, u16, u32" mean 8, 16... bits long unsigned integers, float32 means 32 bits long float. "little endian" byte order is used.

RUN HEADER

MagicNumber	u16
FormatVersion	u16
SoftVersion	u16
RunType	u16
RunNumber	u32
ProjectName	char*22
SourceName	char*12
SourceRA	float32
SourceDEC	float32
SourceEpochChar	char*2
SourceEpochDate	u16
NumBoards	u16
NumPixInBoard	u16
LoGainNumSamples	u16
HiGainNumSamples	u16
NumEvents	u32
StartYear	u16
StartMonth	u8
StartDay	u8
StartHour	u8
StartMinute	u8
StartSecond	u8
EndYear	u16
EndMonth	u8
EndDay	u8
EndHour	u8
EndMinute	u8
EndSecond	u8
PixAssignment[]	NumBoards × NumPixInBoard × u16
Dummy[]	char*16

EVENT HEADER

DAQEvtNumber	u32
TimeSec	u32
TimeSubSec	u32
FstLvlTrigNumber	u32
SecLvlTrigNumber	u32
TriggerPattern	u32
CalibrationPattern	u32
TriggerType	u16
AllLowGainON	u16
PixLowGainON[]	NumBoards × NumPixInBoard × 1 bit or more precisely ceil(NumBoards × NumPixInBoard / 8) × u8
Dummy	u32

BOARD DATA * NumBoards

FADCBoardNumber	u16
FADCEvtNumber	u32
FADCClockTick	u32
ABFlag	u8
FADCSamples[]	u8 × NumPixInBoard × ([LoGainNumSamples +] HiGainNumSamples)

3 Full Description of variable names and file name

Each DAQ raw file corresponds to a data-taking run. It consists of a run header with some general information valid for all the events in the run and the events themselves. Each event consists of an event header plus event data grouped in normally 74 bunches of "board data", i.e., data for each bunch of 8 channels in a board.

3.1 RAW FILE NAME

This convention has been taken from the minutes of the "MAGICT Control Software Meeting March 2000" [3] and slightly modified in this new version.

The name consists of five sections of numbers or upper case letters. The sections are separated by underscores (_). The raw data files have the extension ".raw". The sections are (left to right):

- Date of the evening after (before in the previous version) the shift start (YYYYMMDD, 8 digits fixed length)
- Run number within the shift (nnnnn, 5 digits fixed length)
- Run type (one character to determine the run type: D = data run, C = calibration run, P = pedestal run, ...)
- Project ID (up to 20 characters without hyphens which denote the source and the proposal for which the observation is done)
- event or slow (one character: E = event data, S = slow data)

Examples for filenames produced by the DAQ are 20010621_00001_D_MKN501FIRSTLIGHT_E.raw or 20010621_00001_D_MKN501FIRSTLIGHT_S.raw.

3.2 RUN HEADER

MagicNumber

u16

These two bytes allow to identify the file as a MAGIC raw data file. The highest bit of both bytes in the magic number is always set to 1. By using this convention no operating system will identify the file as an ASCII file.

A file that is correctly closed by the DAQ has the MagicNumber hex C0C0 = binary 1100000011000000 = decimal 49344. When the DAQ starts a new raw data file it temporarily writes the MagicNumber hex C0C1 = binary 1100000011000001 = decimal 49345. This is changed to the (final) MagicNumber hex C0C0 when the run is finished and the raw data file is closed. The MagicNumber hex C0C1 therefore identifies a raw data file which was not correctly closed and may be corrupted.

FormatVersion

u16

Version of data format. Whenever the raw data format is changed this number is increased by 1. The present raw data format is 2.

SoftVersion

u16

Version of DAQ program. It does not neccessarily mean any change in data format.

RunType

u16

Type of run: normal data run (0), pedestal (1), calibration (2)...

RunNumber

u32

Run number. This number increases by 1 for each new run.

ProjectName

char*22

Code name of the project. The project name is actually 20 characters long but we allow for 2 chars more to have space for the end-of-string character.

SourceName

char*12

Source name using 12 characters. The project name is actually 10 characters long but we allow for 2 chars more to have space for the end-of-string character.

SourceRA and SourceDEC

float32

Source coordinates.

SourceEpochChar and SourceEpochDate

char*2, float32

Epoch for the source coordinates. SourceEpochChar can be “J” for julian coordinates, “B” for besselian coordinates, “D” for “epoch of date”. The second character must be an end-of-string character.

NumBoards

u16

Number of boards in the system. This variable determines the number of blocks of board data that will be saved for each event.

NumPixInBoard

u16

Number of pixels in one board (for the time being always 8). This variable determines the number of pixels which will be saved for each board data block.

LoGainNumSamples, HiGainNumSamples

u16

Number of time samples digitized in the high gain and low gain branches. The DAQ computer may configure this number in the FADC cards. For the time being we use 15 samples of low gain and 15 samples of high gain. These numbers may increase in the future in order to be able to provide a pedestal together with each event.

The low gain samples may actually not be there if LowGainON flag for this particular event is off. However, at the moment LowGainOn is not filled and is undefined.

NumEvents

u32

Total number of events in the run.

StartYear, StartMonth, StartDay, StartHour, StartMinute, StartSecond

u16, u8, u8, u8, u8, u8.

Date and time of the DAQ computer at the start of the run. The DAQ time should be UTC.

EndYear, EndMonth, EndDay, EndHour, EndMinute, EndSecond

u16, u8, u8, u8, u8, u8.

Date and time of the DAQ computer at the end of the run. Again the DAQ time should be UTC.

PixAssignment[]

u16 × NumBroards × NumPixInBoard ×

Pixel assignments. For each channel in the ADC system the corresponding spiral pixel number on the camera is provided.

Dummy[]

char*16

Space left empty for future variables.

3.3 EVENT HEADER

DAQEvtNumber

u32

Event number. This number comes from a counter in the DAQ program and may not coincide with the other event numbers described below.

NOTE: The next 6 variables are read from the digital module.

TimeSec

u32

Absolute time up to second precision as determined using a GPS clock. This time is encoded as follows: *format to be added*

TimeSubSec

u32

5 MHz clock tick, which is reset with the GPS clock precisely at the start of every second.

FstLevTrigNumber and SecLevTrigNumber

u32

First and second level trigger number from a counter in the trigger system. These numbers should always be greater than DAQEvtNumber since the FADC system may lose triggers during its dead time. Both trigger numbers may used to estimate the system dead time. Currently these numbers are not filled.

TriggerPattern

u32

Trigger pattern for this particular event. See trigger notes for details.

CalibrationPattern

u32

Calibration pattern for this particular event. See calibration notes for details.

TriggerType

u16

Trigger type (normal trigger (0), pedestal (1), calibration (2)...).

AllLowGainON

u16

One flag indicating if any of the pixels have low gain signal (1) or not (0). **NOTE**: This info is not provided to the DAQ by hardware in the current setup.

PixLowGainON[]

1 bit $\times$ NumBoards $\times$ NumPixInBoard $\times$

A flag mask indicating for each single channel if the low gain is there (1) or not (0). **NOTE**: This info is not provided to the DAQ by hardware in the current setup.

Dummy

u32

Space left empty for future variables.

3.4 BOARD DATA * NumBoards

FadcBoardNumber

u16

Board number provided by each board identifying the hardware board number (set with dip-switches on each board).

FADCEvtNumber

u32

Event number as determined by an internal counter in the FADC board. This number should be equal to DAQEvtNumber but not to TriggerNumber. All the boards should have the same FADCEvtNumber so this number can be used to check the board synchronization.

FADCClockTick

u32

Tick of the board clock (currently 9.375 MHz) as determined using a counter in the FADC board. This tick will reset automatically to 0 every $\sim$107 ns * 2^{32} = $\sim$458 s.

ABFlag

u8

A bit pattern with the ABFlag for each of the 8 FADC channels. Although the FADC digitizes with 300MHz the 8 bit digital output data is provided via two 150 MHz output lines (A and B) with a half period relative phase shift (ping pong technique). Since the further readout always reads 2 bytes simultaneously (the bytes from output A and B) there is a time jitter of 2 slices between the trigger and the start of the time window which is read out. The ABFlag indicates whether the trigger arrived in the moment when the 150 MHz output clock was in state A(0) or B(1). The DAQ itself uses the ABFlag to resolve the AB timing uncertainty by shifting the data by 1 byte when the ABFlag is 1. Since the baseline (pedestal) of the A and the B bytes may be different it is desirable to know which of the slices is A and which is B. Therefore the ABFlag is kept in the raw data format.

FADCSamples[]

u8 × NumPixInBoard × ([LoGainNumSamples +] HiGainNumSamples)

THE ACTUAL DATA. Samples of the pixel signal in 1/300MHz time slices. Each sample has 8 bits size.

The number of time samples per pixel is determined by the variables LoGainNumSamples + HiGainNumSamples which are fixed for a given run.

3.5 DATA SWAPPING AND ZERO PADDING

A word of caution has to be said about swapping and padding. Different compilers and different arquitectures swap data in different ways. In particular Alpha stations and Linux PCs internally use “Little Endian”. Since both DAQ and data analysis are based on this kind of computers Little Endian is used when saving to disk.

In addition some compilers and some machines introduce extra bytes when saving 8 bit or 16 bit variables (zero padding). **No** zero padding is allowed for in the raw data at all.

Acknowledgements

Thanks go to all the participants in the MAGICT control meeting which helped find and define all the variables in the DAQ output and to the authors of the previous versions of this document.

References

[1] J. Cortina, Proc. *VI Toward a Major Atmospheric Cherenkov Detector Workshop*, Snowbird USA, 1999.

[2] J. Cortina and J. A. Coarasa, contribution to the MAGICT Control Software Meeting, Hinterriss, March 2000.

[3] T. Schweizer, contribution to the MAGICT Analysis Software Meeting, Hinterriss, March 2000.

[4] D. Petry, "MAGICT Control Software Meeting 15/16 March 2000 Minutes".