DAQ User Guide

for the Muon g−2 Experiment

R. Chislett,, University College London
S. Ganguly, University of Illinois
W. Gohn, University of Kentucky
T. Gorringe, University of Kentucky
F. Hahn, University of Kentucky
T. Stuttard, University College London

Contents

***

Introduction

***

Installation

***

2.0.1 Prerequisites

The DAQ for muon $g-2$ is based on MIDAS, so it is necessary to first install MIDAS, as well as other prerequisites, before installing the DAQ software. We will also need to install ROOT, which is a prerequisite for ROME, our data quality monitoring software.

First, you will want to use yum to install some of the basic prerequisite packages. Two packages that are required by our DAQ software can be installed by:

```
yum install mysql-devel readline-devel zlib-devel
```

Next, install all the prerequisites for ROOT:

```
yum install libXll-devel libXpm-devel libXft-devel libXext-devel
```

, and you might as well install the optional ROOT packages as well:

```
yum install gcc-gfortran openssl-devel pcre-devel \
        mesa-libGL-devel glew-devel ftgl-devel mysql-devel \
        fftw-devel cfitsio-devel graphviz-devel \
        avahi-compat-libdns_sd-devel libldap-dev python-devel \
        libxml2-devel gsl-static
```

Now, you can install ROOT using your favorite method. I prefer to use git, so execute

```
git clone http://root.cern.ch/git/root.git
cd root
git tag -l
git checkout -b v5-34-08 v5-34-08
./configure
make
```

If you are using the calorimeter readout code that processes data on the GPU, you must also install CUDA. To install CUDA, follow the instructions at https://developer.nvidia.com/cuda-downloads.

Now, you are ready to install MIDAS and ROME.

2.0.2 MIDAS Installation

To install MIDAS, first obtain our gm2midas git repository from Redmine.

```
git clone ssh://p-gm2midas@cdcvs.fnal.gov/cvs/projects/gm2midas
```

You will need the environment variable MIDASSYS set to the give the path to your midas installation. For example, if your username is daq and you checked out gm2midas in your home directory, you would

```
export MIDASSYS=/home/daq/gm2midas/midas
```

Now you go to the directory where your MIDASSYS has been defined, and install midas using

```
cd $MIDASSYS
gmake
sudo gmake install
```

The last command, "gmake install", is optional and will copy the MIDAS installation to your system (not good on a shared computer, for instance).

2.0.3 ROME Installation

Since you have already checked out gm2midas, you already have ROME, and it just needs to be compiled (if you did not already checkout gm2midas, refer back to the previous section for instructions on how to do so). You need to set the ROMESYS environment variable to the subdirectory gm2midas/rome, navigate there, and type "make".

```
export ROMESYS=/home/daq/gm2daq/rome
cd $ROMESYS
make
```

2.0.4 Checking out gm2daq

Now, to obtain our DAQ code, checkout our git repository

```
git clone ssh://p-gm2daq@cdcvs.fnal.gov/cvs/projects/gm2daq
```

If you are a DAQ user, you should be on the master branch, and you only need to compile. If you are a DAQ developer, or you would like to test out the newest "experimental" version of the DAQ softwre, you should checkout the develop branch.

```
cd gm2daq
git checkout develop
```

We are currently using Makefiles for compilation. To compile any portion of the DAQ software, navigate to the desired subdirectory and type "make". The directories requiring compilation for normal g-2 running with AMC13 readout are

```
gm2daq/eventbuilderNew/
gm2daq/amc13/amc13StandaloneMAN_2014-05-12/
gm2daq/frontends/CaloReadoutAMC13/
gm2daq/frontends/MasterGM2/
```

If you also need to simulate data (i.e. you do not have an AMC13 as an input source), also compile

```
gm2daq/frontends/AMC13Simulator/
```

2.0.5 Environment Setup

Our environment is most easily setup using a script called setup.sh that is located in the main gm2daq repository. It is recommended to source that script from your own .bash_profile file. You may need to edit the values of some of the necessary environment variables if you are running on your own machine. The variables that this script sets up are:

- GM2DAQ_DIR: The location of your gm2daq repository.
- ROOTSYS: The location of your root installation.
- MIDASSYS: The location of your MIDAS installation.
- ROMESYS: The location of your ROME installation.
- CACTUS_ROOT: Where your IPBUS is installed (usually /opt/cactus).
- AMC13_STANDALONE_ROOT: The location of your AMC13 software.
- CUDASYS: Where your CUDA is installed (usually /usr/local/cuda).
- PATH: Adds the bin directories corresponding to the variables defined above to your PATH.
- LD_LIBRARY_PATH: Add the lib directories corresponding to the variables defined above to your LD_LIBRARY_PATH.

Running the DAQ

***

Before starting MIDAS, you must first define an experiment using the exptab file. By default, MIDAS will look for this file at /etc/exptab, but it is possible to put it somewhere else and tell MIDAS how to find it with an environment variable. You can define up to ten experiments in the exptab, with the format "NAME directory user", where NAME is the name of the experiment, the directory is where MIDAS will write the files necessary to define the experiment, and user is the username of the user who can run this experiment. As an example, an exptab file to define three experiments for users bernie, hillary, and donald , would look like.

```
EXP1 /home/bernie/exp bernie
EXP2 /home/hillary/exp hillary
EXP3 /home/donald/exp donald
```

It is then necessary that each user creates the directory "exp" in their home directory.

To fully run our DAQ software, you need to run three MIDAS programs, and at least two of the programs we compiled in the previous section.

The first program to start is the MIDAS mserver. This is the main server that communicates between MIDAS programs running on different machines. If you are the only user, you can run the mserver under your own username, but we have learned that it works best if run under root, in particular when you have multiple users sharing a machine. One instance of mserver will handle all running experiments. The standard usage is

```
mserver -D
```

Next you have to start the MIDAS webserver, mhttpd, which allows the user to control an experiment via a MIDAS web interface. The standard usage is

```
mhttpd -e GM2 --mg 8080 -D
```

where the experiment name here is GM2 and you are using port 8080. You must run one instance of mhttpd for each experiment running on a system, and each must use a unique port. Since we typically run behind a firewall, we do not use all of the mhttpd securiy features, but it is possible to replace the mg flag with "https 8444", which will allow you to set a password for your webserver if desired.

The MIDAS program mlogger writes the data to disk. It requires only that you specify the experiment name, and the rest of it's settings are controlled via the ODB.

```
mlogger -e GM2
```

It is also necessary to run the event builder, which assembles data fragments from all specified frontends and combines them into a single event. This program is located in gm2daq/eventbuilderNew. After compiling, start the event builder using

```
./mevb -e GM2 -b BUF
```

where again GM2 is the experiment name and BUF is the prefix used to identify each unique event buffer. The event buffers must be defined for each frontend in it's ODB Common block, and they must each start with BUF. For instance, we generally define the buffers as BUF01, BUF02, BUF03, etc.

Finally, start the frontend code for each frontend you are running. As a minimum, you should start MasterGM2 and one other fronend. The usage for these will be discussed in subsequent sections.

Event Builder

***

4.0.6 Running Event Builder

It is also useful to run the event builder, which assembles data fragments from all specified frontends and combines them into a single MIDAS event. This program is located in gm2daq/eventbuilderNew. After compiling, start the event builder using

```
./mevb -e GM2 -b BUF
```

where again GM2 is the experiment name and BUF is the prefix used to identify each unique event buffer. The event buffers must be defined for each frontend in it's ODB Common block, and they must each start with BUF. For instance, we generally define the buffers as BUF01, BUF02, BUF03, etc.

For each enabled frontend, the user may specify whether or not to send it's data to the event builder by setting an ODB flag under the Global settings for that frontend called "Send to Event Builder", i.e.

```
"/Equipment/AMC1301/Settings/Globals/Send to Event Builder" y
```

4.0.7 ODB Options

MasterGM2 Frontend

***

5.0.8 Functionality

5.0.9 ODB Options

5.0.10 Bank Output

Calorimeter Frontends

***

6.0.11 Functionality

The calorimeter readout frontend is located in the directory gm2daqfrontendsCaloReadoutAMC13, and will be displayed with the frontend name "AMC13XX", where XX is an integer corresponding to the index of the μTCA crate, 01-24 for the calorimeters.

Usage is:

```
./frontend -e GM2 -h <backend_name> -i <i>
```

Where the backend_name is the hostname of the computer where your MIDAS mserver is running, usually g2be or g2be1, and "i" is a frontend index, 1-24 for the calorimeter systems.

6.0.12 ODB Options

The ODB settings for the CaloReadoutAMC13 frontend are subdivided into Global setting, which control the basic frontend bahaviour, AMC13 settings, used to configure the AMC13 itself, Link01 settings, used to configure the DAQ link between the readout and source, Calorimeter settings for slow control parameters, and settings for each of the 12 Rider slots in the crate.

The global settings are used to set up the frontend for data taking, allowing the user to choose the type of data that will be saved, and how it will be processed. The available options are:

6.0.13 Bank Output

The CaloReadoutAMC13 frontend outputs MIDAS banks with a variety of information, including Raw waveforms, T and Q method derived datasets, and header and trailer information. All of the included banks are summarized in Table. 6.1.

Bank Name	Function
CBxx	Header information from the AMC13 and Riders for each fill.
CZxx	Trailer information from the AMC13 and Riders for each fill.
CRxx	Raw data, which is often prescaled, so not written for every event.
CTxx	T-method processed islands.
CQxx	Q-method decimated sums.
CHxx	Q-method event-by event summed histograms.
CLxx	Information on timing of the data processing.

Table 6.1: The list of banks that can be printed by the CaloReadoutAMC13 frontend. The xx denotes the frontend index of 01-24, which is written into the bank name.

Fill Type	Characteristics	CB,CZ	CT, CQ, CH	CR	CL
muon	Single, continuous waveform.	yes	yes	pre-scaled	yes
laser	One or more WFD-chopped waveforms.	yes	no	yes	yes
pedestal	Raw pedestals	yes	no	yes	yes

Table 6.2: The data saved for each fill type by the CaloReadoutAMC13 frontend.

The data will come in three fill types, which the CaloReadoutAMC13 will determine from the AMC13 headers. The fill types are muon, laser, or pedestal. For muon fills, the full processing in the GPU is turned on for each event (if the appropriate ODB flag is checked), but for the laser and pedestal types, only the raw data is written. Details of which banks are saved for each fill are given in Table 6.2.

Tracker Frontends

***

7.0.14 Functionality

7.0.15 ODB Options

7.0.16 Bank Output

Fiber Harp Readout

***

8.0.17 Functionality

8.0.18 ODB Options

8.0.19 Bank Output

ROME

***

How to use ROME.

Troubleshooting Tips

***

Troubleshooting tips.

GPS Setup

***

GPS setup.

AMC13 Setup

***

To set up the AMC13, it must be installed in the upper slot 13 of the μTCA crate, with the MCH in the lower slot, as shown in Fig. B. All communication with the AMC13 is performed via the network connection on the MCH. Data is read out of the AMC13 via a 10 Gb fiber that should be connected to the top (DAQ0) SFP+ port on the AMC13.

For general use, all configuration of the AMC13 should be handled automatically by the DAQ code. For expert users, it may be necessary to configure and run the AMC13 manually using the program amc13tool2.exe, which is provided in the gm2daq repository.

Figure B.1: An AMC13 in a μTCA crate.