GEM 用 DAQ ミドルウエア マニュアル

MLF 標準 DAQ ミドルウエアから独自の Gatherer/Monitor を作成するために

初版：2009 年 2 月 4 日
修正：2009 年 3 月 30 日
安　芳次

目次

はじめに

この文献は MLF 標準 DAQ ミドルウエアから独自の Gatherer/Monitor を作成するために作られたものである。しかし、参考のためこれから DAQ ミドルウエアを始める人のための簡単な文献の紹介をする。文献一覧は巻末の「参考文献」にまとめた。

MLF 標準 DAQ ミドルウエアではなく大学・研究所一般用 DAQ ミドルウエアパッケージがある。それは下記の URL から取れる。

http://www-online.kek.jp/~nakayosi/DAQMW/sl52.zip

VMwareplayer を使ったオールインワンの DAQ ミドルウエアであり、初心者には適当なものである。VMwareplayer の整備については次章を参考にすること。

VMwareplayer の整備

千代さんが用意した VMwareplayer で使用するイメージは下記にある。

http://greentea.kek.jp/sendai/sl52_OpenRTM-0.4.1-9.zip

DAQ ミドルウエア 2008.12 版のコンポーネントパッケージは入っていないので、各自入れる必要あり。root のパスワードは abcd1234、登録済みの一般ユーザは daq(パ

スワード daqone)。パスワードはインストール後必ず変更すること。VMWarePlayer のインストールについては下記の URL を見ること。

http://greentea.kek.jp/sendai/

今回はこのパッケージは使わない。以前入れた vmwareplayer を使った。

VMWarePlayer を走らせる。以下 VMWarePlayer の簡単な操作を列記する。

- ctrl+Alt は画面を VMWarePlayer の中から出るときに使う。
- マウス右クリックで open terminal を選んで端末を開く。
- ctrl+"+"は terminal を大きくする時に使う。
- ctrl+"-"は terminal を小さくする。
- カーソルで黒くマークしたテキストをコピー・ペーストするには、マウス右クリックで copy し paste すればよい。あるいは、ctrl+shift+"c"でコピーし、ctrl+shift+"v"でペーストする。

GEM 用 DAQ ミドルウエアのアップグレード

DAQ ミドルウエア１２月版を使ったアップグレードを行った。使用するアカウントは/home/daq である。下記の URL にある「MLF 中性子用 DAQ ミドルウエアインストールおよび操作マニュアル」は MLF 中性子用標準 DAQ ミドルウエアのインストールや操作方法を記述したもので大変参考になるので一読をお勧めする。

http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/tars.2008.12/DAQMM.pdf

また、このソフトウエアはハードウエアなしにソフトウエアエミュレータを使って運用することができるので、ソフトウエアの確認や初期における動作確認に利用できる。下記の文献も参考となる。

http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/tars/SiTCP-PSD-Utils/SiTCP-PSD-Utils.pdf

パッケージの取得と展開

上記の URL から update 用の OpenRTM を取得する。

http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/i386/OpenRTM-aist-0.4.1-9.KEK.el5.i386.rpm

OpenRTM-aist がインストールしているかどうか確かめる

% rpm –qi OpenRTM-aist

次のようにして update する。rpm を使った。

% rpm –Uhv OpenRTM-aist-0.4.1-9.KEK.el5.i386.rpm

これで OK。

次に下記の URL から下記のファイルを取得する。

http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/tars.2008.12/

DaqComponents.2008.12.tar.gz

SiTCP.2008.12.tar.gz

gnuplot.bin.tar.gz

manyo.bin.tar.gz

www.2008.12.tar.gz

そして展開する。WEB で取ったので、/home/daq/Desktop ディレクトリに入った。

```
% cd /home/daq
% tar zxvf ~/Desktop/SiTCP.tar.gz
```

これで、次のディレクトリが作られた。

/home/daq/SiTCP

/home/daq/lib

さらに、GNUPLOT と MANYO ライブラリの展開。

```
% tar zxvf ~/Desktop/gnuplot.bin.tar.gz
% tar zxvf ~/Desktop/manyo.bin.tar.gz
```

これで、次のディレクトリが作られた。

/home/daq gnuplot

/home/daq manyo

さらに www の展開。

```
% tar zxvf ~/Desktop/www.2008.12.tar.gz
```

これで、次のディレクトリが作られた。

/home/daq/www

最後に、本体である DaqComponents.2008.12.tar.gz の展開

```
% tar zxvf ~/Desktop/ DaqComponents.2008.12.tar.gz
```

これで、次のディレクトリが作られた。

/home/daq/DaqComponents

新しいシステム設定および既存システム設定の確認

Xinetd の設定は、MLF 標準 DAQ ミドルウエアには標準でセットアップすべきものだが、GEM の場合は、ROOT 解析システムでの表示は X を利用したものとなっているため、実際は不要である。しかし、規模が大きく解析結果のファイル数が増大した場合、X を使用しないで解析中に画像ファイルを作成してそれを WEB で見るのが有用となる場合がある。MLF 標準 DAQ はその方式を取っているので、当面 GEM では使わないが、セットアップを行っておく。

xinetd の設定

まず yum を使って xinetd をインストールする。root になって、

yum install xinetd

/etc/services に下記の行を追加。

bootComps　50000/tcp　# boot DAQ-Components

xinetd 用ファイルのコピーを行う。

cp /home/daq/DaqCompoments/bootComps-xinetd /etc/xinetd.d/bootComps

pythonプログラムをotherでも実行できるようにするために、

chmod 755 /home/daq/DaqCompoments/bootComps.py

ブート時の自動起動のために

/sbin/chkconfig xinetd on

/etc/init.d/xinetd stop

/etc/init.d/xinetd start

動作の確認のため下記のコマンドを打ってみる。

% echo pwd | nc 192.168.174.130 50000

Bad command: pwd

このメッセージは/home/daq/DaqComponents/bootComps.py の中で、受け取ったコマンドが期待したものとは違ったときに出すもの。

/etc/xinetd.d/bootComps の中で、/home/daq/DaqComponents/bootComps.py を起動コマンドに定義している。

これで、xinetd の設定は終了する。

www の設定の確認

下記のファイルの内容をチェックする。

/etc/httpd/conf.d/python.conf

```
<Directory /var/www/html/daq>
  AddHandler mod_python.py
  PythonHandler mod_python.publisher
  PythonDebug On
  PythonPath "['/var/www/html/daq', '/var/www/html/daq/operatorPanel'] + sys.path"
</Directory>
```

URL で参照すべきディレクトリを/home/daq/www にアサインしてあることを確認する。

```
% ls –l /var/www/html/daq
/var/www/html/daq -> /home/daq/www
```

runNumber.txt を other からの書き込みを可能にする。

```
% chmod 666 runNumber.txt
```

GEM エミュレータの設定確認

GEM エミュレータは１つの IP アドレス(デフォルト 192.168.0.16)を使って GEM ハードウエアのエミュレーションを行う。下記は、IP エイリアスの eth0:0 が 192.168.0.16 にアサインされ、利用可能になっているかどうか確認したものである。

```
% /sbin/ifconfig eth0:0
```

192.168.0.16 がアサインされている。つまり、エミュレータ用の IP アドレスはこれになる。

もしアサインされていなかったら次の手順でアサインすること。

```
% su
#/sbin/ifconfig eth0:0 192.168.0.16
```

これで 192.168.0.16 の IP が利用できるようになるが、システムが再起動しても利用できるようにするためには、つぎのようにしておく必要がある。

```
# cd /etc/sysconfig/network-scripts
# cp ifcfg-eth0 ifcfg-eth0:0
```

これで ifcfg-eth0:0 を編集する

```
DEVICE=eth0:0
IPADDR=192.168.0.16
ONBOOT=yes
BOOTPROTO=none
NETMASK=255:255:255:0
```

このように変更してから、

```
# /etc/rc.d/init.d/network restart
```

これで 192.168.0.16 の IP が利用できるようになる。

エミュレータを次の手順で走らせる。

```
% cd GEM/test
su
# source setup.sh
# ./DataServerGem tt.dat 23 192.168.0.16
```

これでほっておいてよい。

既存ファイルの編集

これからはたくさんのファイルの変更をする必要があるので１つ１つ説明する。ただし、Gatherer と Monitor の変更の詳細は、今後異なる検出器のたびに役立つので別途他の章で扱うこととし、ここでは変更すべきファイルとその概要にとどめる。

run スクリプト周り

下記の run.sh や run-comps.sh は xinetd 機能を用いてプロセスの起動をおこなうために作られている。Xinetd の設定でも述べたが、GEM の場合は下記の変更は不要で、その代わり、X を使った ROOT の表示方法をとるためにプロセス起動を行うスクリプト run.sh を別に用いることとする。しかし、MLF 一般にはこの機能が必要なので設定しておく。

/home/daq/DaqComponents/run.sh の変更

```
remote_host_addrs = (192.168.174.130)
daqoperator_addr='192.168.174.130'
```

home/daq/DaqComponents/run-comps.sh の変更

```
export ROOTLIB=/usr/local/root/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOBLIB
```

別に用意した run.sh を下記のようにコピーして、コンソールモードを入力できるように多少変更した。

```
% cd /home/daq/DaqComponents
% mv run.sh run.sh.new
% cp /home/yasu/ DAQMW-MLF-RC1-GEM/run.sh
```

/home/daq/DaqComponents/setup.sh の確認

ROOT のパス・Socket などライブラリのパスがあるかどうかチェックすること。

/home/daq/DaqComponents/config.xml の変更

hostAddr タグで localhost.localdomin で以前よかったが 192.168.174.130 にしないとだめになった。GATENET は外す。SiTCP は 192.168.0.16、１台のみ。

Gatherer コンポーネント周り

/home/daq/DaqComponents/src/Selector.h の変更

Template を使った GemModule モジュールのための Selector を作ったので変更の必要がある。もともとは PSDMODULE/NEUNET というモジュールを直接扱っているが、GemModule の方ではこのようなやり方ではなく、どのようなモジュールでも入れられるように template 化した。

GemModule モジュールを/home/yasu/ DAQMW-MLF-RC1-GEM/lib/Module からコピー

/home/daq/SiTCP/CPP/Module をコピー。変更も。

/home/daq/SiTCP/CPP/Module/Makefile の変更

Gatherer コンポーネントの変更

/home/daq/DaqComponents/src/Gatherer.h

/home/daq/DaqComponents/src/Gatherer.cpp

/home/daq/DaqComponents/src/Makefile.Gatherer

PSD から GEM に変更する。使用するモジュールも異なってくるので、大幅な変更となる。詳しくは別な章で。

/home/daq/GEM/test を/home/yasu/GEM/test からコピーして作成する。

Monitor コンポーネント周り

Monitor の変更

/home/daq/DaqComponents/src/Monitor.h

/home/daq/DaqComponents/src/Monitor.cpp

/home/daq/DaqComponents/src/MonitorComp.cpp

/home/daq/DaqComponents/src/Makefile.Monitor

MLF 中性子 DAQ ミドルウエアでは解析用のツールとして Manyo ライブラリと GNUPLOT を使っているが、GEM では ROOT を使った。詳しくは別な章で。

ROOT がインストールされているかどうかチェックすること。

```
% which root
/usr/local/root/bin/root
% root
```

GEM 用 DAQ ミドルウエアの動作確認

新しい DAQ ミドルウエアの動作確認を行った。コンソールモードと HTTP モードがある。

コンソールモードでの動作確認

GEM エミュレータをはしらせる

端末を開いて、下記のようにエミュレータを走らせる。

```
% cd GEM/test
su
```

```
# source /home/daq/DaqComponents/setup.sh
# ./DataServerGem tt.dat 23 192.168.0.16
```

DAQ ミドルウエアをはしらせる

端末を開いて、下記のように DAQ ミドルウエアを走らせる。

```
% run.sh -c
```

上記のようにすると、４つの端末(gatherer, dispatcher, logger, monitor)が開き、次のメッセージが出される。最後に下記のようなメッセージが出ると、成功といえる。

```
Current State:0
 Command  0:configure  1:start  2:stop  3:unconfigure  4:pause
5:resume
```

また、４つの端末のメッセージをよく見て、エラーが起こっていないか確かめておくのが良い。

入力はまず 0 で、configure を行う。次は 1 で start。これで ROOT の表示でインクリメンタルに２次元ヒストグラムができる。止めたいときは 2 を押して stop。

途中経過で、下記のようなメッセージがでる。

```
3036024 3036024 3036024 3036024
Current State:1
Command  0:configure  1:start  2:stop  3:unconfigure  4:pause
5:resume
```

上記の４つの数字は、それぞれのコンポーネントが取得した中性子の数で、途中経過は必ずしも同じ数字にはならない。

動作の確認では、configure-> start-> stop-> unconfigure-> start-> (pause-> resume-> を数回繰り返す) (stop-> start->を繰り返す) stop-> unconfiugre という手順で問題なく動いた。

それから、/home/Data/edata/NOVA000000_20090130 というディレクトリが作られ、データはそこにたまる。たとえば、NOVA000000_00_000_000.edb というファイル名となる。コンソールモードでやる場合は、run 番号の制御ができないので、一度は走らせたら必ず名前を変えるか、消去すること。つまり、start->stop を繰り返すたびに行うこと。

```
% rm –r /home/Data/edata/NOVA000000_20090130
```

HTTP モードでの動作確認

エミュレータを走らせる点はコンソールモードと同じ。端末を開いて、下記のように DAQ ミドルウエアを走らせる。今度は、-c は含まない。

```
% run.sh
```

上記のようにすると、４つの端末(gatherer, dispatcher, logger, monitor)が開き、次次のメッセージが出る。最後に下記のようなメッセージが出ると、成功といえる。

*** Ready to accept a command

WEB ブラウザを走らせる。

firefox は RedHat Enterprize5 にはプレインストールされていない。新しい firefox を使うことをお勧める。使った firefox のバージョンは 3.0.5。下記のようにインストールして、走らせた。

% su
# yum install firefox　あるいはすでに firefox をインストールしているのであれば
# yum update firefox
# exit
% firefox
指定した URL は、http://localhost/daq/operatorPanel/operatorPanel0.html

DAQ button には Configure Begin End Unconfigure Pause Restart のボタンがある。DAQ status には Get State ボタンがある。Get State ボタンはいつでも押せる。押さないとステータスは更新されない。Logger0 Monitor0 Dispatcher0 Gatherer0 の順に State 状態とイベントの数が表示される。

Run 番号を変更できる。テキスト欄に数値を入れればそれが番号となり、ボタンを押して更新される。さらに、Begin/End のたびに番号は１つ増える。従って、ディスクに書くデータのディレクトリは自動的に更新されるので、コンソールモードでのようにデータディレクトリを心配する必要はない。

Gatherer と Monitor の移植の詳細

検出器が変われば読み出しモジュールが変わり、Gatherer や Monitor の変更が伴う。従って、MLF 標準 DAQ ミドルウエア（現在は PSD 検出器をベースとしている）が提供されたとき GEM 用に変更するため、移植作業が必要となる。手順を確立しておくと、DAQ ミドルウエアがバージョンアップされたとき対応が容易になる。

Gatherer の移植

Gatherer の移植には大きく分けて２つの変更が伴う。１つはモジュールライブラリ。もう一つは Gatherer コンポーネント。

GEM モジュールライブラリ

モジュールライブラリは Sock ライブラリをベースに作られる。GEM の場合も Sock モジュールを使用している。

/home/daq/SiTCP/CPP/Sock

/home/daq/lib

/home/daq/lib はライブラリが作られたあと、インストールされるディレクトリである。Sock ディレクトリは Sock.h の参照に使われるが、libSock.so は lib ディレクトリに入る。GEM モジュールは下記のディレクトリに入る。

home/daq/SiTCP/CPP/Module

ファイルは下記の通り。

GemModule.h

GemModule.cpp

Makefile

GemModule はできるだけ PSD に合わせようと努力したが、固有の命名がふさわしいものもある。共通なものとしては

getSockFd()

init()

connect()

disconnect()

getModuleNumber()

setModuleNumber()

などがある。

Gatherer コンポーネント

イベントのフォーマットは現在 10 バイトだがいずれ 8 バイトになる予定なので、10 バイトのデータを受けて Gatherer 以降に流すデータは 8 バイトとした。

変更されたファイルは下記の通り。

- Selector.h
- Gatherer.h
- Gatherer.cpp
- Makefile.Gatherer

Selector.h

Selector.h は PSD 用のものと多少違っている。GEM 用は template を作って、GemModule クラスを入れられるようにした。もともとは Neunet クラスを直接書いている。どちらがよいかは別として、いずれにしても変更が必要。

Template <class T>　これを追加

Class Selector

それに伴い、

旧　int add(Neunet* module) {

新　int add(T module) {

旧　typedef std::vector< Neunet* > psd_module_list;

新　typedef std::vector< T > module_list;

下記のステートメントを削除する。

#include “PsdModule.h”

#include “Neunet.h”

上記のように、変更が必要になった。他にも修正箇所があるが、このような考えに基づいて変更が行われる。

Gatherer.h

まずは include ファイルの変更

旧　#include “Neunet.h”

新　#include “GemModule.h”

次に、Selector の違いによる変更がある。下記はその１つ。他にもあるので対応すること。

旧　int connect_modules(DAQMW::Selector& m_selector, …);

新　int connect_modules(DAQMW::Selector<DAQMW::GemModule*>& m_selector, …);

次は GEM モジュールの本体に関わるもので、いくつか追加を必要とする。

```
Void unpackMlfCoinMsg(
    DAQMW::GemModule::GemCoinMsgRO* gemCoinMsgRO,
    DAQMW::GemModule::GemMlfCoinMsg* gemMlfCoinMsg
);
```

また、下記のような定数が不必要となる。これらに関連するパラメータは GemModule クラスで定義してある。

旧　Static const int REQUEST_EVENT_NUM = 4*1024;

旧　Static const int NEUNET_BUF_BYTE_SIZE = 64*1024;

Gatherer.cpp

daq_start メソッドの変更

request to all NEUNET modules という部分は GEM と PSD の違いで、GEM には不要なもの。それらに関連する下記のコードが不要となる。

```
DAQMW::psd_module_list* myModuleList;
for(int i = 0; i<(int)m_dataSrcAddr.size(); i++) {
  myModuleList->at(i)->requestEvent(REQUEST_EVENT_NUM);
```

daq_resume メソッドの変更

daq_start と同じような理由で下記のコードを削除。

```
DAQMW:: psd_module_list* myModuleList = m_selector.getAllList();
for(int i = 0; i<(int)m_dataSrcAddr.size(); i++) {
  myModuleList->at(i)->requestEvent(REQUEST_EVENT_NUM);
```

daq_run の変更

この変更がメインである。データバッファは GemModule の中にあるので、そのポインターを取得するため、下記のようにする。

```
char* buffer = m_readList->at(i)-getBuf();
```

そして、GEM からのデータ読み出しのために、下記を実行する。

```
int status = m_readyList->at(i)->readCoinMsgU(buffer, &length);
```

正しく取れたら、convertRawData で、10bit->8bit のデータ変換を行い、同じバッファに詰めなおす。それが終われば、データにヘッダ・フッターを付けて OutPort にデータを転送する。

そもそも、Seletor を使った読み出しは、select システムコールで複数の検出器からデータの到着を待ち、先に到着した順で、任意の長さのデータを読み出すものである。

connect_modules/disconnect_modules の変更

Selector の違いと GemModule と Neunet のメソッドのが多少違い出てきているのでそのための変更が必要。

Makefile.Gatherer

Neunet モジュールは/home/daq/DaqComponents/src に展開したが、GemModule は /home/daq/SiTCP/CPP/Module に展開したので、その違いが出ている。

```
旧　SOCKINC = -I ../../SiTCP/CPP/Sock
新　SOCKINC = -I ../../SiTCP/CPP/Sock –I../../SiTCP/CPP/Module
旧　SOCKLIB = -L../../lib
新　SOCKLIB = -L../../lib –lSock –lGemModule
```

Monitor の移植

GEM 用 Monitor 移植の最大のポイントは ROOT を使うことである。MLF 中性子においては MANYO ライブラリと GNUPLOT が標準の解析ツールであるが、今回は大下さんが作ったオフライン解析用の ROOT プログラムを使って Monitor を使ったので基本は ROOT となった。MLF 中性子実験で使用する場合は標準解析ツールを使うのが自然だが、すでにプログラムがあったこと、MANYO ライブラリと GNUPLOT だけでは簡単に移植できないこと、という理由で今回は ROOT を使う。

Monitor コンポーネント

Monitor コンポーネントの移植には下記のファイルの変更が必要。

- ✧ Monitor.h
- ✧ Monitor.cpp
- ✧ MonitorComp.cpp
- ✧ Makefile.Monitor

Monitor.h

ROOT のための include 文がいくつかある。MANYO ライブラリーや GNUPLOT 関連の include 文は外す。それから GemModule もモジュールが持つパラメータを使用するため必要。

下記の定数は最大のバッファのサイズと最大のイベント数を決めている。

```
static const int MAX_BUF_SIZE = DAQMW::GemModule::MAX_BUF;
static const int MAX_EVENT_NUM = MAX_BUF /
DAQMW::GemModule::EVENT_BYTE_SIZE
```

下記の２つのメソッドは前者が１つの（構造化された）イベントデータから位置情報などを引き出しているのに対して、後者は（生の）イベントデータからそれらの情報を引き出す。後者は前者を使っている。

```
void extractMlfCoinMsg(DAQMW::GemModule::GemMlfCoinMsg*
gemMlfCoinMsg, unsigned char* sig, unsigned char* id, unsigned char* x,
unsigned char* y, unsigned int* time);
bool decode(unsigned char* event, unsigned int* module_id, unsigned
int* gem_id, unsigned int* time, unsigned char* x, unsigned char* y);
```

下記のメソッドはヒストグラム等の初期化を行う。

```
void ROOTSet();
```

ROOT に関連する重要なメンバーで、前者が canvas 後者が２次元ヒストグラム。

```
TCanvas* m_canvas;
TH2F* m_histo;
```

Monitor.cpp

コンストラクタの変更

コンストラクタの初期化で、canvas と histo の初期化を行う。

```
Monitor::Monitor(RTC::Manager* manager)
 : DAQMW::MlfComponent(manager),
新規追加　m_canvas(0), m_histo(0)
```

daq_configure メソッドの変更

PSD 関連を削る

daq_unconfigure メソッドの変更

PSD 関連を削る

daq_start メソッドの変更

canvas の生成と histogram の生成を行う。

daq_stop メソッドの変更

PSD 関連を削り、ROOT 関連を追加。

daq_run メソッドの変更

PSD 関連を削り、ROOT 関連を追加。

MonitorComp.cpp

次の２つのコメントをはずす。

```
TROOT root(“GUI”, “GUI”);
TApplication theApp(“App”, &argc, argv);
```

Makefile.Monitor

下記の変更を行う。

```
ROOTLIBS = `$(ROOTSYS)/bin/root-config –glibs`
ROOTINC = -I ‘$(ROOTSYS)/bin/root-config –incdir`
SOCKINC = -I ../../SiTCP/CPP/Sock –I ../../SiTCP/CPP/Module
CXXFLAGS = `rtm-config –cflags` $(MANYO_INC) –Wall –g
$(ROOTINC) $(SOCKINC)
```

参考文献

１．千代浩司、仲吉一男、安　芳次(KEK 素核研)
MLF 中性子用 DAQ ミドルウェアインストールおよび操作マニュアル
http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/tars.2008.12/DAQMM.pdf
２．仲吉一男、安　芳次、千代浩司(KEK 素核研)、DAQ ミドルウエアの概要
３．千代浩司(KEK 素核研)、SiTCP-PSD 用ユーティリティープログラムマニュアル
第 1.1 版、
http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/tars/SiTCP-PSD-Utils/
SiTCP-PSD-Utils.pdf
４．千代浩司(KEK 素核研)、VMWarePlayer のインストール
http://greentea.kek.jp/sendai/
５．仲吉一男、千代浩司 (KEK 素核研)、
OpenRTM および DAQ ミドルウエアパッケージ
http://www-jlc.kek.jp/~sendai/OpenRTM/EL5/
６．大学・研究所一般用 DAQ ミドルウエアパッケージ
http://www-online.kek.jp/~nakayosi/DAQMW/sl52.zip
７．仲吉一男(KEK 素核研)、DAQ Component Developer's Guide