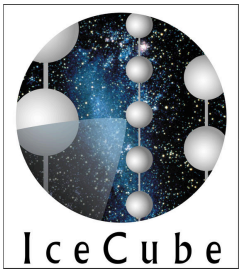


10 inch IceCube PMT and DOM Measurement Manual

ver 1.2

2010年3月5日

目次

PMTとDOMの測定について

PMTとDOMの測定はいくつかに分けられる。

- PMTとDOMで共通な測定
    (1) 絶対較正：absolute calibration
    (2) ２次元感度スキャン：2D scan
- PMTのみにある測定
    (1) 電荷応答：charge response
    (2) ノイズ：Dark Count
    (3) アフターパルス：afterpulse
    (4) SPEパルス測定：SPE waveform
- DOMのみにある測定
    ▸なし

これらの測定の方法を記述する。前半にはPMTの測定について、後半にはDOMの測定について述べる。

使用されているソースコードとスクリプトについて

測定で使用されているソースコードとスクリプトはgorappaの以下にまとめてある。

/disk0/data/IceCube/DataBranch/src
また、オリジナルのソースコードとスクリプトは、ppldaq1についてはホームディレクトリのFreezer、PmtScanのディレクトリの中にある。ppldaq3についてはホームディレクトリより、DAQ-CHIBAのディレクトリの下にある。

10 inch PMT

PMTは 1 光子レベルを判別できるほど感度がある。当然、部屋の光はPMTにとってさけなければいけない。High Voltage(HV)をPMTにかけるとき、PMTのシグナルをオシロスコープでチェックしながら行うことが望ましい。さらに、HVをかけている最中はFreezer Boxなどの光を遮蔽しているものは絶対あけてはならない。

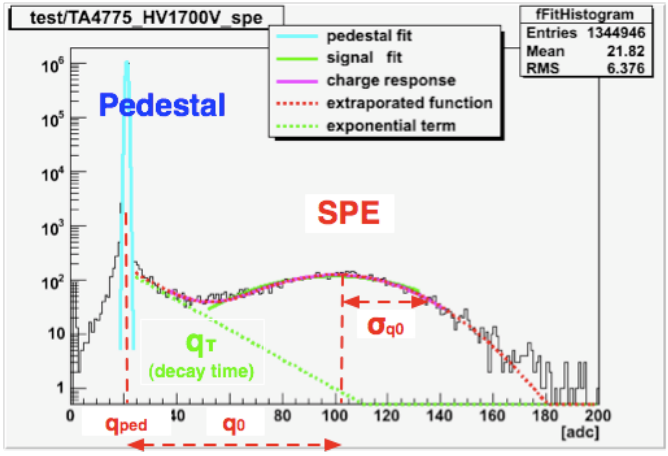
電荷応答：charge response

ここではSingle Photo-Electron(SPE)を利用して、電荷応答(charge response)を測定する。PMTにかける電圧を高くするとゲインは増倍するので、ゲインのリニアリティをチェックする。電荷応答は、ゲインにあまりよらないので、これもチェックする。

Charge Response

電荷応答について説明する。下の図は、X軸はSPEが増倍された後の電荷(単位はADC)であり、Y軸はイベント数である。ペデスタルはSPEが発生しなかった場合に相当し、ベースラインのようなものである。PMTの性質を考えればSPEはガウス分布を持つが、見ても分かる通り指数関数(緑)も現れている。これはこの型のPMTの重要な性質である。

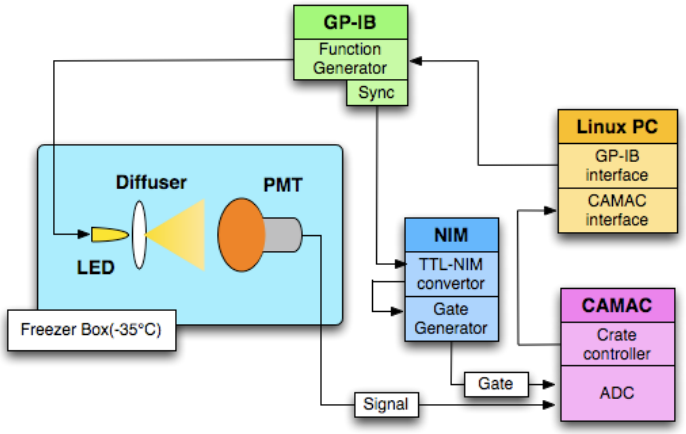
理論的には、次の式で表される。



$$f(q) = \frac{P\_ex}{q_\tau} Exp\left(-\frac{q - q_{ped}}{q_\tau}\right) + (1 - P\_ex)\frac{1}{\sqrt{2\pi}\sigma_{q_0} N} Exp\left(-\frac{(q - (q_0 + q_{ped}))^2}{2\sigma_{q_0}^2}\right)$$

- q0 : peak charge of the gaussian
- σq0 : charge resolution
- qτ : decay constant of the exponential term.
- P_ex : fraction of the exponential term contribution to the response function

Measurement Setup



図のように回路を組み立てる。Gateは100ns、Function Generatorは2kHz。PMTをセットしてから少なくとも半日は待つ。

データ取得準備

まず、データを置く場所を作る。

$ cd ~/Freezer/Data/ADCTaking
$ mkdir 2009_05 && cd 2009_05
2009_05のところはPMT測定時の西暦、月を代入する。

$mkdir TA1234 && cd TA1234
PMTのシリアルナンバーを代入する。

次に、パルサーを使い、LEDを操る。

$ PulseShot [LED Voltage(V)]

ここで、ADCにさす二本のケーブルをオシロスコープに差し込む。SPE信号がゲート信号に入るようディレイを調節する。SPE信号が一秒に20個程度になるようにLEDの電圧を調節する(1.5V~2V)。そして、二本のケーブルをADCに差し込む。

始めはペデスタルを測定するため、LEDが光らないようにする。

$ PulseShot 0.1

[LED Voltage(V)]

0.1 VでLEDは光らない。この状態で、

$ Auto_TakeADC TA1234_HV1500V_ped.data 0 100000

[file name] [threshold] [event number]

これで、ペデスタルのデータが取得できる。これをless見るかgnuplotなどを使い、ペデスタルに異常がないか確認する。一般的に、ペデスタル幅はピークから±5 ADC程度である。

データの取得と解析

先ほど調節して求めたSPE信号がとれるLED電圧に変え、データを取る。

$ PulseShot [LED Voltage]

$ Auto_TakeADC TA1234_HV1500V_spe.data [pedestal peak + 5] 10000

ただし、thresholdは、ペデスタルのピークから+5ADCくらいに設定する。データが取り終わったら、

$ root -l

root[0] .L ~/Freezer/Analysis/ADCTaking/ChargeResponseAnalysis.C

root[1] ChargeResponseAnalysis("TA1234_HV1500V_spe.data",0,kFALSE,[threshold +2]);

[file name] [mode] [fix qtau] [threshold]

引数については、測定結果をチェックするのでmodeは０でよい。ちなみに、解析に使用する１は0.15 p.e threshold、２は0.2 p.e threshold、３は0.25 p.e thresholdであるが、ここでは使わない。fix qtauはqtauを0.2に固定するかどうかであり、ここではkFALSEとする。フィットする範囲としてはthresholdより右側を使う。だいたいSPEのデータをとるときに設定したthreshold + 2くらいが適当である。

このマクロは、次の順序でフィットを行い、ゲイン、電荷応答の各パラメータを得ている。

- ペデスタルをフィット(0~threshold)
- ガウシアンでフィット(threshold~120)
- もう一度ガウシアンでフィット(q0-sq0 ~ q0 + sq0)
- charge response functionでフィット(threshold ~ q0 + 2×sq0)
- もう一度charge response functionでフィット

さて、これでゲインと電荷応答の各パラメータが分かった。他のゲインも測るのだが、5V単位でHVを変えて次のことを実行し、データシートに記録する。

- ５×10^7のゲインがでるHVを探す。
- 前のHVの+ 100, + 200, -100, -200Vでゲインを測定する。
- 1×10^7のゲインがでるHVを探す。

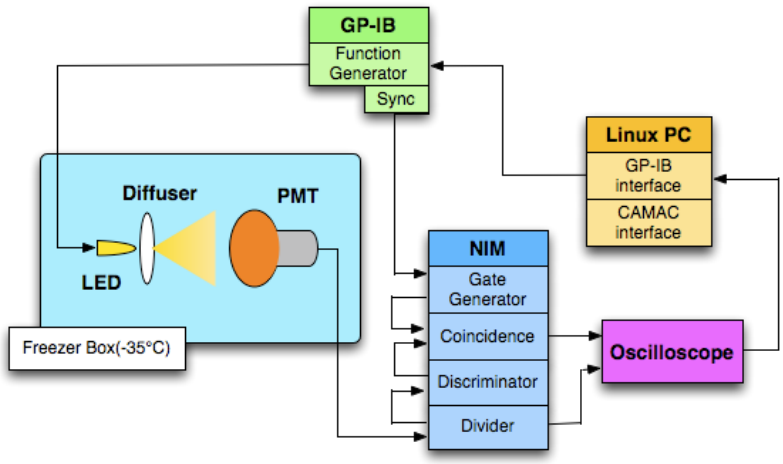
さらに解析するためのスクリプト、解析されたデータなどは以下にまとめておく。

/disk0/users/hasegawa/icecube/PMT/freezer/charge_response

SPEパルス測定：SPE waveform

オシロスコープを使い、SPEの波形データを取る。これにより、SPE波形の広がりをチェックする。

Measurement setup



PMTからの信号はだいたい40mV程度。Discri Levelをだいたい17mVにし、オシロスコープのレンジは、SPE全体がとれるようにレンジを切り替える。

データ取得準備

以下のコマンドをじっこうする。

$ cd ~/Freezer/Data/WaveForm/

$ mkdir 2009_05 && cd 2009_05

$mkdir TA1234 && cd TA1234

$ WaveForm_Taking TA1234_HV1500V 10 50

[file name] [Voltage(mV)] [Time(ns)]

VoltageとTimeは、オシロスコープを見て決める。１０００イベント取るが、だいたい１時間かかる。

次に、データを解析する。

$ root -l
root[0] .L ~/Freezer/Analysis/WaveForm/A1000evt.C
root[1]A1000evt(“TA1234_HV1500V_Waveform”,1500);
次に、

$ root -l

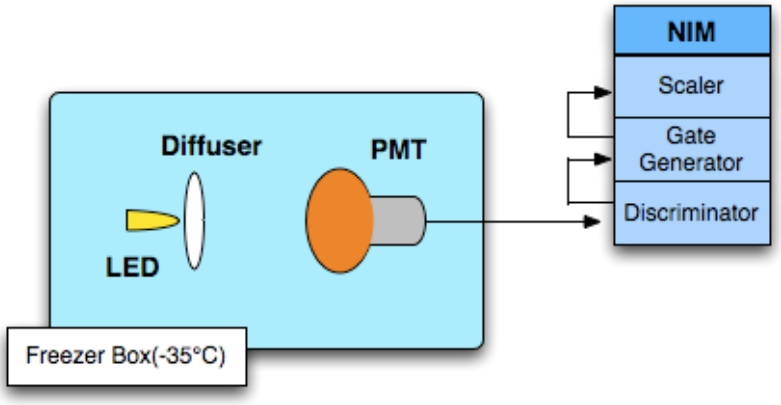
root[0] .L ~/Freezer/Analysis/WaveForm/stage1_analysis.C

root[1] stage1_analysis(“TA1234_HV1500V_Waveform”);

これで、Chisquareなどをチェックする。

Dark Count

Measurement Setup



データ取得

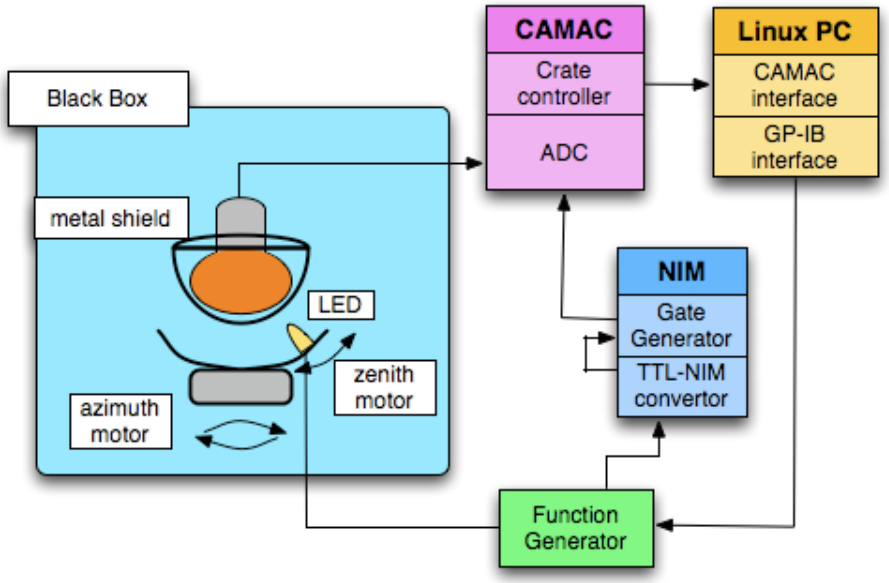
ストップウォッチを使用し、スケーラで10000カウントになるまでの時間を測定し、データシートに記録する。Gate Generatorからの信号は、幅を変えて何度も測定する。

2-Dimensional Scan

２軸モーターを用い、PMT光電面の感度を相対的にマッピングする。

Measurement Setup

配線は以下の図のようになる。



回路は、上図のように組み立てる。

PMTは、ダイノードが実験室のブラックボックスにおける壁側に向くようにセットする。PMTの円柱部分を、黒い板と固定用ねじで固定する。その黒い板をBlack Boxに取り付ける。固定する支柱とナットは４組あるが、一つは固定してあり動かさない。残る３組は、黒い板が平行になるように水準器を使い調節する。

確かめておく必要があるのは、LEDがきちんと最後まで、コード類が絡まずにモーターを動かせるかということと、LEDがシールドにひっかからないかということである。

$ PMTScan_ver5_bin test.data

> step_zenith 2500

> step_azimuth 10000

などとしてLEDを動かす。azimuthの原点は、ブラックボックスにおける壁側である。もし、ずれていたら

> azimuth_reset

でカウントをリセットする。測定では、azimuthの最大値は1000000で、一周に該当する。2D scanのコマンド一覧は、Appendixにまとめてある。

これらのチェックが終わったら、Black Boxをとじ３時間以上待つ。

データ取得準備

まずは、光漏れがないかチェックする。PMTのアウトプットをオシロで見ながらHVを5×10^7のゲインが出るまで上げる。途中でノイズが1MHzを超えていたら光漏れを起こしているのできちんと遮光する。

HVを上げても数kHzであればOK。次にペデスタルを取る。

$ PmtScan_ver5_bin test.data

> volt 0.1　(LED電圧を設定)

> pulse_shot (LEDをうつ)

> adc

threshold -> 0

number of event -> 100000

> exit

このtest.dataファイルをチェックする。

$ MkPertition test 0 1032 1032 .data

[file name] [min ADC] [max ADC] [pert num] [suffix]

そして、gnuplotで確認する。ペデスタルが広がっていなければOKだ。

次に、LEDの適切な電圧を見つける。

$ PmtScan_ver5_bin sig_test.data
> volt 1.6
> pulse_shot
> adc
threshold -> 0
number of event -> 100000
> exit

そして、取得されたデータの平均値が600~800ADCになるような電圧を次の測定で利用する。

千葉大学

データ取得

データを取得する。

$ PmtScan_ver6 TA1234_HV1500V.data

-> volt 1.6 (先ほどのLED電圧)

-> pulse_shot

Function GeneratorからLEDケーブルをぬく。

-> adc

LEDケーブルをつけ、座標が Axis[0] = 0 Axis[1] = 0であることを確認し、

> auto_scan

count of zenith -> 2500

count of azimuth -> 31250

-> take_adc

この後、８時間ほど自動的に測定される。この間は部屋の明かりをあまりつけないことが望ましい。

測定が終わったら、データがきちんととれているかどうか確認する。

$ gzip TA1234_HV1500V.data

$ zcat TA1234_HV1500V.data.gz | plotADCdist1D 0 1200 0 32 > TA1234_HV1500V.gks

$ grafig TA1234_HV1500V.gks

これにより、一番始めのazimuth軸と一番最後のazimuth軸を重ねて表示できる。横軸がzenith座標、縦軸がADC値である。ADC値の絶対値は違うが、形は同じになっていれば問題なくデータは取得できている。

テーブルを作るためには、grappaにおいて以下のコマンドをうつ。

$ /misc/disk0/data/IceCube/DataBranch/src/PmtScan/analysis/uniformity/makeADCBSoff [file name] [min adc] [mac adc] > [output file name(.table)]

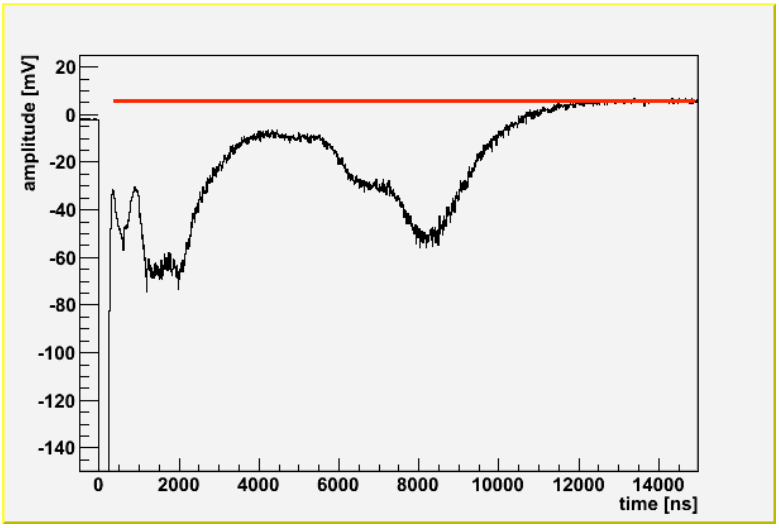
テーブルが正しく読めるかどうかは、ROMEOを使い確かめる。テーブルファイルをromeo/resources/data/に移動する。次にromeo/stand_alone/analysisに移動し、以下のコマンドをうつ。

$ root -l
root[0] .L RMOPMT2DCE.C
root[1] RMOPMT2DCE(“table_name”);

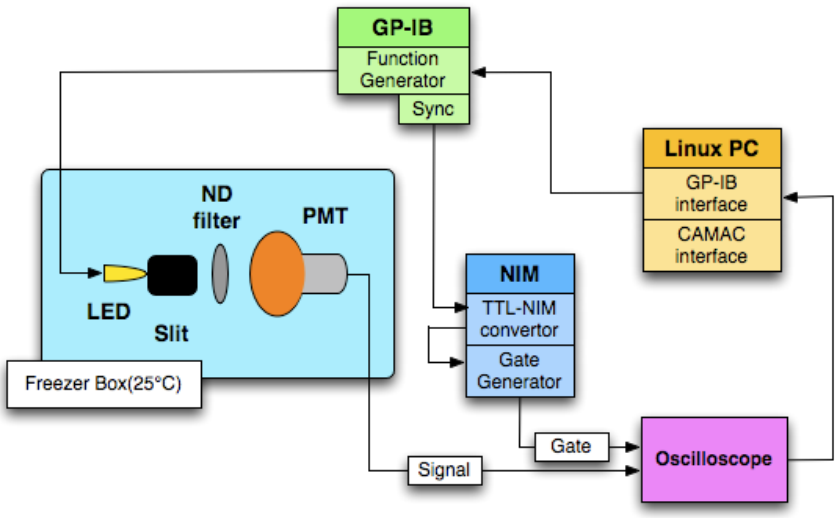
Afterpulse Measurement

10 inch PMTに独特な現象であるアフターパルスを測定する。これは主にPMT内の残留ガスが、光電効果によって発生した電子と衝突し、イオン化することに起因する。オシロスコープを使い時間分布を測定する。

時間分布は次の図のようになる。



Measurement Setup



ND filter Calibration

まずは、PMTのゲインを2×10^7程度する（測定方法は前章を参照）。６枚のNDフィルターの測定の順番は特に決まりはない。CAMACを使い、主に隣り合ったフィルターで相対的な透過率を出す。推奨する組み合わせとしては、

- 100%と50%をMPEで測定する。
- 50%と10%をMPEで測定する。
- 100%をMPEで、0.1%をSPEで測定する。
- 100%をMPEで、1%をSPEで測定する。
- 1%と5%をMPEで測定する。

それぞれの比率を出したら、100%を基準にそれぞれの絶対透過率を出す。CAMACでデータを取得できる範囲では、サチレーションは影響しない。

具体的な測定の方法は、

$ AutoTake2.sh [output file name(no suffix)] [LED voltage in mV] 1
でデータを取得し、異なるND filterでADCを比較し透過率を出す。

データ取得

ゲートとシグナルをオシロスコープに突っ込む。

まずは0.1% ND filterにしてデータをとる。

$ mkdir TA1234 && cd TA1234

$ mkdir Npe afterpulse gain

Npeは0.1%で取ったデータを、afterpulseは10%、50%、100%で取ったデータを、gainはゲインの測定データを格納する。gainの測定方法は電荷応答(charge response)を参照。

$ cd Npe

$ PulseShot 1.91

ここで、オシロスコープを10 mV、50 nsのレンジにして、数十NpeになるようにLEDの電圧を調整する。

$ WaveForm_TakingHase100evt TA1234_HV1500V 10 50

次に、LEDの電圧はそのままで10%のNDフィルターに替える。オシロスコープのレンジは500mV、2500nsに設定する。

$ PulseShotHase 1.91

$ WaveForm_TakingHase1000evt_10per TA1234_HV1500 500 2500 2000

最後の引数を変えて、メインパルスがオシロスコープの左端から一目盛りくらいにくるように調節する。

次に50%、100%とNDフィルターを変えて測定する。

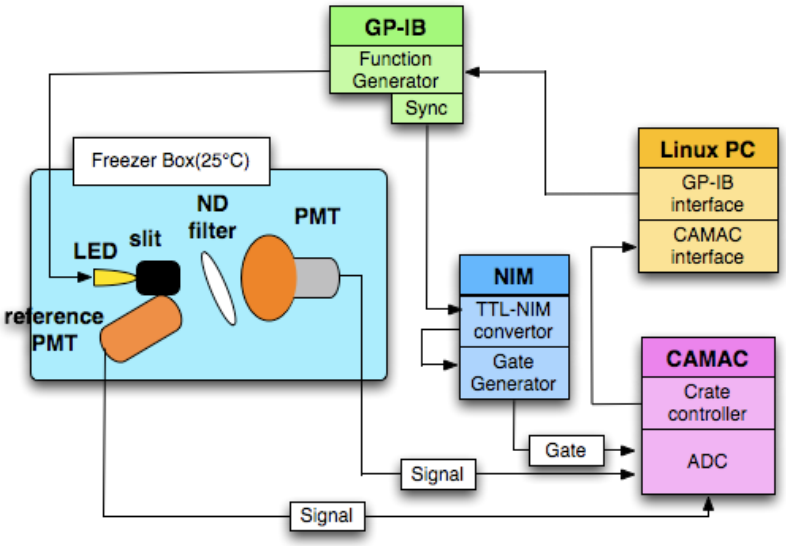
これまでの手順を、LEDの電圧を変えてもう一度繰り返す。

取得されたデータの解析方法としては、平均波形を出しアフターパルスの量を見積もる。解析用マクロや解析方法などはgrappaにまとめてある。

/disk0/data/IceCube/DataBranch/src/Freezer/analysis/waveform_taking/

Absolute Calibration

Measurement Setup



回路は上図のようになる。この測定では、QEの分かっている2 inch PMT(reference PMT)を使用する。また、ND filterの透過率と反射率を測定するときには、10 inch PMTとしてQEの分かっているものを置く。ND filterから各PMTまでの距離は約11cmである。CAMACでは、二つのPMTからの信号を同時に取得する。LEDはXステージにより移動可能である。

この測定方法では、LEDの位置やND filterの傾きに敏感である。ND filterとreference PMT、LEDの座標は、始めに決めたら固定する。まず、420nm LEDからND filterの中心を通り、10 inch PMTの中心に光が当たっていることを確認する。次に、ND filterから反射した光がreference PMTの中央にあたっているかを確認する。この操作を他の波長のLEDにも適用し、そのときのXステージの座標を記録する。この座標は固定である。

System Calibration

始めに、50%ND filterの透過率と反射率の比を求める。求めたいものを式にすると以下のようになる。

$$\frac{T}{R} = \frac{N_{pe}^{ic3}}{QE^{ic3}\bar{q}} \frac{QE^{ref}}{N_{pe}^{ref}}$$

方法としては、Freezerを閉めて一時間ほどしたらHVを上げ、gainを測定する。reference PMTについては、別途waveformによりgainを出す。次に、反射率と透過率の比を測定する。LEDの電圧は、10 inch PMTにおいてADC値が平均800程度になるようにする。

$ AutoTake2.sh [file name(no saffix)] [LED voltage in mV] 1
データが取得できたら、透過率と反射率の比を求める。

$ root -l

root[0] .L NDMeasure.C

root[1] NDMeasure("[file name(no suffix)]",[LED lambda[nm],[LED voltage],[10inch PMT charge response mean],[reference PMT gain],[10 inch PMT gain])
これにより、透過率と反射率の比(T/R)が出てくる。この値を全てのLEDにおいて求め記録する。ただし、それぞれのPMTのQEをNDMeasure.Cの中に記述する必要がある。

Data Taking and Analysis

反射率と透過率の比(T/R)を求めたら、10 inch PMTを置く場所にQEを知りたいPMTを置きgainを測定する。次に、T/Rを求めた時と同じように、LEDの光量を調節する。そして、reference PMTと10 inch PMTでデータを取得する。取得方法はT/Rを求めた時と同じAutoTake2.shを使用する。

　データ取得の後は、QEを計算する。

$ root -l

root[0] .L PMTQE_ver8.C

root[1] PMTQE_ver8("[file name(no suffix)]",[LED lambda[nm]],[LED voltage],[charge response mean],[ref gain],[ic3 gain])

これでQEが計算されるので、記録する。ここで求めるQEを図式化すると、

$$QE = \frac{N_{pe}^{ic3} QE^{ref}}{N_{pe}^{ref}} \frac{R}{T}$$

のようになる。二つの式より、reference PMTのQEはキャンセルすることになり、最終的にはT/Rを測定する時の10 inch PMTのQEだけ事前に知っておく必要がある。

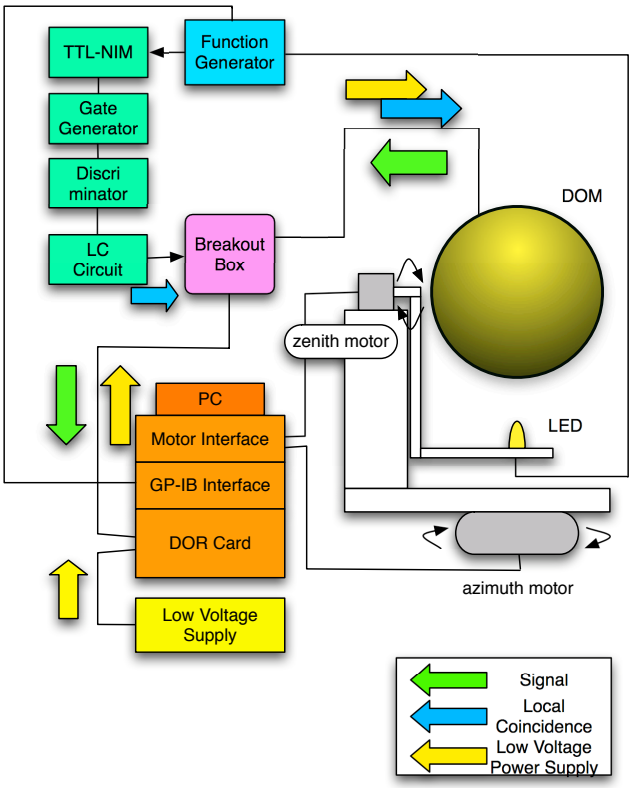
DOM(Digital Optical Module)

2-Dimensional Scan

PMTの時のように、DOMの感面全体を相対的に求める。

Measurement Setup

セットアップは以下の図のようにする。Low Voltageは100Vにする。DOMの電源が入っていない状態では、電流は0である。



ここで、DOMの設置の仕方について述べる。DOMの向きは下である。上側の３本のワイヤーはアルミの金具で束ねる。下側の３本のワイヤーは上側に持ってきてひもなどでとめておく。上側の３本の先にあるわっかを、Black Boxの天井についているフックにかける。DOMの高さはフックについているねじにより調節する。DOMの高さは正確にセットしたいので、LEDを90°の位置に動かしLEDの光を当てて確かめる。

$ DOMscan_noGPIB test

>

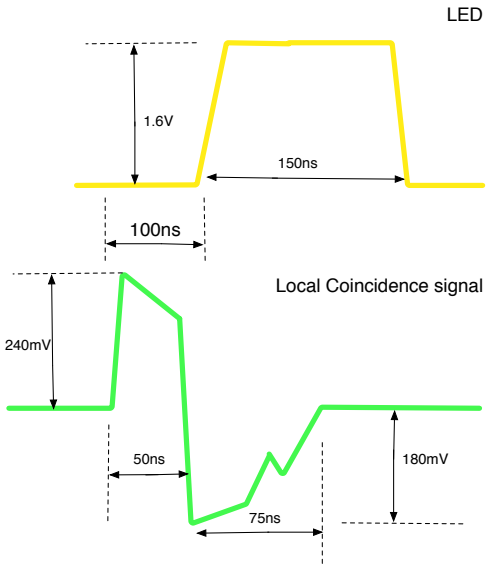
このようなプロンプトになる。zenithとazimuthのモーターの動かし方はPMTの時と同じである。zenithは2500カウントで90°となる。zenithについては、0°の時に水準器を使用して調節しておく。azimuthの原点は、DOMのメインボードに取り付けられている金色のLEDである。

LEDを光らせるには、

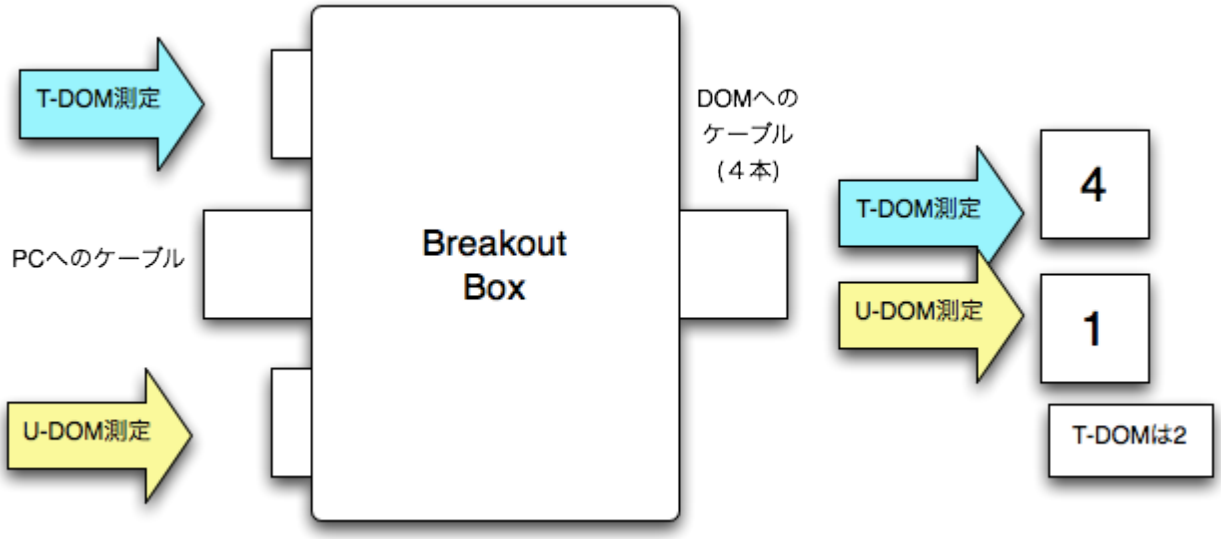
$ PulseShot [voltage]

とうつ。

LCサーキット部分については、オシロを使用し以下の図のようになることを確認する。



この図のように、LEDインプットのパルスと100ns程度の違いにする。LCサーキットからBreakout Boxにつなげるコネクタは T-DOMとU-DOMで違うので注意する。以下の図は、Breakout Boxへの配線の図である。



DOMのケーブルは、T-DOMの場合には４番、U-DOMの場合には１番のコネクタと接続する。U-DOMについては、別途T-DOMも２番につなげておく必要がある。このU-DOMとペアになるT-DOMは、Black Boxの外に設置する。T-DOMからのケーブルは、Black Boxの底面にある穴より通す。次に、azimuthを動かしてコードのからまりやDOMケーブルのからまりをチェックする。

Low Voltage Supplyは、電圧を100Vに設定し、電流を流した時(output on)に電流は0Aであることを確認する(DOM電源off)。

DOMのゲイン測定

DOMのゲインを測定するが、ゲインはDOM内部で計算する。よって、以下のコマンドを打つだけで良い。

$ on all
をしてDOMを起動する。

(T-DOMの場合)

0 0 A - not communicate　###ケーブル番号は2

0 0 B - not communicate　###ケーブル番号は1

0 1 A - communicating　###ケーブル番号は4

0 1 B - not communicate　###ケーブル番号は3

(U-DOMの場合)

0 0 A - communicating　### T-DOM
0 0 B - communicating　### U-DOM
0 1 A - not communicate
0 1 B - not communicate

次に、DOMをポートに割り当てる。

$ gotoiceboot
$ dtsxall
次に、DOMのIDなどをチェックする。

$ test.py 0 1 A (communicatingと出たもの)

遮光がきちんとできているか確認するには、

$ rate.py -v 600 -l 500 -h 600 -s 20 0 1 A
ここで、-vはHV、-lと-hはディスクリレベルの最大値と最小値、-sはそのステップである。

これを実行してしばらくすると

Setting HV to 600V for rate measurement
disc = 500 average rate = 4 Hz
disc = 520 average rate = 6 Hz
disc = 540 average rate = 136720 Hz
disc = 560 average rate = 7662 Hz
disc = 600 average rate = 6789 Hz
のようなメッセージが現れる。540(DAC値)のところにピークがたっているが、ベースラインなのでモンヂアはない、それ以降のレートがノイズとなる。常温においては数kHzが正常である。もし、数十kHz以上ならば光漏れをしているので、遮光を再度確認する。

　次のステップとして、DOMのファームウェアを、実験室で測定できるように書き換える。

$ stoptestdaq (全ての測定を中止し、DOMの電源をoff)

$ ldall /home/testdaq/release/release322/release.hex

communicating doms : 01A
01A:configboot v2.71

このコマンドを撃った後10分程度待つ。すると大量のメッセージが出てくるが、書き換えが完了する。

$ echo “0” > /proc/driver/domhub/blocking
次にDOMのゲインを測るためにDOMCalを起動する。

$ cd ~/domcal/
$ stoptestdaq
$ on all
$ gotoiceboot
$ dtsxall
$ domcal-wrapper
そして40分ほど待つ。

プロンプトが返ってきたら、

$ cp latest-domcal-[year]-[month]-[day]/*.xml .　###domcalが行われた年月日がディレクトリになる。

$ java icecube.daq.domcal.HV2DB /mnt/data/testdaq/domcal
とうつ。これでゲインが測定されデータベースに登録された。

ゲインが正しく測定されたかを確かめるには、以下のコマンドを打ち、pngファイルとhtmlファイルを作る。

$ cd latest-domcal-[year]-[month]-[day]/
$ java icecube.daq.domcal.HVHistogramGrapher [xml file directiory] [output dir for png] [output dir for html]
現在のディレクトリであれば、引数は全て現在のディレクトリ(.) で問題はない。

生成されたpngファイルをdisplayなどでチェックする。*****.xml_hv.pngのファイルは、HVとゲインの関係図であり、これを特にチェックする。

測定準備

始めに、steering fileと呼ばれる実行ファイルを作成する。適当なディレクトリを作り、そのディレクトリの中で

$ stoptestdaq
$ on all && gotoiceboot && dtsxall
$ autogen-wrapper

とうつと、幾つかのファイルが生成される。その中で、LC-****.xmlのファイルが二つあるので、そのうちのATWD0を使う。ファイル名をLC-ATWD0-[DOM name].xmlのように、分かりやすい名前に変える。次にこのファイルを編集する。

(T-DOMの場合)

<途中省略>

<param name='NUM_SAMPLES' atwdChannel='2' value='128'/>
<param name='SAMPLE_SIZE' atwdChannel='2' value='2'/>
<param name='NUM_SAMPLES' atwdChannel='3' value='128'/> ###0を128にする。LCシグナルを見る。
<param name='SAMPLE_SIZE' atwdChannel='3' value='2'/>
<param name='NUM_FADC_SAMPLES' value='128'/> ### 0を128にする。FADCを見る。
<param name='TRIG_MODE' value='2'/>
<param name='ATWD_SELECT' value='0'/>
<param name='ANALOG_MUX_SELECT' value='5'/> ###この行を追加する。値は5。
<param name='DAC_LED_BRIGHTNESS' value='0'/>
<途中省略>
<param name='LOCAL_COIN_MODE' value='3'/> ###値を3にする。
<param name='LOCAL_COIN_WIN_UP_PRE' value='0'/> ###値を0にする。
<param name='LOCAL_COIN_WIN_UP_POST' value='0'/> ###値を0にする。
<param name='LOCAL_COIN_WIN_DOWN_PRE' value='400'/> ###値を入れる。
<param name='LOCAL_COIN_WIN_DOWN_POST' value='400'/> ###値を入れる。
<途中省略>

LOCAL_COIN_WIN_DOWN_PREとLOCAL_COIN_WIN_DOWN_POSTは、DOMがキャプチャーしたシグナルと同期したLCシグナルを見つけるウィンドウの幅(ns)である。LCシグナルとLEDシグナルの差が100ns程度であれば、400nsで問題はない。

(U-DOMの場合)

注意しなければならないのは、T-DOMについてのsteering fileもふくまれていることである。T-DOMの部分については、PMT_DACを0に設定する必要がある。以下はU-DOMについてである。

<途中省略>

<param name='NUM_SAMPLES' atwdChannel='2' value='128'/>
<param name='SAMPLE_SIZE' atwdChannel='2' value='2'/>
<param name='NUM_SAMPLES' atwdChannel='3' value='128'/>　###0を128にする。LCシグナルを見る。
<param name='SAMPLE_SIZE' atwdChannel='3' value='2'/>
<param name='NUM_FADC_SAMPLES' value='128'/> ### 0を128にする。FADCを見る。
<param name='TRIG_MODE' value='2'/>
<param name='ATWD_SELECT' value='0'/>
<param name='ANALOG_MUX_SELECT' value='4'/>　　###この行を追加する。値は4。
<param name='DAC_LED_BRIGHTNESS' value='0'/>
<途中省略>
<param name='LOCAL_COIN_MODE' value='2'/>　　###値を2にする。
<param name='LOCAL_COIN_WIN_UP_PRE' value='400'/>　　###値を入れる。
<param name='LOCAL_COIN_WIN_UP_POST' value='400'/>　　###値を入れる。
<param name='LOCAL_COIN_WIN_DOWN_PRE' value='0'/>　###値を0にする。
<param name='LOCAL_COIN_WIN_DOWN_POST' value='0'/>　###値を0にする。
<途中省略>

次にシグナルとLCが同期しているかを確かめる。編集したsteering fileを~/chiba-config/test-runs/にコピーし、~/bin/automateを編集しsteering_dirを~/chiba-config/test-runsにする。

$ PulseShot [LED voltage(typical 1.6)]
$ stoptestdaq
$ on all
$ gotoiceboot
$ dtsxall
$ ready
$ go [comment]

とうつと測定が始まる。1分ほどで終わると思う。コメントは測定されたデータのディレクトリのなかに*****.commentファイルに格納されている。

データの波形を見るために

$ ezplot

とうつ。するとGUIが起動するので、File→LoadHitFileを選択する。latest-dataディレクトリにある。Run番号が最も後にあるものを選択し、CHIBA-***.hitファイルを選択する。Control→Startでとれたデータの波形が見れる。うまく取得できていれば常に波形が出ている。もし、取得できていなければ、WINDOW時間を変えて再度たしかめる。

次に、光量がどれくらいかをNpeで見積もる。~/DAQ-CHIBA/DOMscan/analysisに移動し、

$ mkdir dump_features/datFiles/2Dscan/[DOM name]

$ sh dump_features.sh [DOM name] 2Dscan [run number]

run numberは、~/latest_dataにある測定されたデータファイル名の中に記述されている。

つぎに、rootを使いNpeを計算する。

$ root -l

root[0] .L DOMhist_2DCE.C

root[1] DOMhist_2DCE(“dump_features/datFiles/2Dscan/[DOM name]/CHIBA-PPLDAQ3_run00[run number]”, [ATWD channel],0);

出てきたヒストグラムをチェックする。ガウス分布であればOK。[ATWD channel]は、見たいATWDチャンネルを指定するが、ezplotでsaturationを起こしていないチャンネルを見るべきである。結果の1Dヒストグラムは横軸がNpe数である。

データ取得

実際にデータをとる。~/bin/automateを編集し、steering_dirを~/special_steeing_files/chiba/2Dscanにする。先ほどのsteering fileを~/special_steeing_files/chiba/2Dscan/2DscanStFile/にコピーする。コピーの名前はLC-[DOM name]-ATWD0-LED1600Hz.xmlとする。

準備が完了したら、

$ DOMscan_ver2 [LED voltage] [DOM name]

とうつと、DOM scanが開始される。データ取得時間は約6時間~8時間程度である。２番目に取得したデータをezplotを使用して確かめる。問題があった場合にはstoptestdaqのコマンドを実行する。

データ解析

データを解析するにはicetrayのhttp://code.icecube.wisc.edu/svn/sandbox/hase/gdom-analysis/trunk/のプロジェクトを使用する。

使うスクリプトはresorces/scripts/gdom-chiba_portia_ATWD0.pyである。

$ python resorces/scripts/gdom-chiba_portia_ATWD0.py [data dir] [data file name] [comment file name] [run number]

これを全てのデータに対して行い、

$ cd resorces/analaysis

$ root -l

root[0] .L DOMScanAnalysis.C

root[1] DOMScanAnalysis(“[DOM name]”,[first run number],[last run number])

これにより、テーブルが生成される。

チェックのためには、ROMEOのDOM2DCE.Cを使用する。基本的な使い方はPMT2DCE.Cと同じである。

DOM Absolute Calibration

DOMについてのabsolute calibrationの方法は、PMTのときとほぼセットアップが同じである。データの取得方法などは、DOM scanと同じであるので、ここでは詳しくは記述しない。

Appendix

主要なPmtScan、DOMscanにおけるモーター操作のコマンド

コマンド名	説明
count	現在の座標をプリントする。zenithはAxis[0]、azimuthはAxis[1]。
step_az [count]	azimuth方向のモーターを動かす。一周は100000カウントである。
step_ze [count]	zenith方向のモーターを動かす。カウントと移動距離の関係はそれぞれの測定を参照。
az_reset	azimuthカウントをリセットする。azimuthの原点を変えたい時に使用する。
ze_reset	zenithカウントをリセットする。
help	使用できるコマンド一覧をプリントする。
volt [LED amp]	LEDの電圧(V)を変える。
pulse_shot	LEDを撃つ。LEDの電圧を変えた時は必ず実行する。
adc	CAMACによりデータを取得する。後にthresholdとイベント数を決める。
auto_scan	2Dスキャンを開始する。後にzenithステップとazimuthステップを決める。

主要なDOM測定のコマンド

コマンド名	説明
on all	全てのDOMを起動する。HVはかかっていない。
gotoiceboot	on allしたあとに必要。機能はよくわかっていない。
dtsxall	gotoicebootのあとに必要。DOMとPCの情報のやりとりのためのポートを割り当てる。
test.py	DOMのMain-board IDやDOMの名前、内部温度などを調べる。
rate.py	DOMのノイズレートを測定する。
stoptestdaq	DOMが行っている全てのプロセスを消し、電源を切る。
ready	DOMが測定のための準備を行う。
go	readyのあとに、測定を行う。一度readyを実行すれば、何回でも行える。