Milagro Calibration System

Lazar Fleysher Roman Fleysher

June 28, 2000

Abstract

The Milagro detector is an air shower array which uses the water Cherenkov technique and is capable of continuously monitoring the sky at energies near 1 TeV. The detector consists of 20000 metric tons of pure water instrumented with 723 photo-multiplier tubes (PMTs). The PMTs are arranged in a two-layer structure on a lattice of 3 m spacing covering 5000 m^2 area. The direction of the shower is determined from the relative timing of the PMT signals, necessitating a common time reference and amplitude slewing corrections to improve the time resolution. The calibration system to provide these consists of a pulsed laser driving 30 diffusing light sources deployed in the pond to allow cross-calibration of the PMTs. The system is capable of calibrating times and the pulse-heights from the PMTs using the time-over-threshold technique. The absolute energy scale is provided using single muons passing through the detector. The description of the calibration system of the Milagro detector is presented.

Acknowledgments

We would like to express our gratitude to Todd Haines for constant guidance and support without which many hardware problems and questions on methodology and data analysis would still have been open. We thank him for help with preparation of this manual as well. We also acknowledge many discussions and cooperation with Peter Nemethy which led to several new methods of calibration data processing and interpretation. We are also thankful to Isabel Leonor for her help in getting started with the calibration project and suggestions on laser operations. We thank Richard Miller for his help with optics adjustments and manufacturing.

Contents

0 *Release Notes*

GPIB addresses

Device Name	Address
GPIB-ENET	0
DiCon F/O Switch	3
Pulse/Delay Generator	15[1]
Motion Controller	30

Secondary addressing is not supported by any of the devices.

Designated host name and IP address of the GPIB-ENET Controller is: mgrogpib.lanl.gov, 128.165.51.243

Hardware address of the GPIB-ENET Controller is: 00:80:2F:FF:18:E7

All source files reside on mgroems.lanl.gov in `/home/calibration/SOURCE`

```
$Id: Calibration.prj,v 1.5 2000-04-14 17:30:20-06 calibration Exp calibration $

      STABLE LEVEL ----- Calibration.prj ----- STABLE LEVEL

**********************************************************************
   File name       V     Depends
**********************************************************************

SocketIO.c: Fixed a minor problem in handling interrupted system calls in
ReadData() and SendData().
PGenerator.c: PG_Delay() changes the delay in the trigger relative to the
default value. First delimiter in MorseCode is at 400Hz, the rest at
200Hz.
MController.c: MM_Generator() now counts the number of pulses sent to the
laser and returns it when the generator is turned off.

Begin.c: TDC-laser calibration is now an option. It will fire the laser
with different(preset) trigger delays. It will report to the logfile, the
number of triggers sent to the laser. For better userinterface, index.html
is now a symlink and is automatically set to point to
choose.html/progress.html

Current (fifth) release of the calibration software is:

Acquire.c         1.3  GpibErr.c        v1.2
                       Logger.c         v1.2
                       MController.c    v1.4
                       PGenerator.c     v1.5
```

[1]The default address can be changed from the front panel of the device!

```
                        ESP-488 package
Begin.c            1.5  Acquire.c        v1.3
                        Finish.c         v1.1
                        Logger.c         v1.2
                        MController.c    v1.4
                        OpticalSwitch.c  v1.2
                        PGenerator.c     v1.5
                        ReadURL.c        v1.1
                        SocketIO.c       v1.2
Begin.make         1.3
Finish.c           1.1
GpibErr.c          1.2  ESP-488 package
Logger.c           1.2
MController.c      1.4  ESP-488 package
                        GpibErr.c        v1.2
                        Logger.c         v1.2
OpticalSwitch.c    1.2  ESP-488 package
                        Acquire.c        v1.1
                        GpibErr.c        v1.2
                        Logger.c         v1.2
PGenerator.c       1.5  ESP-488 package
                        GpibErr.c        v1.2
                        Logger.c         v1.2
ReadURL.c          1.1
SocketIO.c         1.2
Server.c           1.2  SocketIO.c       v1.2    # runs on kahuna

ESP-488/
   esp.c
   esp.h
   espproto.h
   gpibcnst.h
```

National Instruments ESP-488 packages has been modified and fixed. `iblock()` and `ibunlock()` are still in alpha-level testing.

1 User's Manual

A true black box is a system which can be fully exploited without knowledge of what is inside of it.

1.1 Introduction[8].

This section describes the method which was used to calibrate the Milagro detector. The layout of Milagro detector is described in (McCullough et al., 1999), but, for clarity, some information is provided here.

Milagro is the first detector designed to study air showers at energies near 1 TeV using water Cherenkov techniques. The detector is built in the Jemez Mountains near Los Alamos, New Mexico at an altitude of 2650 m. The pond, which is 60m x 80m x 8m, is filled with clean water, covered with a light barrier and instrumented with 723 - 20 cm PMT's.

The PMTs collect Cherenkov light produced by the shower particles which traverse the detector's water volume. Whenever a PMT pulse exceeds a preset discriminator threshold a multihit time-to-digital converter (TDC) is started. Each PMT has its own TDC which is capable of recording up to 16 discriminator level crossings per event with 0.5 ns resolution. These constitute the raw data from the PMT.

The calibration procedure described below is applied in order to transform the raw counts to physically meaningful arrival times and light intensities which then can be used for event reconstruction. Considerable effort has been made to construct all PMT channels of the detector as uniformly as possible. However to achieve the high precision required for the event reconstruction, the remaining variations between channels have to be compensated for. A separate set of calibration parameters is determined for each PMT channel.

The calibration system has been designed to reflect the physics goals of the detector and is capable of calibrating the times recorded by each PMT to provide the best available shower direction; it is capable of calibrating the "pulse-heights" from the PMTs needed to estimate the absolute energy of each event. The absolute time of the events is retrieved from the GPS (Global Positioning System) clock system to 100 ns accuracy.

1.1.1 Pointing

The desire to reconstruct the position of events on the Celestial sphere with systematic errors $\ll 1^\circ$ dictates that the locations of the photo-tubes be known to about 10 cm accuracy in horizontal direction and about 3 cm in vertical. To meet this requirement, photographic and theodolite surveys of the pond have been performed. At the end of the construction period, when the pond was filled with water, an "as-built" measurement of the elevation of all PMTs has been made.

Knowledge of the PMT coordinates is necessary, although not sufficient, to achieve the stated goal. Times, registered by PMTs, have to have resolution of about 1 ns. This is limited by the transit time jitter of the PMTs at low light levels. Thus, it is important to calibrate the TDC conversion factors, compensate for the PMT-pulse amplitude dependence of TDC measurements (known as a slewing correction) and synchronize all TDCs (find TDC time offsets) to the required accuracy.

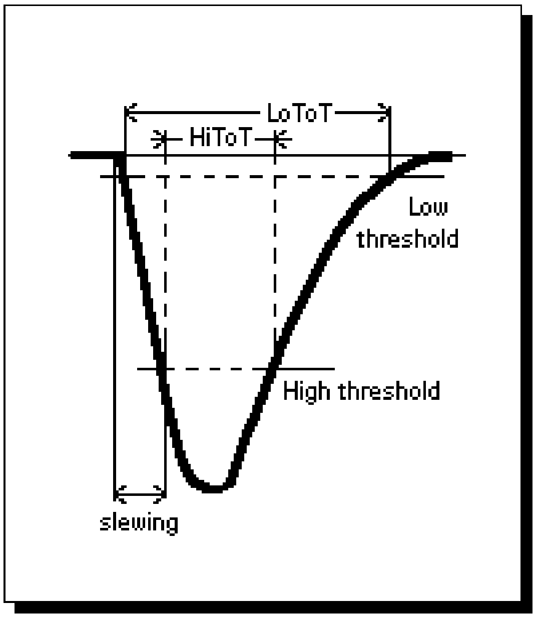


Figure 1: Time-over-Threshold concept.

1.1.2 Energy

The relative "pulse-height"-to-photo-electron conversion must be determined to convert all amplitude measurements to a common unit for each event. The photo-electron counts (PEs) then have to be converted to the absolute scale of the energy deposited in the water to reconstruct the shower size and, ultimately, to estimate the energy of the primary particle.

1.2 Time-over-Threshold

Traditionally, the area of the PMT pulses is measured using amplitude-to-digital converters (ADCs). The major draw-back of this method is that ADCs have narrow dynamic range and are relatively slow devices which causes dead time during data taking. A new technique has been developed to overcome these problems.

The idea behind the Time-over-Threshold (ToT) method is simple (See Fig. 1). The PMT pulse quickly charges a capacitor C, which is then slowly discharged via a load resistor R. In such a setup the total area of a PMT pulse can be measured by the discharge time (time over threshold).

This method will work only if the time between registered pulses is greater than the discharge time constant $\tau = RC$. Two small pulses not separated in time will appear as one large pulse. To avoid this problem and prepulsing, and for noise reduction, a second higher threshold level had been introduced. Now true large pulses cross both thresholds and time-over-high-threshold (HiToT) is a much better measure of the pulse area. It also provides a method of separating small single pulses from everything else.

Similarly, the PMT pulse amplitude is related to ToT and the pulse amplitude dependence of TDC measurements can be compensated using ToT.

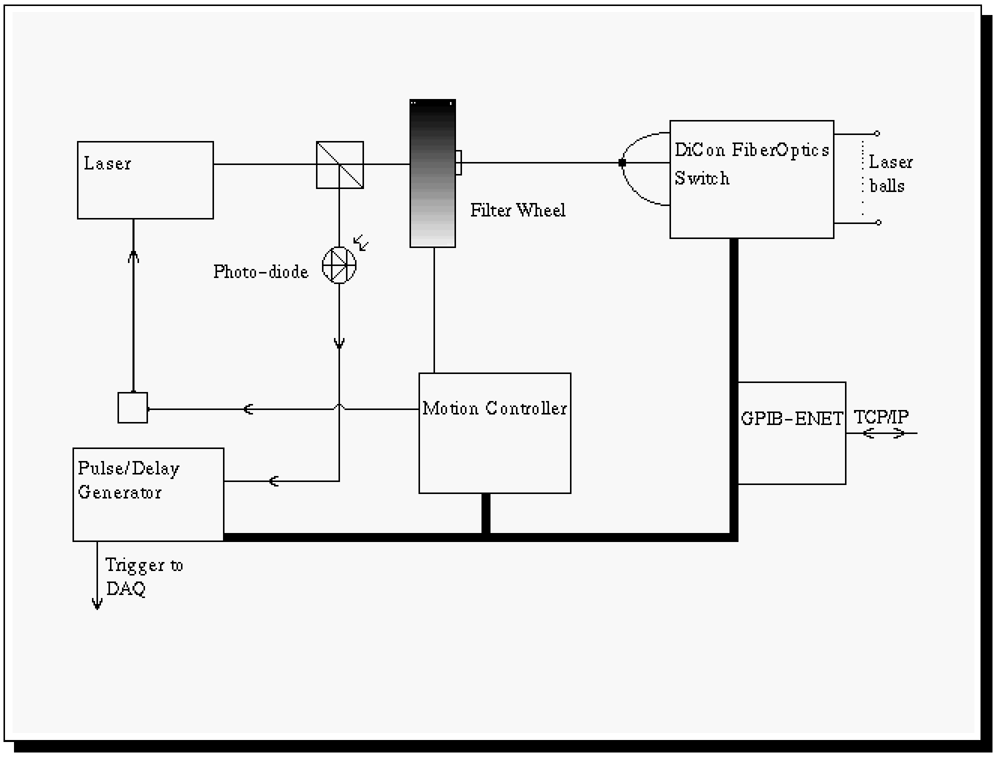


Figure 2: Calibration system setup

1.3 Laser calibration system: description

The Milagro calibration system is based on the laser – fiber-optic – diffusing ball concept used in other water-Cherenkov detectors (See, for instance, Becker-Szendy et al., 1995). A computer operated motion controller drives a neutral density filter wheel to attenuate a pulsed nitrogen dye laser beam. The beam is directed to one of the thirty diffusing laser balls through the fiber-optic switch (See Fig 2). Part of the laser beam is sent to a photo-diode. When triggered by the photo-diode, the pulse-delay generator sends a trigger pulse to the data acquisition system. A laser fire command is issued by the motion controller, providing full automation of the calibration process. The balls are floating in the pond so that each PMT can "see" more than one light source. Such a redundant setup allows us to calibrate the PMTs and the electronics.

1.4 Timing calibration

Because of finite rise-time of a PMT pulse, its registration time depends on the amplitude of the signal. The corrections were found by studying how TDC outputs vary as a function of PMT-pulse ToT. For different laser pulse intensities, the time of registration (t_{reg}) of the PMT response by its TDC with respect to the photo-diode "zero" and ToT were measured. The slewing correcting curve was found by fitting a polynomial to t_{reg} vs ToT.

However, since all the time measurements are done with respect to the photo-diode, the slewing curve is artificially shifted by fiber-optic delay, light travel time in water and TDC

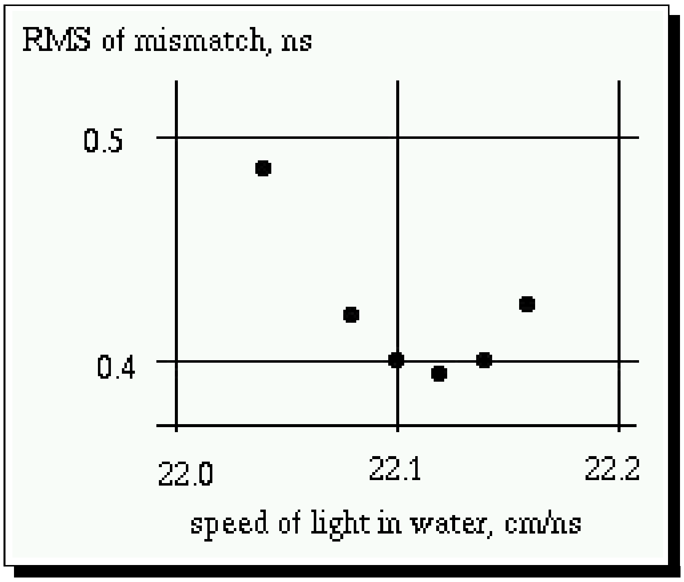


Figure 3: Width of the TDC-offset mismatch distribution as a function of speed of light in water.

time offsets. Knowing the locations of the diffusing laser balls and PMTs, the speed of light in water and fiber-optic delay, the TDC offsets can be found.

This procedure has been repeated for both low and high thresholds (LoToT and HiToT) for each PMT-TDC channel. Now, a meaningful interpretation for the TDC outputs exists.

1.5 Consistency check

The Milagro calibration system has been designed to allow cross-calibration of the PMTs. This fact was used to check the accuracy of the obtained calibration parameters and disclose some problems.

TDC time offsets obtained for a given PMT from different laser balls should be identical. Thus, the TDC-offset mismatch distribution becomes a useful diagnostic tool. The mean mismatch in offsets over all PMTs from two laser balls gives the fiber-optic delay difference between them. The width of the mismatch distribution is a measure of the offset quality. In fact, if we used a wrong speed of light in water, it would widen the mismatch distribution. Eventually, this allowed us to determine the effective speed of light in the pond water to four decimal places, by comparing measured offsets from a pair of far separated laser balls (See Fig. 3).

Another use of redundancy was the reconstruction of laser ball coordinates. Direct reconstruction of the coordinates by reversing the procedure in section 1.4 will yield either a perfect result or will lead to an inconsistency. In either case, it will not give any constructive information. To overcome this difficulty, a method of pairwise correlations was developed to obtain the laser ball locations. The positions of two laser balls s1 and s2 can be obtained by comparing relative time differences from the pair as observed by a PMT. If we define:

$$\tau = t_{s1 \to PMT} - t_{s2 \to PMT}$$

where $t_{s1\rightarrow PMT}$ and $t_{s2\rightarrow PMT}$ are TDC times, then τ does not depend on the TDC time offset for the particular PMT channel. From geometrical point of view, a given PMT lies on a branch of hyperbola, defined by parameter τ and with the laser balls s1 and s2 at its foci. Thus, four different PMTs with their corresponding τ's for the same laser ball pair would define the coordinates of the foci; by using more PMTs the problem becomes over-constrained and yields a best fit. This procedure was used successfully to reconstruct laser ball coordinates for Milagrito and Milagro.

1.6 Energy calibration

The ToT information was converted to a pulse amplitude scale by moving a set of ADCs to all PMT channels and collecting simultaneous ToT and ADC data. The single photo-electron peak was clearly visible yielding the ToT-to-PE conversion, assuming the ADC outputs are linear in the number of PEs. Alternatively, assuming that the number of registered PEs obeys a Poisson distribution, the occupancy method was used to obtain the ToT-to-PE calibration. Both methods are in reasonable agreement.

Absolute energy calibration measurements will be done using through-going muons. The imaging capabilities of the detector will be exploited in order to find, fit and select well-defined through-going muon tracks. Once the geometry of the track is known, the Cherenkov energy deposit will be estimated and compared against the photo-electron distribution in the event. This was the primary absolute energy calibration method used in the IMB detector (Becker-Szendy et al., 1995).

References

Becker-Szendy, R. et al., Nucl. Instr. and Meth. A **352** 629 (1995)
McCullough, J. et al., Proc. 26th ICRC (Salt Lake City, 1999), HE.6.1.02

1.7 Laser pointer.

Laser pointer is designed and installed for the purpose of study of water properties.

1.8 Operation

1.8.1 Starting and Stopping calibration

- Laser Shack

  1. Turning the laser ON

     - open the gas bottle (open the valve about 1/4–1/2 of a turn)
     - turn on power/key on the laser
     - set laser trigger to internal and run at moderate rate (~5Hz) for 20-30min. to warm-up.

2. Final check
   - turn on the NIM crate
   - turn on the "master" switch on surge protector
   - turn on pulse/delay generator on surge protector
   - set repetition rate on laser to maximum
   - measure the rate, should be >19Hz
3. Remote Enable !!!!
   - set laser trigger to external
   - going left to right (!! in this order !!) on surge protector, turn on:
     (a) "Master" switch (should be already on)
     (b) Pulse/Delay generator (should be already on)
     (c) GPIB controller
     (d) Fiber Optic switch
     (e) Motion controller (MM3000)

- Counting House
  1. Reconfigure DAQ to take calibration data
     - page shift person with a message: "`taking calibration data. ignore pages`"
     - make an entry on the "status" page
     - STOP data taking
     - replace shower trigger with calibration trigger
       * trigger to disconnect is in the 4-fold logic gate in the 3rd slot of the "trigger" NIM bin, channel 'B'
       * calibration trigger is the TTL< − >NIM converter in the 7th slot of the same bin
     - start calibration server
       * log into kahuna as daq
       * `%cd /CONTROL/CALIB`
       * `%./Server.exe &`
       * check that server is running and waiting for connection by `%ls -l lock`. The output should be `lock ->pid:idle` where pid is the program id of the server, idle is the state of the server
  2. Taking data
     - open a web browser to http://mgroems/∼calibration.
     - choose laserballs and filter wheels to use; set the desired laser trigger rate.
     - START the DAQ system from a terminal window.
     - After DC2 reset is finished, press "START" button on calibration page.

    - Periodically monitor the laser output trigger rate.
    - on kahuna `%ls -l /CONTROL/CALIB/lock` should give `lock->pid:ball#:wheel#` where ball# and wheel# are current calibration settings.
    - If data taking is interrupted, data for the current laser ball should be discarded.

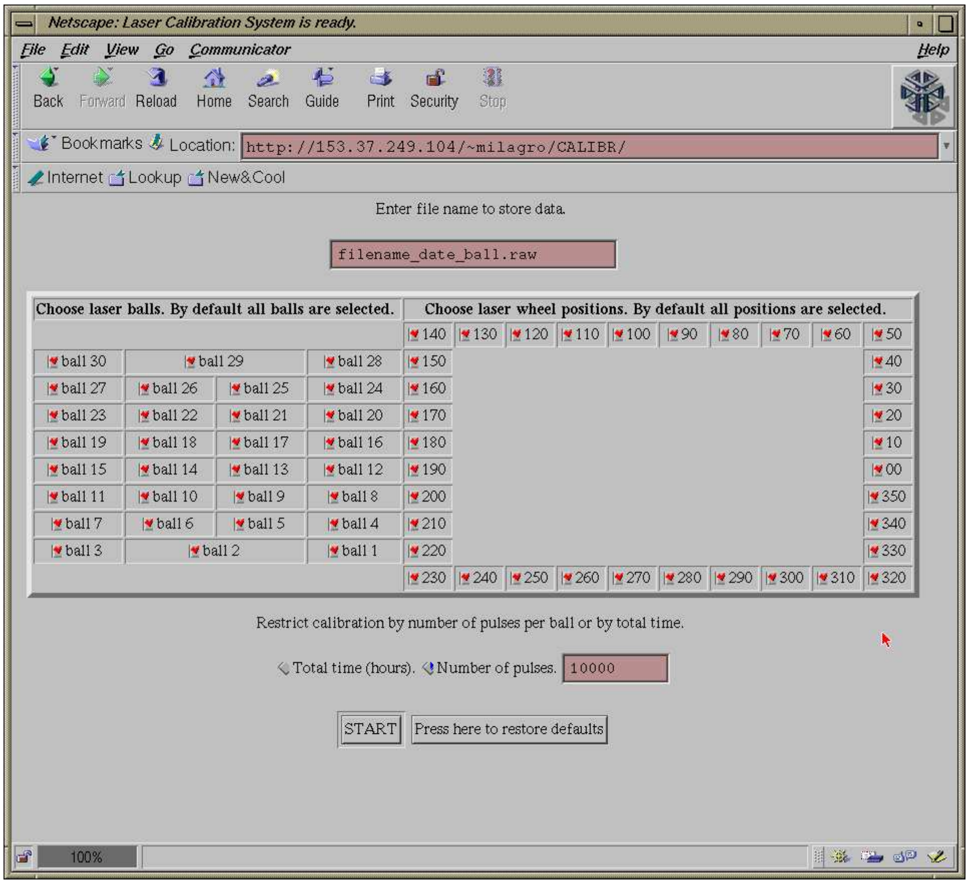
  3. Calibration is finished
    - close the calibration web page
    - STOP the run on a DAQ terminal window
    - re-connect the shower trigger
    - kill calibration server
      * `%ls -l /CONTROL/CALIB/lock`
      * `%kill pid` where pid is the output of the `ls` command
    - START new run
    - page shift person with a message: "`normal data taking is resumed, respond to pages`"
    - make a "status" page entry
    - make a log book entry

- Laser Shack. Turn everything OFF
  - on surge protector, going right to left (!!), turn off:
    1. Motion controller (MM3000)
    2. Fiber Optic switch
    3. GPIB controller
    4. Pulse/Delay generator
    5. "Master" switch
  - turn off the NIM crate
  - turn off the laser power/key
  - close the valve on the gas bottle

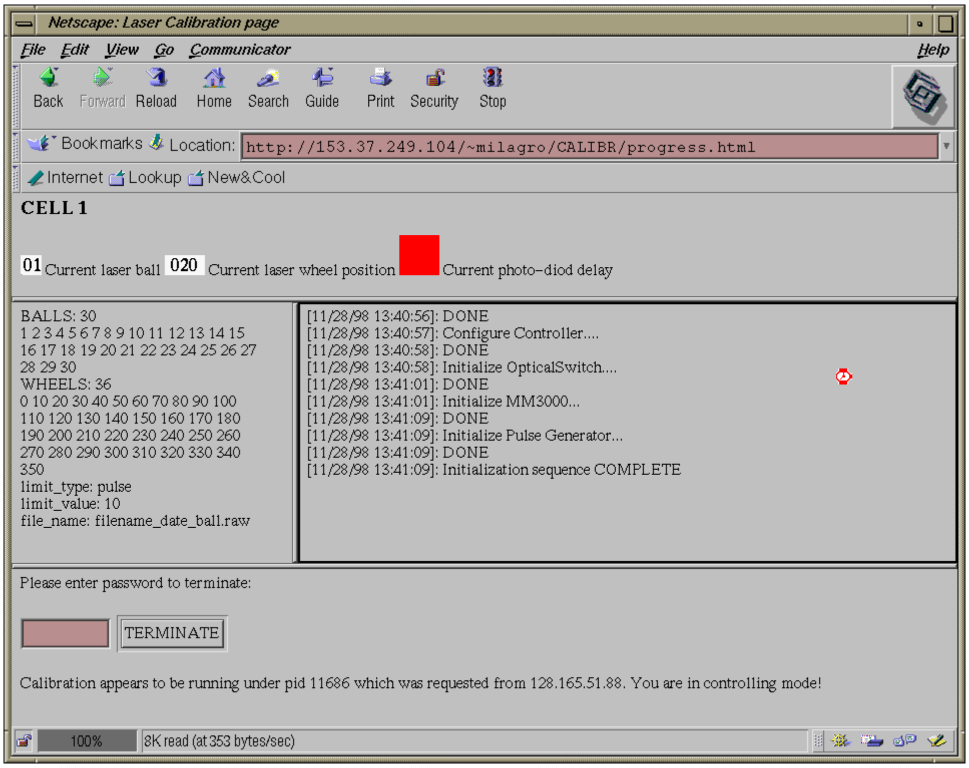
1.8.2 Remote Control

The web controls are designed to facilitate the management of the calibration system. The first page will ask a user for a specific set of laser balls which have to be fired and the filter wheel positions (attenuation) to be used during the calibration run, laser trigger frequency and type of calibration (Slewing and PE or TDC). The user can also choose how long the whole process should take or how many pulses per setting should be issued. All those parameters are only approximate!

After pressing the "Start" button, the calibration sequence will initialize. The left vertical frame of a new page will show the chosen parameters. The upper horizontal frame will show the current ball and wheel settings. The main frame will show last few lines of the calibration



Netscape: Laser Calibration System is ready.
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://153.37.249.104/~milagro/CALIBR/
Internet Lookup New&Cool
Enter file name to store data.
filename_date_ball.raw
Choose laser balls. By default all balls are selected.
Choose laser wheel positions. By default all positions are selected.
ball 30 ball 29 ball 28
ball 27 ball 26 ball 25 ball 24
ball 23 ball 22 ball 21 ball 20
ball 19 ball 18 ball 17 ball 16
ball 15 ball 14 ball 13 ball 12
ball 11 ball 10 ball 9 ball 8
ball 7 ball 6 ball 5 ball 4
ball 3 ball 2 ball 1
140 130 120 110 100 90 80 70 60 50
150 160 170 180 190 200 210 220
40 30 20 10 00 350 340 330
230 240 250 260 270 280 290 300 310 320
Restrict calibration by number of pulses per ball or by total time.
Total time (hours). Number of pulses. 10000
START
Press here to restore defaults
100%



Netscape: Laser Calibration page
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://153.37.249.104/~milagro/CALIBR/progress.html
Internet Lookup New&Cool
CELL 1
01 Current laser ball 020 Current laser wheel position Current photo-diod delay
BALLS: 30
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27
28 29 30
WHEELS: 36
0 10 20 30 40 50 60 70 80 90 100
110 120 130 140 150 160 170 180
190 200 210 220 230 240 250 260
270 280 290 300 310 320 330 340
350
limit_type: pulse
limit_value: 10
file_name: filename_date_ball.raw
[11/28/98 13:40:56]: DONE
[11/28/98 13:40:57]: Configure Controller....
[11/28/98 13:40:58]: DONE
[11/28/98 13:40:58]: Initialize OpticalSwitch....
[11/28/98 13:41:01]: DONE
[11/28/98 13:41:01]: Initialize MM3000...
[11/28/98 13:41:09]: DONE
[11/28/98 13:41:09]: Initialize Pulse Generator...
[11/28/98 13:41:09]: DONE
[11/28/98 13:41:09]: Initialization sequence COMPLETE
Please enter password to terminate:
TERMINATE
Calibration appears to be running under pid 11686 which was requested from 128.165.51.88. You are in controlling mode!
100%
8K read (at 353 bytes/sec)

system log file. It will give some additional information if a problem occurs. If there is a need to cancel the calibration process, a required password should be entered in the lowest horizontal frame.

Even though all information on the calibration status page is updated via "server push",[2] it is safe to press the "STOP" button on your browser and close the browser at all. Reloading the page will automatically "connect" back to the active status page.

If a user tries to start calibration and it is already running, the chosen parameters will be ignored and the web page will "connect" to the current calibration process. The user will be informed by an appropriate message in the lowest frame.

Of course, your browser has to support frames otherwise you probably will not be able to review current status of the process. However, it does not prevent you from starting one.

1.9 Maintenance

In general, all of the devices are maintenance free. However, the laser requires some routine servicing. After prolonged use, the laser may become dirty and must be cleaned. When it is new, it should be cleaned every $5 \cdot 10^5$ shots, but as the laser ages, the interval between cleaning may increase to $\sim 5 \cdot 10^6$ shots. Consult the laser manual.

1.10 Troubleshooting

Majority of errors which might occur during operation of the calibration system are fatal and as a result of such an error, the calibration process will be aborted. See table 1 for possible solutions to selected problems.

If during calibration process "dirty"power goes down — turn off the equipment in the laser shack. Terminate the calibration program. "Home" position of the filter wheel might be lost!

[2]Netscape 1.1 and later versions support "server push", however, all versions of Internet Explorer – do not!

Problem	Solution
Optical Switch is not ready after "clear".	Calibration will terminate automatically. Restart calibration program from the control panel.
Wheel remains parked, does not move.	MM3000 is hung. Terminate the calibration program and restart it. It might be required to cycle the power on the MM3000. In any case one should let it warm-up for 10-20 min.
Connection to DAQ server refused.	Calibration server is not running. Start the server on kahuna.
GPIB: Can not acquire unit descriptors.	Network connection to GPIB-ENET or 'dirty' power is down. Check the power and the network connection.
For some reason need to terminate calibration.	Enter password and press "Terminate". Wait for at least 1min for wheel to reposition before starting again or turning the power off.
Starting the Server.exe produces a message: `lock is present`.	Most probably Sever is running or did not clean-up due to power failure. Check if Server.exe is running.
Fuse detects slow trigger	Check the laser

Table 1: Possible problems and solutions

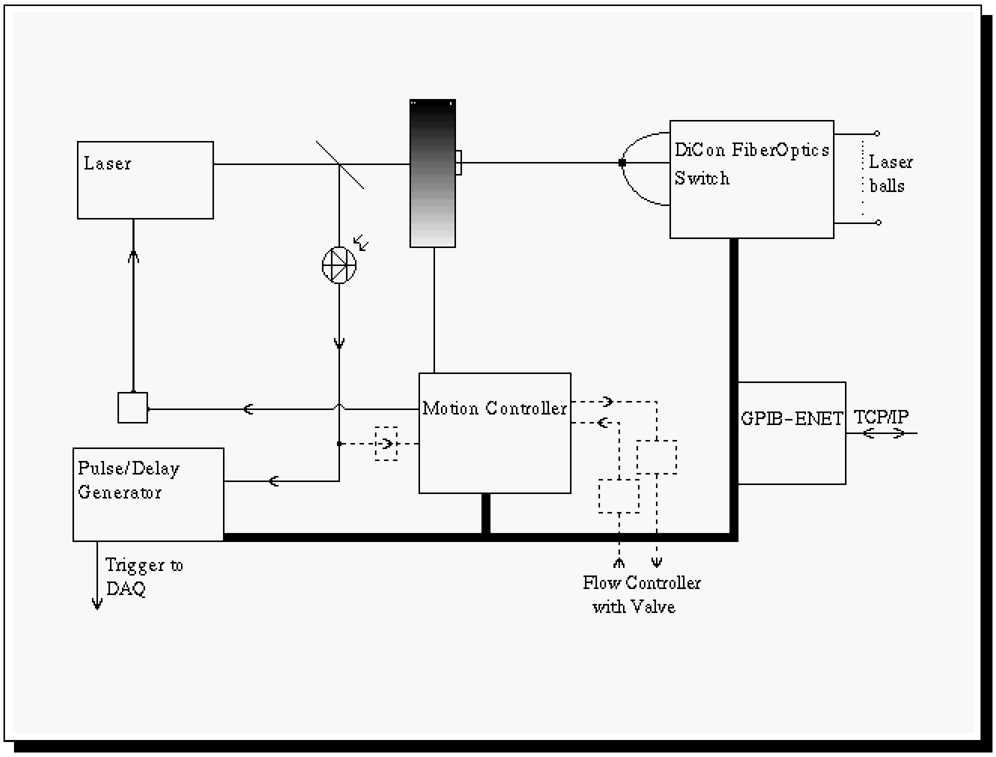


Figure 4: Calibration system setup

2 Hardware manual.

It is easier to change the specifications to fit the program, than vice versa

The aim of this section is to give a general description of all the equipment installed in the laser shack and some auxiliary information about each device. One should consult corresponding manuals for more detailed description.

2.1 General

The calibration system should be fully automatic and remotely controlled. This imposes some additional requirements on the hardware used. It had been chosen to use TCP/IP network to control the equipment. However, the major fraction of the devices needed to build the calibration complex support either serial or GPIB protocol. GPIB-ENET controller serves as the necessary interface between the devices and the network.

2.2 DiCon FiberOptics Switch[2]

DiCon's FiberOptic Facility Switch MC606 is designed to handle up to four 1xN stepper motor switches with up to 11 channels each and up to six On-Off solenoid switches.

Our Switch is equipped with 3 stepper motor switches, 10 channels each and no solenoid ones.

Power-On Self Test

When the switch is turned on, the front panel displays GPIB address setting for about 5 seconds. After that, the switch runs through a 5 second self test in which all the switches are parked to the "OFF" or 0 position. Thus, total duration of the self test is about 10 seconds.

GPIB Control

Complete control of the switch is possible using only a few high level commands:

Function	Call	Timing
Write	int ibwrt(int ud, char *command, int length)	300μsec/char
Read	int ibrd(int ud, char *string, int length)	300μsec/char
Clear	int ibclr(int ud)	10 sec
Go To Local	int ibloc(int ud)	300μsec
Serial Poll	int ibrsp(int ud, char *string)	700μsec

Serial Poll Response Byte

Response on a serial poll is a single byte:

Hex	Decimal	Binary	Meaning
00	0	00000000	Busy
01	1	00000001	Ready
08	8	00001000	Power-On Test Failure
09	9	00001001	Bad Byte 1 in Channel number
0A	10	00001010	Bad Byte 2 in Channel number
0F	15	00001111	Proximity Failure
20	32	01000000	Reconfiguration Command Acknowledged

Select a New Channel Configuration

<u>1xN Switches:</u>

Generally, each reconfiguration command is a two-byte word "[B][C]" where [B] is the bank (A,B,C,..) and [C] is channel number within the bank. C={0,1,...9,:,;} Notice that ':' and ';' follow the '9' in the ASCII code and thus represent 10-th and 11-th channels.

Examples:

A1 Turns Bank A Switch to Channel 1
C: Turns Bank C Switch to Channel 10
B0 Turns Bank B Switch to Channel 0 (OFF)
P Park. Turns All Bank Switches to Channel 0 (OFF)

```
sprintf(cmd,"%c%c",'A','1');  /* Configure the switch to A1 */
ibwrt(ud,cmd,strlen(cmd));
```

After a channel reconfiguration command is sent, the switch will issue a service request (SRQ). Subsequently, polling the switch will produce a busy response until it has completed its switching motion. At that point the switch should be ready for a new command, and will respond to a poll with the Hex 01 “ready” message. The sequence of serial status polls is 20,00,00, ...,01 if successful. **If 20 is not returned issue the ibwrt command again.** If an error is found the sequence would be 20,00,00,09(0A,0F),01. **The calling program should have the loop-count escapes to insure that it will not lock up if 20 or 01 codes are not returned.**

Here is an example of a function which does that:

```
int PollSwitch (int ud)
/* ud- is unit descriptor of the switch */

/* PollSwitch Return
 *  0  Ready
 *  1  Out of Range
 *  2  Proximity Failure
 *  3  Time Out
 *  4  Serial Poll/Power-On Test failed
 */
{
  unsigned ret;
  int i=0, timeout=20000;

  do
    {
     if(ibrsp(ud, &ret)&ERR)
       {
        printf("SWITCH: Could not conduct a serial poll.\n");
        return 4;
       }
     switch(ret)
       {
        case 8:  /* ==== Power-On System Test Failure  ===== */
                 printf("SWITCH:Power-On Test Failure.\n");
                 return 4;
        case 9:  /* ===Channel number out of range: Byte 1 09h=== */
                 printf("SWITCH: Byte 1 in channel number is out of
                        range.\n");
                 return 1;
        case 10: /* ===Channel number out of range: Byte 2 10h=== */
                 printf("SWITCH: Byte 2 in channel number is out of
```

```
                        range.\n");
                 return 1;
        case 15: /* === Proximity Failure 0Fh === */
                 printf("SWITCH: Proximity Failure!\n");
                 return 2;
        default: i=i+1; break;
       }
    } while(i<=timeout && ret!='\x01');
  if (ret!='\x01')
    {
     printf("SWITCH: Serial Poll timeout. Device is not responding.\n");
     return 3;
    }
  else return 0;
}
```

Read Status String

As soon as the switch has completed its motion the new status string is available to read. **The current status is returned as an 11-byte string. (The manual says 10!)** First four bytes represent 1xN switches, next six indicate On-Off switches. The last character of the string is a carriage return (0x0D).

For 1xN switches a numerical decimal indicates the current switch channel status. For On-Off switches, a "0" indicates an "OFF", "1" — "ON". **If a switch is not present or nor operating correctly, its status byte will be a "*".**

Example: `ibrd(ud,buffer,11)` may return

03;*0011**

1-st module is on channel 0, 2nd – on 3, 3rd – on 11, 4th – not present
first 2 On-Off switches are Off, second 2 are ON, last two are not present

Hard Failures

Under normal operating conditions the switch should never require resetting as a constant potential maintains the position of the armature. However, if optical path continuity is known to be lost, park command should be used to recalibrate the specific module.

If serial poll returns "0F", the bad switch may be indicated by an "*" in the status string. Recalibrating the module by parking it may not recover the module. Sending `ibclr(ud)` will reset each switch module. If this does not help, the module needs repair or replacement.

2.3 MM3000 Motion Controller[3]

The MM3000 is a stand-alone integrated motion controller/driver. It can control up to four axes. Communication with the controller can be established via standard IEEE 488 or RS-232C interface.

Other important features include:

- 8-bit digital user definable inputs/outputs
- 8 channel 10 bit analog to digital converters 0-9V
- Remote motor off input (interlock)

GPIB Programming

As commands arrive from the host terminal (GPIB Controller), they are placed into the command buffer. When a carriage-return is received, the command is interpreted. If the command is valid and its parameter is within the specified range, it will be executed. If the command contains an error, it will not be executed and a corresponding error message will be stored in the output buffer.

Also note the necessity explicitly instruct the MM3000 with the `WS` or `WA` command to wait for an axis to stop. **Unless instructed otherwise, the MM3000 executes commands in the order received without waiting for completion of previous commands.**

Communication Protocol

The IEEE488 interface is implemented somewhat differently from a typical instrument because the standard IEEE488.2 command set and format are inadequate for a complex motion controller. The IEEE488 interface is used **only** as a communication port. The extended IEEE488.2 protocol is **not** supported.

MM3000 features a 512 byte command input buffer. If the buffer fills up, the MM3000 will not allow further communication until space becomes available. To send a command to the MM3000, use the command specific to your IEEE488 terminal (`ibwrt(ud,cmd,cmd-len)`).

An MM3000 command (or a command sequence) has to be terminated with a carriage return character. Responses from the MM3000 are always terminated with a carriage return/line feed combination. This setting may not be changed. IEEE controller has to be configured to terminate the input (read) function when it senses the line feed character. (End-of-String(EOS)).

Use of SRQ Line

The MM3000 can be programmed to generate an IEEE488 service request on occurrence of certain events. These events are:

- Motion complete on any of the 4 axes, or command errors and messages. The `FI` command can be used to select the event that should cause this type of service request.

- Execution of an `RQ` command. Sometimes, it may be more efficient and desirable to generate a service request when a sequence of commands is completed. If `FI` interrupts are disabled, a service request is generated only when an `RQ` command is executed.

Summary

1. The general format of a command is **xxAAnn**. Both upper and lower cases are accepted. Depending on a command **xx** and **nn** might be optional or required.
2. Blank spaces are allowed and ignored. Use blank spaces with restraint.
3. Commands are executed line by line. A line can consist of one or more commands. Controller interprets the commands in order they are received. Maximum number of characters allowed on a command line is 80. **To be executed correctly, a line MUST consist of ONE command ONLY. Maybe it is a feature of 2.3a firmware. . .**
4. Commands issued on the same line must be separated by semicolons(;)
5. Each command must end with a line terminator (carriage return).
6. Command missing axis prefixes default to the last reference, e.g., `1PA+1000;WS` the WS defaults to axis 1.
7. The `EP` command invokes program entry mode.
8. `/QP` starts a new program.
9. The % command causes the system to exit program entry mode.
10. A stored program must be successfully compiled with the `CP` command. The `EX` command compiles and executes programs if no errors exist.
11. **New real-time commands are accepted and executed even while running a program.**

GPIO Connector Pinouts

Labeling Conventions

GND	$\Rightarrow$	ground
N.C.	$\Rightarrow$	Not connected
UTIL	$\Rightarrow$	Test/utility signal. **DO NOT USE**
I	$\Rightarrow$	Input
O	$\Rightarrow$	Output

GPIO Connector (37-Pin D-Sub)

The MM3000 features this 37 pin input-output connector for access to its auxiliary signals. These signals provide the ability to interact with and control the environment of a motion control system.

Pin #		**Description**
1	—	**Channel 1 A/D Input** ***(10 bit resolution)***
2	—	**Channel 2 A/D Input** ***(10 bit resolution)***
3	—	**Channel 3 A/D Input** ***(10 bit resolution)***
4	—	**Channel 4 A/D Input** ***(10 bit resolution)***
5	—	**Channel 5 A/D Input** ***(10 bit resolution)***
6	—	**Channel 6 A/D Input** ***(10 bit resolution)***
7	—	**Channel 7 A/D Input** ***(10 bit resolution)***
8	—	**Channel 8 A/D Input** ***(10 bit resolution)***
9	—	**A/D GND**
10	—	**A/D Reference voltage**
11	—	**A/D** V_{DD} **input** `(supply voltage for A/D converter)`
12	—	**N.C.**
13	—	**N.C.**
14	—	**N.C.**
15	—	**N.C.**
16	—	**N.C.**
17	—	**N.C.**
18	—	**UTIL**
19	—	**Channel 1 D/A output** ***(8 bit resolution)*** `optional`
20	—	**Channel 2 D/A output** ***(8 bit resolution)*** `optional`
21	—	**Channel 3 D/A output** ***(8 bit resolution)*** `optional`
22	—	**Channel 4 D/A output** ***(8 bit resolution)*** `optional`
23	—	**D/A GND**
24	—	**Programmable TTL I/O, bit 1**
25	—	**Programmable TTL I/O, bit 2**
26	—	**Programmable TTL I/O, bit 3**
27	—	**Programmable TTL I/O, bit 4**
28[3]	—	**Programmable TTL I/O, bit 5**
29[4]	—	**Programmable TTL I/O, bit 6**
30[5]	—	**Programmable TTL I/O, bit 7**
31[6]	—	**Programmable TTL I/O, bit 8**
32	—	**-12V**
33	—	**+12V**
34	—	**+5V**
35	—	**GND**
36	—	**System reset**
37	—	**N.C.**

[3]Gas valve on/off

[4]Laser trigger.

[5]Slave, turn the laser on/off. Shorted to 31

[6]Master, turn the laser on/off. Shorted to 30

2.4 Pulse Generator[4]

GPIB Programming

Command errors are reported in the Error Status Byte. The controller should verify that the Error Status Byte is zero after each command is sent. When an error is detected, all of the pending commands in the communication buffer are canceled.

Blanks are ignored, case is ignored, multiple commands may be sent on one line if separated by semicolons. All responses are terminated by a carriage return and line feed. The line feed is sent with an EOI (End-or-Identify line is asserted).

The DG535 will respond to GPIB commands only if it is addressed and the Remote Enable line (REM) is asserted. When this happens the device goes to REM state, all keys except the “mode” key are disabled which allows user to leave the REM state. A controlling program can prevent users from using the keyboard by setting the GPIB bus to the Local Lockout state (LLO).

The instrument requires 1μs after last delay has finished to reset all the delay channels. (thus, maximum trigger rate is $10^6 Hz = 1 MHz$)

2.5 GPIB-ENET[1]

In 1965, Hewlett-Packard designed the Hewlett-Packard Interface Bus (HP-IB) to connect their line of programmable instruments to their computers. Because of its high transfer rates (1MBytes/s), this interface bus quickly gained popularity. It was later accepted as IEEE Standard 488-1975, and has evolved to ANSI/IEEE Standard 488.1-1987. Today, the name General Purpose Interface Bus (GPIB) is more widely used than HP-IB. ANSI/IEEE 488.2-1987 strengthened the original standard by defining precisely how controllers and instruments communicate. Standard Commands for Programmable Instruments (SCPI) took the command structure defined by IEEE 488.2 and created a single, comprehensive programming command set that is used with any SCPI instrument.

GPIB-ENET is GPIB-Ethernet controller that enables connecting instruments to a computer network. GPIB-ENET supports TCP/IP protocol since it must be a part of a network and IEEE 488 standard to communicate with instruments. It is expected to recognize commands sent to it from an instrument, send commands to the instrument, answer to the host address it received commands from, etc.

GPIB devices communicate by message exchange. There are two kinds of messages: data which refer to GPIB devices and commands which refer to the bus interface.

Main component of GPIB-ENET module is 16-bit microprocessor, which supervises data flow from Ethernet to GPIB and back. Inside EEPROM is firmware that comes as software module support. The interface to Ethernet is ST-NIC Ethernet controller, product of National Instruments, while interface to GPIB devices is Turbo488/NAT4882 GPIB controller, which is compatible to ANSI/IEEE 488.2-1987 standard.

The module is controlled by specially developed NI-488.2 software package, a product of National Instruments. **However, one should use the ESP-488 software kit in operating environments that do not have a corresponding NI-488.2 software package.**

2.6 Laser

Abandon hope, all ye who enter here...

The LN120C NitroDye laser is a class IIIb UV (337.1 nm) pulsed Nitrogen laser with an optional dye-laser module. The dye modules use the nitrogen beam as the pump source and can be installed/removed/replaced when required. While the nitrogen laser wavelength is fixed, the modules offer variable wavelengths. The mechanism for this feature is based on emission properties of certain organic liquids (dyes) when irradiated with an intense UV light. Depending on the module, the dye laser can emit a broadband radiation or a spectrally narrow one. The LN120C delivers up to 250 kW of peak power in a 300 ps pulse.

Tuning

The laser pulse width is approximately 300 ps when the laser head is operated at 1(one) atmosphere (i.e. a gauge pressure of zero!). If it is desirable to have even shorter pulses, these may be obtained by increasing the pressure in the laser head.

Adjusting the laser output is easier if the discharge plasma is viewed while adjusting the electrode spacing. To do this, open the laser and remove the laser head cover. Defeat[7] the interlock and turn the laser on. The discharge plasma can be observed by looking through the laser head lid.

When the laser is operating optimally, the plasma will be a uniform purple glow discharge and the spacing between the electrodes will be as wide as possible without the formation of bright white arcs. It is possible to operate the laser from 0 Hz to about 20 Hz without adjusting the electrode spacing with little or no decrease in power.

Cleaning

After prolonged use, the laser head and the spark gap may become dirty and must be cleaned. When the laser is new it should be cleaned every $5 \cdot 10^5$ shots, but as the laser ages, the interval between cleaning may increase to $\sim 5 \cdot 10^6$ shots.

Clean the interior surfaces of the head and the spark gap using lens cleaning tissue or a Q-tip and ether, acetone or methanol. Wipe off any deposit on the electrodes, inner walls, mirrors and window. Allow the parts to dry off before assembling them. Follow the instructions in the laser manual on how to disassemble the spark gap and the laser head.

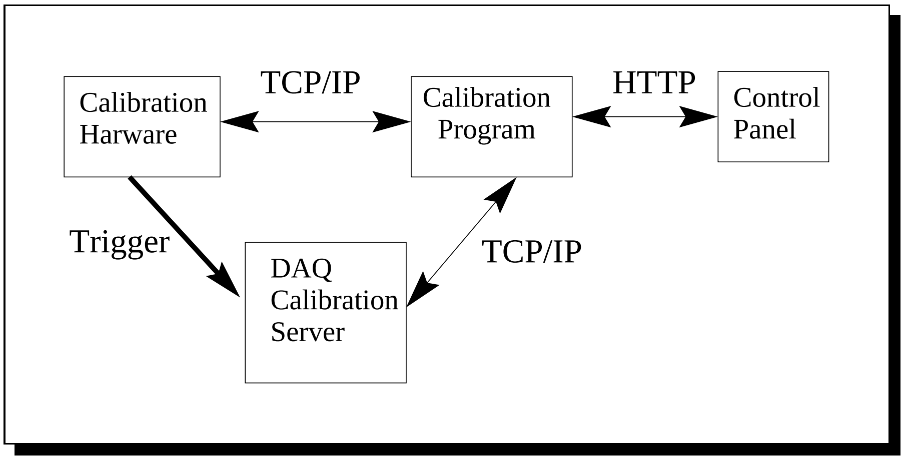
Purging

The spark gap is designed to operate with nitrogen gas. In order to fill the gap with nitrogen, a screw is provided on the spark gap which allows nitrogen to be purged through the gap to flush out the air. The screw is then tightened to allow the spark gap to be pressurized.

[7]This is prohibited by the LANL Hazard Control Plan

2.7 Known hardware problems

- DiCon FiberOptics Switch
  - Often, the device fails initialization. The initialization should be repeated until successful.
- Motion Controller MM3000
  - There is no hardware "home"("zero") on the filter wheel stage. The motorized stage should be replaced.
  - The controller does not execute commands according to its specifications. It might be the firmware problem (current version 2.3a) the chip might have to be replaced or something else. Purging the memory seemed to help. Purging can be done several thousand times before a repair is needed.
  - Programmable TTL I/O ports are not protected. Any surge on one of the lines will burn the CPU bus. Opto-electronic couplings should be used to protect the device.
- Pulse/Delay Generator
  - Fixed bit in the instrument status byte is wrong!. Status can not be checked although seems to execute commands correctly.
- GPIB-ENET Controller
  - The GPIB-ENET driver uses blocking input/output socket operations. Calibration program **will** hang if connection to GPIB-ENET is lost. The ESP-488 driver should be rewritten using synchronous I/O multiplexing with `select()`
- Laser
  - The stability of laser light output is difficult to establish. Instead, one uses the frequency of miss-fires. It was found that if laser miss-fires less than 2.5%, the operation is found to be acceptable.



3 Software manual.

Never include a comment that will help someone else understand your code.
If they understand it, they don't need you.

The "Calibration program" is a cgi-script written in C and is executed each time the "Start" button is pressed on the web-based "Control Panel". The calibration program gets all the settings via "POST" method and then processes the request. After the request is processed, the main program forks a child and finishes. This ensures the process is not aborted if the control panel is closed. The child does all the control functions on calibration hardware and communication with the DAQ system. Current hardware settings are "pushed" back to the Control Panel to be displayed.

Pressing the "Terminate" button with the correct password sends the "`USR2`" signal to the calibration child and terminates it. The child will also exit on any other unexpected signal. As a part of termination process, the child will try(!) to park the devices and close all open file/device descriptors.

3.1 GPIB-ENET Driver[5]

The ESP-488 (Engineering Software Package) software from National Instruments includes source code that performs **basic** IEEE 488 functions. The code contains functions for synchronous communication, device configuration, and status reporting.

There are some useful functions which are not included in ESP-488 function set.

ibclr()

`ibclr(ud)` sends the GPIB Selected Device Clear (SDC) message to the device described by `ud`. Clearing the device resets its internal functions to a default state. `ibclr()` sends the following series of messages:

MTA (board)
MSA (board) if applicable
UNL
MLA (device)
MSA (device) if applicable
SDC

Using ESP-488, SDC-message can be sent by `ibcmd()` function. (see appendix A for a list of *Multiline Interface Messages*.)

```
/*
 * @ - MTA0 ( My Talk Address 0) (Our GPIB-ENET is on PAD 0)
 * ? - UNL ( Unlisten)
 * # - MLA3 (My Listen Address 3) (Switch is on PAD 3)
 * \x04 - SDC (Selected Device Clear)
*/
  if(ibcmd(GpibEnet,"@?#\x04",4)&ERR)
    {
     fprintf(stderr,"GPIB: Could not Clear the Switch.\n");
     exit(1);
    }
```

iblock() and **ibunlock()**....

`iblock(ud)` is used to obtain exclusive access to a GPIB-ENET interface. The `iblock()` function blocks other processes from accessing the interface designated by `(ud)` while the lock is in effect. While locked, the software guarantees that subsequent calls made from privileged board are completed without interruption. The interface is released via an `ibunlock()` function call, made with the same descriptor. It is recommended that `ibunlock()` be used immediately after critical board accesses are made to the locked interface. Errors:

> ELCK Occurs if the GPIB-ENET board being locked is already locked by another process.

See appendix B for more complete description and realization of the function.

3.2 Writing an application program

This section discusses items one should include in an application program, general program steps.

Items to include:

- Include the GPIB header files (esp.h, espproto.h, gpibcnst.h,...)
- Check for errors after **each** ESP-488 function call
- Declare and define a function to handle GPIB errors. This function takes devices off-line and closes the application...

The following steps demonstrate how to write a generic application program using ESP-488 function set:

1. Acquire device descriptors with `ibfind()`
2. Set GPIB addresses of the devices with `ibconfig(ud,IbcPAD,#)` and if necessary secondary addresses with `ibconfig(ud,IbcSAD,#)`
3. Initialize GPIB controller
   - `ibconfig(ud,IbcSC,1)` Become System Controller
   - `ibconfig(ud,IbcSRE,1)` Assert REN GPIB line
   - `ibsic()` Assert IFC, make GPIB CiC, activate controller with ATN asserted
4. If necessary, lock devices with `iblock()`
5. Initialize devices
   - If necessary, clear devices using `ibcmd()` by sending the following message: UNT, UNL, MLA, MTA, SDC, UNL
   - If necessary, initialize (configure) internal functions of each device with `ibwrt()` command (or by any other appropriate means)
   - If necessary, set EOS-character with `ibconfig(ud,IbcEOSchar,0xHH)`
   - If necessary, terminate read on EOS or on EOI with `ibconfig(ud,IbcEOSrd,1) or ibconfig(ud,IbcEOT,1)`
6. Send commands to devices with `ibwrt()`. If you need to trigger a device send a GPIB "GET" message using `ibcmd()`
7. If necessary, wait for a measurement (or other event) with `ibwait()`
8. Conduct serial poll and check if the data is valid or a fault condition exists with `ibrsp()`. Some devices do not support serial poll, use device specific function.
9. Read data with `ibrd()` and check the number of bytes read.
10. Process the data
11. If you need anything else goto 6 (7)
12. Release the locks with `ibunlock()` and take the devices off-line with `ibonl(ud,0)`

Mnemonics	Bit	Hex	Type	Description
ERR	15	8000	dev, brd	GPIB error
TIMO	14	4000	dev, brd	Time limit exceeded
END	13	2000	dev, brd	END or EOS detected
SRQI	12	1000	brd	SQR interrupt received
RQS*	11	800	dev	Device requesting service
CMPL	8	100	dev, brd	I/O complete
LOK	7	80	brd	Lockout state
REM	6	40	brd	Remote State
CIC	5	20	brd	Controller-In-Charge
ATN	4	10	brd	Attention is asserted
TACS	3	8	brd	Talker
LACS	2	4	brd	Listener
DTAS[8]	1	2	brd	Device Trigger State
DCAS[8]	0	1	brd	Device Clear State

Table 2: Status word (ibsta) layout

Checking Status with Global Variables

Each ESP-488 function updates the global variables to reflect the status of the device or controller. The status word (`ibsta`), the error variable (`iberr`), and the count variable (`ibcntl`) contain useful information about the performance of an application program. The program should check these variables frequently. `gpibcnst.h` from ESP-488 contains the mnemonic constants for all global variables.

Status word `ibsta` contains information about the state of the GPIB hardware. `ibsta` is a 16-bit variable. A bit value of “1” indicates that a certain condition is in effect. A bit value of “0” indicates that the condition is not in effect. The table 2 shows the status word layout.

One can check a bit position in `ibsta` using its numeric value or its mnemonic constant.

```
if (ibsta & ERR) gpiberr();
or
if (ibsta &0x8000) gpiberr();
```

where `gpiberr()` is an error handling function.

When the **ERR** bit in the `ibsta` status word is set, a GPIB error has occurred. The error type is specified by the value in `iberr`. **The value `iberr` is meaningful as an error type only when the ERR bit is set, indicating that an error has occurred!!!** Table 3.2 lists possible error indications.

[8]The ESP-488 package does not contain complete functions for handling DCAS and DTAS. The actions one should take are described by the IEEE 488.2 standard and are device specific. Therefore, one must **clear these bits** programmatically when one develops routines to handle these events.

Mnemonics	Value	Description
EDVR[9]	0	Operating system error
ECIC	1	Function requires GPIB board to be CIC
ENOL	2	No Listeners on the GPIB
EADR	3	GPIB board not addressed correctly
EARG	4	Invalid argument to function call
ESAC	5	GPIB board not System Controller as required
EABO	6	I/O operation aborted
ECAP	11	No capability for operation
EBUS	14	GPIB Bus error
ELCK	21	Lock error
ECFG[10]	24	Board is configured differently than requested

Table 3: iberr descriptions

The count variable (`ibcntl`) is updated after each read, write, or command function. `ibcntl` is a 32-bit integer. If data or commands are being sent, the count variable indicates the number of bytes sent. If data is being read, the count variable indicates the number of bytes read.

3.3 DAQ Calibration Server

In principle, DAQ calibration server should not exist as a separate entity. It should be an integrated part of the DAQ system. However, it has a "separate life" and it is described in this section. The purpose of the server is to obtain current settings of the calibration hardware (laser ball, filter position) and supply it to the DAQ system to be saved as a part of the data stream.

Communication Protocol

Communication between DAQ calibration server and calibration control program (client) is performed by exchange of 3-byte words. The exchange is initialized by the client and

[9] If this error is reported, additional information is recorded in the `ibcntl`. If `ibcntl` is 0xe0 or greater, the error is returned from the ESP-488 software. Otherwise, the error is reported by `errno`.

ibcntl	value	meaning
INVALDEND	0xe7	DATA END header was non-zero
INVALHDRT	0xe9	Invalid header type rcvd
INVALHDRS	0xea	Invalid header state
NODESCRPT	0xeb	No more GPIB descriptors left
INVALROLE	0xec	Invalid role (Not BOARD or DEVICE) requested.

[10] Occurs only when opening a (any) descriptor with `ibfind()`. It indicates that the GPIB-ENET is already online and configured. The returned descriptors are valid. It is a warning that the controller itself is configured differently than requested(?). **It seems that this error message will be generated when opening a descriptor, unless the GPIB-ENET has the default configuration. We consider this as a bug in the ESP-488 driver. Nevertheless, one should make sure that no-one else is using the GPIB-ENET before reconfiguring it.**

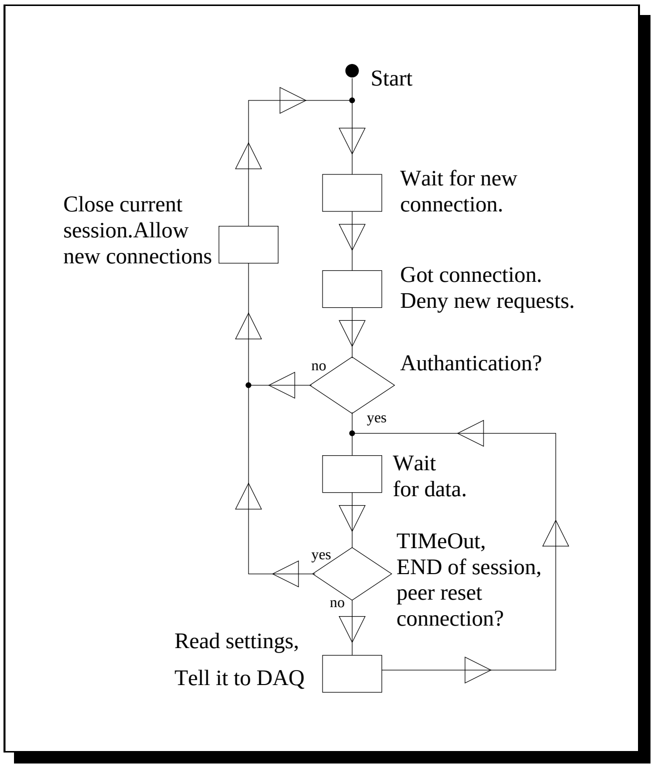


Figure 5: DAQ calibration server flow-chart

performed when reconfiguring calibration hardware. After obtaining new data, server reports current settings to DAQ and acknowledges to the client. First two bytes carry information about current laser ball and filter wheel setting (in that order). The third byte informs the server when to expect a new data set. If within this time interval the server does not receive any instructions from client, it assumes that the connection is lost and resets the session. Client, on the other hand, has to supply correct timeout interval in the third byte and get an acknowledgment from the server that the instruction has been received. If the client does not get a confirmation within a preset time interval (currently 10 seconds), it assumes that the connection to DAQ server is lost and shuts itself down. This scheme allows graceful recovery from all known types of network-related fault conditions.

Encoding: BWT

B=(laser Ball number)+48
W=(Wheel position in degrees)/10 +48
T=(Timeout in minutes) +48

Special words:

EOT = End Of Transmission
CALL SIGNS

3.4 Morse Code

Before the DAQ server was implemented, the laser ball and filter position were "transmitted" to the DAQ system and were encoded into the data stream as a special sequence of triggers generated by the Pulse/Delay generator. The sequence became known as "the Morse code". Each message consists of four digits (two for laser ball setting and two for filter position) separated from each other and from the main laser data by five delimiters. Because the laser can operate at 20 Hz at the most, the delimiters can be recognized as series of triggers at a high rate (more than 100 Hz). Since the laser ball numbers range from 1 to 30[11] and filter positions — from 0 to 35[12], the "message" was converted to the base-6 system before the transmission. To ensure high fidelity of the "channel", the following error-correcting algorithm was employed. Assuming low probability of loosing more than one trigger pulse per digit in a message, for each digit of value x $3(x + 1)$ trigger pulses are sent to the DAQ at about 10 Hz. During data processing, the number of registered triggers in a digit message is forced to the nearest integer multiple of 3 allowing for errors of ± 1 to be corrected. Furthermore, the first delimiter is longer in time and is at a higher frequency than the others, providing the beginning of the Morse code.

Besides its calibration setting information content needed to recognize the wheel and ball changes in the data, the Morse code pulses are used as "random" triggers in PMT noise estimation (See section 4.5), making it a very important part of the calibration process.

[11]Laser ball numbers from 31 through 35 are reserved and when used, the filter wheel field may have a different meaning.

[12]Filter wheel is rotated in $10°$ quanta

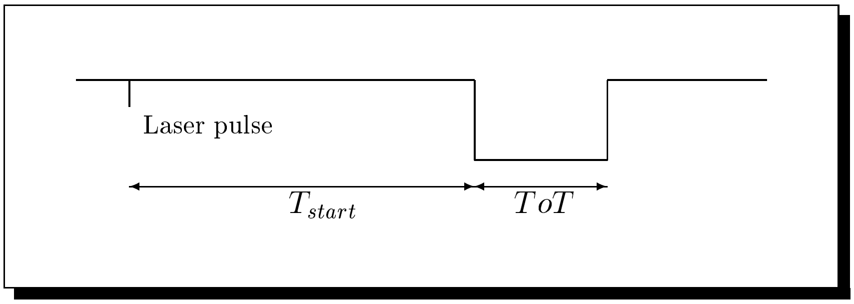


Figure 6: concept

4 Analysis

4.1 Slewing calibration

The slewing generation method can be explained with the help of the figure 6. This diagram is Hi/Lo threshold independent and is needed to serve as a conceptual example. Slewing curve is the function of T_{start} vs ToT. Note, that T_{start} includes the time propagation of the pulse in water and in the optical fiber. After taking it into account, which effectively shifts the curve up or down, it will become the complete timing calibration (including T-pedestal).

This scheme is applied independently for both threshold levels. In the situation where both Hi and Lo ToT are available, $T_{start}Hi$ and $T_{start}Lo$ are different, but after slewing correction with appropriate curve will result in the same time of the pulse origination. Conversion from $T_{start}Hi$ to $T_{start}Lo$ is not required.

In the case of pre-pulsing, LoToT will be artificially extended and $T_{start}Lo$ artificially early. Since the process of pre-pulsing is stochastic, one can not apply a correction, therefore the use of HiToT is preferred. Thus, switching to HiToT should be done as soon as it makes sense and the HiToT slewing becomes narrow. Typically, it happens when $HiToT \geq 75, 100$ counts.

The case when pre-pulsing is so strong that Hi threshold is also affected, can not be handled because of stochastic nature of the problem. Such a pulse will be treated as a stronger one than it is. The hope is that it will never happen due to relatively high level of the Hi threshold ~ 5 pe.

General slewing generation method:

1. Histograms Tstart vs ToT are generated with 25 count bin size in ToT and 1 count bin size in Tstart.

2. During histograming only 2 and 4 edge events are considered. NO edge finding is applied except for polarity check.

3. In each slice of ToT the peak of Tstart is found. The ± 5 counts in Tstart region around the peak is used in Gaussian fit to estimate the most probable Tstart for that ToT bin (slice). The peak search is considered successful if the fitting region contains more than 100 entries.

4. Most probable values of all successful peak searches are fed into polynomial fitting routine. Currently 3-rd order polynomial is used.

5. The slewing coefficients with the fitted ToT region (ToTleft ToTright) are output. ToTleft is defined as the left edge of the left most ToT slice with successful peak search, ToTright is defined as the right edge of the right most ToT slice. Gaps are ignored.

6. Slewing error distribution is calculated. Namely, the deviation of the Tstart from the slewing curve between ToTleft and ToTright is histogramed. This is used for diagnostic purposes, the mean and rms are not output, but can be viewed in PAW.

7. If a PMT gets slewing curve from more than one ball, the selection is made according to ToTright.

4.2 Consistency Check

Before the time-pedestals can be determined the speed of light in water, fiber optic delays and coordinates of light sources and PMT's have to be found or be given.

Only a typical index of refraction of water is found in tables, but for our short pulses in a dispersive medium (water) care should be taken to use the group velocity of light. If, also, only nominal coordinates of the laser balls are known, because the anchor strings are loose (or for any other reason), and since fiber optic delays are only nominally zero, these input parameters have to be evaluated on the basis of the calibration data itself assuming only that the positions of all PMTs are known from the survey and all TDCs operate with the same conversion (in our case 2 counts/nsec). Please, refer to [11] for the description the method how all these problems can be resolved.

Inasmuch as, pedestals should not depend on calibration procedure, in particular, they should not depend on laser ball used to determine them, pedestal mismatch distribution becomes a useful diagnostic tool. The method of improved fiber optic delay and speed of light determinations can be found in [12].

Note, that time-pedestals themselves are never needed. The raw slewing curve is shifted by the fiber optic delay and propagation time in water to provide the final complete timing calibration including time-pedestal.

4.3 PE calibration using Occupancy method

ToT-to-PE conversion is split into two steps [13]:

1. At low light levels, calibrate using occupancy method.

2. Extend to higher light levels using known attenuation properties of the filter wheel.

Low light level calibration

measure PE as a function of ToT directly

Occupancy method is based on the following assumptions:

- number of photo-electrons produced at PMT's photocathode obeys Poisson distribution: $P(n, \lambda) = \frac{\lambda^n}{n!} e^{-\lambda}$
- constant light intensity source is used in calibration.

measure PE as a function of ToT directly

The probability that **at least one** photo-electron was produced when the PMT's photocathode was illuminated is called occupancy (η) and is given by the following expression:

$$\eta = P(n > 0, \lambda) = 1 - P(n = 0, \lambda) = 1 - e^{-\lambda}$$

where λ — is the mean number of the photoelectrons observed. Occupancy can be easily measured[13] and the mean number of photoelectrons can be inferred:

$$\eta = \frac{\#\ of\ observed\ pulses}{total\ \#\ laser\ pulses}, \qquad \lambda = -\ln(1 - \eta)$$

However, this method is applicable only at relatively low light levels ($\lambda \leq 2$), because at high light levels a small measurement error in η will lead to an exponentially big error in λ:

$$\Delta\lambda = \frac{1}{1 - \eta}\Delta\eta = e^{\lambda}\Delta\eta$$

High light level calibration

At high light levels we assume the following:

- mean number of produced photo-electrons is proportional to the light intensity at the photocathode.
- constant light intensity source is used in calibration.

Under these assumptions and if the transmittance of the filter wheel T is known, then:

$$\lambda = a \cdot T$$

where a is some coefficient. This coefficient can be found by fitting a straight line to the λ vs T plot for low light levels when λ can be determined by the occupancy method. a varies from PMT to PMT.

$$\chi^2 = \sum_i (\lambda_i - aT_i)^2 \quad \Rightarrow \quad a = \frac{\sum_i \lambda_i T_i}{\sum_i T_i^2}$$

Thus, given the transmittance T, average number of photoelectrons λ can be determined at high light levels and the ToT-to-PE conversion can be extended to the region beyond $\lambda \geq 2$.

[13]See section 4.5 for some comments on this problem

4.4 Filter Wheel Calibration

As it was mentioned in section 4.3, it is important to know the transmission properties of the filter wheel to be used in the occupancy method. The method of filter wheel calibration is based on the same assumptions as the occupancy method in section 4.3 with the following supposition:

For any two sufficiently close transmittances of the filter wheel (T_1 and T_2) there exist a PMT in the pond for which the occupancy method can be applied on both light intensities.

Since for such a PMT the occupancy method is valid, the average number of photoelectrons (λ_1 and λ_2) can be determined and:

$$\lambda = a \cdot T \quad \Rightarrow \quad \frac{T_2}{T_1} = \frac{\lambda_2}{\lambda_1}$$

This line of arguments can be used to relate T_2 with T_3, T_3 with T_4 and so on, leading to the restoration of the levels of transmittance for the whole filter wheel. Without any doubt, one can always set $T_0 = 1$.

4.5 Dynamic Noise Suppression

Procedures, described in sections 4.3 and 4.4, rely on correct knowledge of PMT occupancy η. The total number of light pulses sent to the PMT is known exactly, but problems arise when trying to estimate the number of pulses registered by the PMT. If noise level is relatively high, noise (random) hits will increase the apparent occupancy. To cope with the problem, the following line of arguments is followed. An event, registered by a PMT, could have come from a laser pulse or from a noise hit which are assumed to be uncorrelated. Thus, the probability of observing a hit $P(any)$ is:

$$P(any) = P(laser + noise) = P(laser) + P(noise) - P(laser) \cdot P(noise)$$

where $P(laser)$ – is probability of observing a pulse due to laser (true occupancy η), $P(noise)$ – is probability of observing a pulse due to noise. $P(any)$ can be measured by sending laser pulses to a PMT (as described in section 4.3) and $P(noise)$ can be estimated by sending "random" triggers to the data acquisition system (see section 3.4), then

$$\eta = P(laser) = \frac{P(any) - P(noise)}{1 - P(noise)}$$

This scheme was applied for low light level PE calibration (section 4.3) and for the filter wheel calibration (section 4.4).

4.6 Extrapolation

A scheme of estimation of slewing correction for ToT's outside of calibrated range (extrapolation) must model the function beyond the known points by some plausible functional form. In the case of slewing, the functional shape depends on the discriminator threshold level, amplification coefficients, gains of PMTs, wiring and so on. Instead of trying to take all the unknown parameters into account and putting together a physical model of the slewing we have built a statistical one taking the following approach.

We believe that all PMT channels (PMTs themselves and electronic boards) were designed and manufactured to meet common characteristics. Therefore, the study of the channels' responses (calibration) can be viewed as a multiple (about 700 times) measurement of a singe function: T_{start} vs ToT. The fact that the curves obtained for different channels are slightly different can be attributed to the "manufacturing imperfections", and can be treated as such. Here, we mean that the channels differ only due to unavoidable uncontrollable reasons such as spread of characteristics of electronic components and/or actual slewing measurement errors. Thus, we view all slewing curves for all PMTs as particular realizations of the same random function. All slewing curves together form one slewing function family.

The power of this method is that the extrapolation model is learned on the example of the "fortunate" channels, and the obvious extrapolation test will be automatically satisfied: cut the existing long curve, extrapolate it and compare with the existing data.

For details of implementation please see memo [9] describing the extrapolation method used to extend slewing curves for the HiToT calibration. The method is based on random function's approach. The random function framework is set and carried out to the first order of canonical expansion. The statistical error of the extrapolation is shown to be within 1.4 counts at the end of the range.

For a brief review of the notion of random functions please refer to memo [10].

4.7 PE calibration using ADC

ToT-to-PE conversion can be performed using ADC measurements. Please refer to [14] for methods of how ADC scale is converted into the PE scale using single photo-electron peak. Linearity of ADC is then used for ToT-to-PE conversion. The method relies on the linearity of the PMT-electronics channel and starts saturating at about 75-100 Pe.

4.8 Absolute energy calibration.

Absolute energy calibration can be achieved with the help of through-going μ-ons, fitting their track and calculating energy deposit in the detector.

A Multiline Interface Messages

Hex	Bin	Dec	ASCII	Msg	Hex	Bin	Dec	ASCII	Msg
00	00000000	0	NUL		20	00100000	32	SP	MLA0
01	00000001	1	SOH	GTL	21	00100001	33	!	MLA1
02	00000010	2	STX		22	00100010	34	”	MLA2
03	00000011	3	ETX		23	00100011	35	#	MLA3
04	00000100	4	EOT	SDC	24	00100100	36	$	MLA4
05	00000101	5	ENQ	PPC	25	00100101	37	%	MLA5
06	00000110	6	ACK		26	00100110	38	&	MLA6
07	00000111	7	BEL		27	00100111	39	’	MLA7
08	00001000	8	BS	GET	28	00101000	40	(	MLA8
09	00001001	9	HT	TCT	29	00101001	41	)	MLA9
0A	00001010	10	LF		2A	00101010	42	*	MLA10
0B	00001011	11	VT		2B	00101011	43	+	MLA11
0C	00001100	12	FF		2C	00101100	44	,	MLA12
0D	00001101	13	CR		2D	00101101	45	-	MLA13
0E	00001110	14	SO		2E	00101110	46	.	MLA14
0F	00001111	15	SI		2F	00101111	47	/	MLA15
10	00010000	16	DLE		30	00110000	48	0	MLA16
11	00010001	17	DC1	LLO	31	00110001	49	1	MLA17
12	00010010	18	DC2		32	00110010	50	2	MLA18
13	00010011	19	DC3		33	00110011	51	3	MLA19
14	00010100	20	DC4	DCL	34	00110100	52	4	MLA20
15	00010101	21	NAK	PPU	35	00110101	53	5	MLA21
16	00010110	22	SYN		36	00110110	54	6	MLA22
17	00010111	23	ETB		37	00110111	55	7	MLA23
18	00011000	24	CAN	SPE	38	00111000	56	8	MLA24
19	00011001	25	EM	SPD	39	00111001	57	9	MLA25
1A	00011010	26	SUB		3A	00111010	58	:	MLA26
1B	00011011	27	ESC		3B	00111011	59	;	MLA27
1C	00011100	28	FS		3C	00111100	60	<	MLA28
1D	00011101	29	GS		3D	00111101	61	=	MLA29
1E	00011110	30	RS		3E	00111110	62	>	MLA30
1F	00011111	31	US		3D	00111111	63	?	UNL

Hex	Bin	Dec	ASCII	Msg	Hex	Bin	Dec	ASCII	Msg
40	01000000	64	@	MTA0	60	01100000	96	`	MSA0,PPE
41	01000001	65	A	MTA1	61	01100001	97	a	MSA1,PPE
42	01000010	66	B	MTA2	62	01100010	98	b	MSA2,PPE
43	01000011	67	C	MTA3	63	01100011	99	c	MSA3,PPE
44	01000100	68	D	MTA4	64	01100100	100	d	MSA4,PPE
45	01000101	69	E	MTA5	65	01100101	101	e	MSA5,PPE
46	01000110	70	F	MTA6	66	01100110	102	f	MSA6,PPE
47	01000111	71	G	MTA7	67	01100111	103	g	MSA7,PPE
48	01001000	72	H	MTA8	68	01101000	104	h	MSA8,PPE
49	01001001	73	I	MTA9	69	01101001	105	i	MSA9,PPE
4A	01001010	74	J	MTA10	6A	01101010	106	j	MSA10,PPE
4B	01001011	75	K	MTA11	6B	01101011	107	k	MSA11,PPE
4C	01001100	76	L	MTA12	6C	01101100	108	l	MSA12,PPE
4D	01001101	77	M	MTA13	6D	01101101	109	m	MSA13,PPE
4E	01001110	78	N	MTA14	6E	01101110	110	n	MSA14,PPE
4F	01001111	79	O	MTA15	6F	01101111	111	o	MSA15,PPE
50	01010000	80	P	MTA16	70	01110000	112	p	MSA16,PPD
51	01010001	81	Q	MTA17	71	01110001	113	q	MSA17,PPD
52	01010010	82	R	MTA18	72	01110010	114	r	MSA18,PPD
53	01010011	83	S	MTA19	73	01110011	115	s	MSA19,PPD
54	01010100	84	T	MTA20	74	01110100	116	t	MSA20,PPD
55	01010101	85	U	MTA21	75	01110101	117	u	MSA21,PPD
56	01010110	86	V	MTA22	76	01110110	118	v	MSA22,PPD
57	01010111	87	W	MTA23	77	01110111	119	w	MSA23,PPD
58	01011000	88	X	MTA24	78	01111000	120	x	MSA24,PPD
59	01011001	89	Y	MTA25	79	01111001	121	y	MSA25,PPD
5A	01011010	90	Z	MTA26	7A	01111010	122	z	MSA26,PPD
5B	01011011	91	[	MTA27	7B	01111011	123	{	MSA27,PPD
5C	01011100	92	\	MTA28	7C	01111100	124	|	MSA28,PPD
5D	01011101	93	]	MTA29	7D	01111101	125	}	MSA29,PPD
5E	01011110	94	^	MTA30	7E	01111110	126	~	MSA30,PPD
5F	01011111	95	_	UNT	7F	01111111	127	DEL	

Message Definitions

DCL	Device Clear	PPE	Parallel Poll Enable
GET	Group Execute Trigger	PPU	Parallel Poll Unconfigure
GTL	Go To Local	SDC	Selected Device Clear
LLO	Local Lockout	SPD	Serial Poll Disable
MLA	My Listen Address	SPE	Serial Poll Enable
MSA	My Secondary Address	TCT	Take Control
MTA	My Talk Address	UNL	Unlisten
PPC	Parallel Poll Configure	UNT	Untalk
PPD	Parallel Poll Disable		

The *Multiline Interface Command Messages*[6] table is organized into four groups. The first group of columns (hex 00-1F) represents the primary GPIB addresses. The next group (hex 20-3F) tells corresponding listen (MLA) addresses. The listen address of a device is formed by adding hex 20 to the GPIB primary addressed. Again, move to the next group of columns (hex 40-5F) for the corresponding talk (MTA) addresses. One can form the talk address of a device by adding hex 40 to the GPIB primary address. The last group of columns (hex 60-7F) represent the secondary GPIB address commands. A secondary address itself can be in the range of 0 to 30. To form a secondary address command add 96 decimal (or hex 60) to the secondary address. One can address a device with a secondary address by sending the primary GPIB address then the corresponding secondary address.

The miltiline messages are grouped in unaddressed (universal and addressed. Unaddressed commands (hex 10-1F) are send to all devices on the bus. An addressed command (hex 00-0F) is send to one or more specified devices. One should precede this command with the addresses of the devices.

The GPIB Controller uses two special commands to clear the bus of Talkers and Listeners before assigning new Talkers and Listeners. These commands are Untalk and Unlisten. Untalk (UNT) unaddresses the current Talker. The Untalk command is a command of convenience since addressing one Talker unaddresses all others. The Unlisten (UNL) command unaddresses all current Listeners on the bus. You must use the UNL command to guarantee that you address only one desired Listener.

A Multiline Interface message can be send using `ibcmd()` command. The general sequence is the following:

UNT	everybody Untalk, listen...
UNL	Unlisten (assure no listeners on the bus)
MTA	My talk address (so every one knows who is talking, usually Controller)
MLA	Listen address (so you know, I am talking to you)
YYY	Letters representing to a message
UNL	Unlisten (end of transmission)

B iblock() and ibunlock()[7]

The OPCODEs (first byte in the 12 byte command packet sent to the GPIB-ENET) for IBLOCK is 0x11 and IBUNLOCK is 0x21. The remaining 11 bytes of the command block are unused, but should be set to 0 to maintain upward compatibility with future GPIB-ENET firmware releases.

The network activity for each of these commands is similar to the CONFIG command. You send the packet (12 bytes) and you get status back (16 bytes). Calls which are disallowed while a LOCK is in effect behave depending on the BLOCK bit configured in the opening of the descriptor. There are two constants defined in esp.h, BDFLAGS and DEVFLAGS. In both of these definitions, there's the word BLOCK. If this is the setting, a call which is not allowed due to a LOCK in effect will BLOCK INDEFINITELY until the lock is released. If you remove BLOCK from the definition, a call failing to execute due to a LOCK in effect will return immediately with an ELCK error in the GPIB error portion of the 16 byte status block.

Here are some guidelines to the IBLOCK functionality: Device level network descriptor: The pad/sad associated with this descriptor is prevented from being accessed by ANY other DEVICE level connection. While ANY device level lock is in effect NO board level calls are allowed.

Board level network descriptor: This is the MOST powerful. When a board level lock is in effect, basically NOTHING else is allowed in the GPIB-ENET on ANY other connection (board or device level) until the lock is released.

To implement everything said one has to follow these steps:

esp.h In `#define BDFLAGS` and `#define DEVFLAGS`: remove `BLOCK` to allow immediate return with **ELCK** error if the lock is in effect.

esp.h In supported GPIB-ENET opcodes section add

```
#define EG_LOCK   0X11
#define EG_ULOCK  0X21
```

esp.h we will use `struct egtype6` to implement iblock/ibunlock.

espproto.h Add the function prototypes

```
int16 iblock(GPIBDESCRIPTOR Descriptor);
int16 ibunlock(GPIBDESCRIPTOR Descriptor);
```

esp.c Add the functions iblock() ibunlock()

```
int16 iblock(GPIBDESCRIPTOR Descriptor)
{
  TYPE6 EgLock;            /* The same structure as for RSP RPP */
  NETDESCRIPTOR Fd=GPIBDescriptors[Descriptor].Connection;

  memset(&EgLock,0,sizeof(TYPE6));           /* Clear all bytes */
  EgLock.op=EG_LOCK;         /* USE EG_ULOCK for ibunlock() */
  CmdIoctl(Fd,(int8 *)&EgLock,sizeof(TYPE6));
  return ibsta;
}
```

C Statistical error of occupancy method.

In order to address the question of accuracy of the occupancy method we suppose that n shots of the laser beam were sent, out of which m were detected by the PMT. Occupancy η is estimated as:

$$\eta = \frac{m}{n}$$

The question of the accuracy of this estimation is that of confidence interval. Since the probability of a PMT registering a signal in a shot is equal to η, then the number of detected pulses m is distributed according to binomial distribution:

$$P_\eta(m,n) = C_n^m \eta^m (1-\eta)^{n-m}, \quad C_n^m = \frac{n!}{m!(n-m)!}$$

The upper bound of the confidence interval η_{upper} corresponding to the significance α (we choose α to be at least 95 %) is defined as such, that probability of detecting $k \leq m$ pulses is less than $(1-\alpha)$:

$$P_{\eta_{upper}}(k \leq m, n) = \sum_{k=0}^{m} C_n^k \eta_{upper}^k (1-\eta_{upper})^{n-k} \leq 1-\alpha$$

Correspondingly, lower bound η_{lower} is such that

$$P_{\eta_{lower}}(k \geq m, n) = \sum_{k=m}^{n} C_n^k \eta_{lower}^k (1-\eta_{lower})^{n-k} \leq 1-\alpha$$

or

$$\sum_{k=0}^{m-1} C_n^k \eta_{lower}^k (1-\eta_{lower})^{n-k} \geq \alpha$$

If the required relative error on the value of occupancy is $\delta(\eta) = \frac{\Delta\eta}{\eta}$ then η_{upper} should be no greater than $(1+\delta(\eta))\eta$ and η_{lower} should be no less than $(1-\delta(\eta))\eta$. Thus, for given number of laser shots and number of registered pulses we can check if the required accuracy is met. We require the accuracy $\delta(\eta)$ to be such as to provide $q = \frac{\Delta\lambda}{\lambda} = 0.01$ relative error of the PE determination:

$$\lambda = -\ln(1-\eta)$$

$$\Delta\lambda = \frac{1}{1-\eta}\Delta\eta$$

$$q = \frac{\Delta\lambda}{\lambda} = -\frac{\Delta\eta}{(1-\eta)\ln(1-\eta)}$$

Thus, $\delta(\eta) = -q\frac{1-\eta}{\eta}\ln(1-\eta)$ and we look for the range of occupancy levels η for which the following criteria are satisfied ($n = 2000$ laser beam shots sent):

for $\eta < 0.5$:

$$\sum_{k=0}^{\eta * n} C_n^k \eta_{upper}^k (1-\eta_{upper})^{n-k} \leq 1-\alpha, \quad \eta_{upper} = (1+\delta(\eta))\eta$$

for $\eta > 0.5$

$$\sum_{k=0}^{\eta * n-1} C_n^k \eta_{lower}^k (1-\eta_{lower})^{n-k} \geq \alpha, \quad \eta_{lower} = (1-\delta(\eta))\eta$$

The results of these tests for a list of η's are as follows:

η	$\delta(\eta)$	pass/fail
0.1	0.009	f
0.2	0.009	p
0.3	0.008	p
...	...	...
0.8	0.004	p
0.9	0.002	f

Therefore, the allowed range of occupancies to be used is $0.2 < \eta < 0.8$. For the phototube noise measurements ($\eta \sim 0.03$) the requirements are eased: $q = 0.1$. We also verify that using 60000 random "shots" is just enough to reach the goal. (The requirement of $q = 0.01$ is met at 66 % confidence level only.)

D Threshold effect on the occupancy measurement.

The problem we address in this section is that of a finite threshold level which phototube signal should cross in order to be "seen" by the electronics. The presence of the finite threshold leads to an under estimation of the occupancy and we intend to investigate by how much. If the final effect on the number of PEs is bigger that q than we need to correct for this effect.

The assumption we make here is that electron multiplication in each stage is a statistical process with some average gain. This assumption is similar to that of occupancy method and is based on proposition that probability of emitting an "amplified" electron does not depend on other electrons being emitted. Then, the number of electrons on the output of the first amplification stage obeys Poisson distribution with average gain $g = 5.3$[14] (for one electron on the input):

$$P(n, a) = \frac{a^n}{n!} e^{-a}$$

It is the number of electrons on the output of the first stage that dominates the total on the output of the entire cascade. Thus, the ability of the signal to cross the discriminator

[14]The typical gain of the entire cascade is $2 \cdot 10^7$ and consists of 10 stages. Thus, the gain of a single stage is $(2 \cdot 10^7)^{\frac{1}{10}} = 5.3$. (see Milagrito NIM paper for the measured gain of PMT's at the operated voltage.)

threshold level is governed by the fluctuations on the first stage. If less that $g/4 \sim 1$ electrons are emitted the threshold level will not be crossed. (The discriminator threshold is set to about $\frac{1}{4}$ of 1 PE in amplitude.) The probability of that happening given one electron (1PE) on the input is:

$$\beta = P(k \leq 1, a = g) = \sum_{k=0}^{1} \frac{a^k}{k!} e^{-a}$$

For $a = 5.3$ $\beta = 3.0 \cdot 10^{-2}$. (For two electrons on the input $a = 2g = 10.6$ $\beta = 2.5 \cdot 10^{-4}$ and can be neglected.)

The probability of observing a signal is decreased by the amount that some 1 PEs will be missed:

$$occ = 1 - P_\lambda(0) - P_\lambda(1)\beta$$

$$\eta = 1 - e^{-\lambda} - \beta\lambda e^{-\lambda}$$

The magnitude of the correction $\beta\lambda e^{-\lambda}$ has to be compared with the required accuracy on η which leads to the direct comparison of β and q.

Thus we conclude that the systematic error in occupancy is comparable to the statistical one, therefore, for the desired accuracy of PE determination (few per cent) the threshold effect can be neglected. If one knows the probabilities β for each tube and wishes to make the correction, then it can be done with the following approximate formula:

$$\lambda = -\frac{1}{1-\beta} \ln(1 - \eta)$$

Also note, the filter wheel calibration is immune from this problem because it is based on the ratio of the λ's for a given PMT and the $\frac{1}{1-\beta}$ factor cancels.

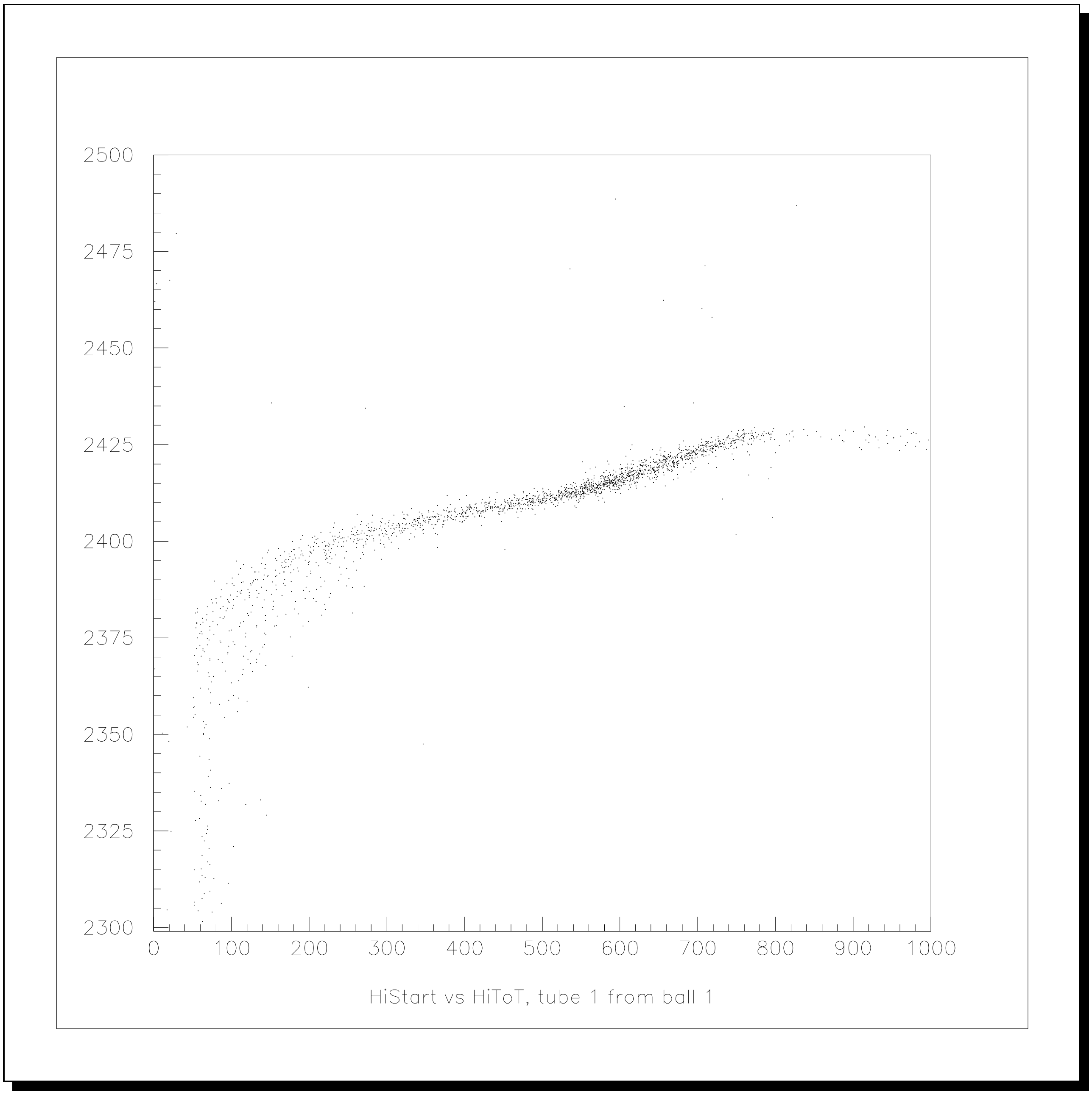
E Format of calibration database

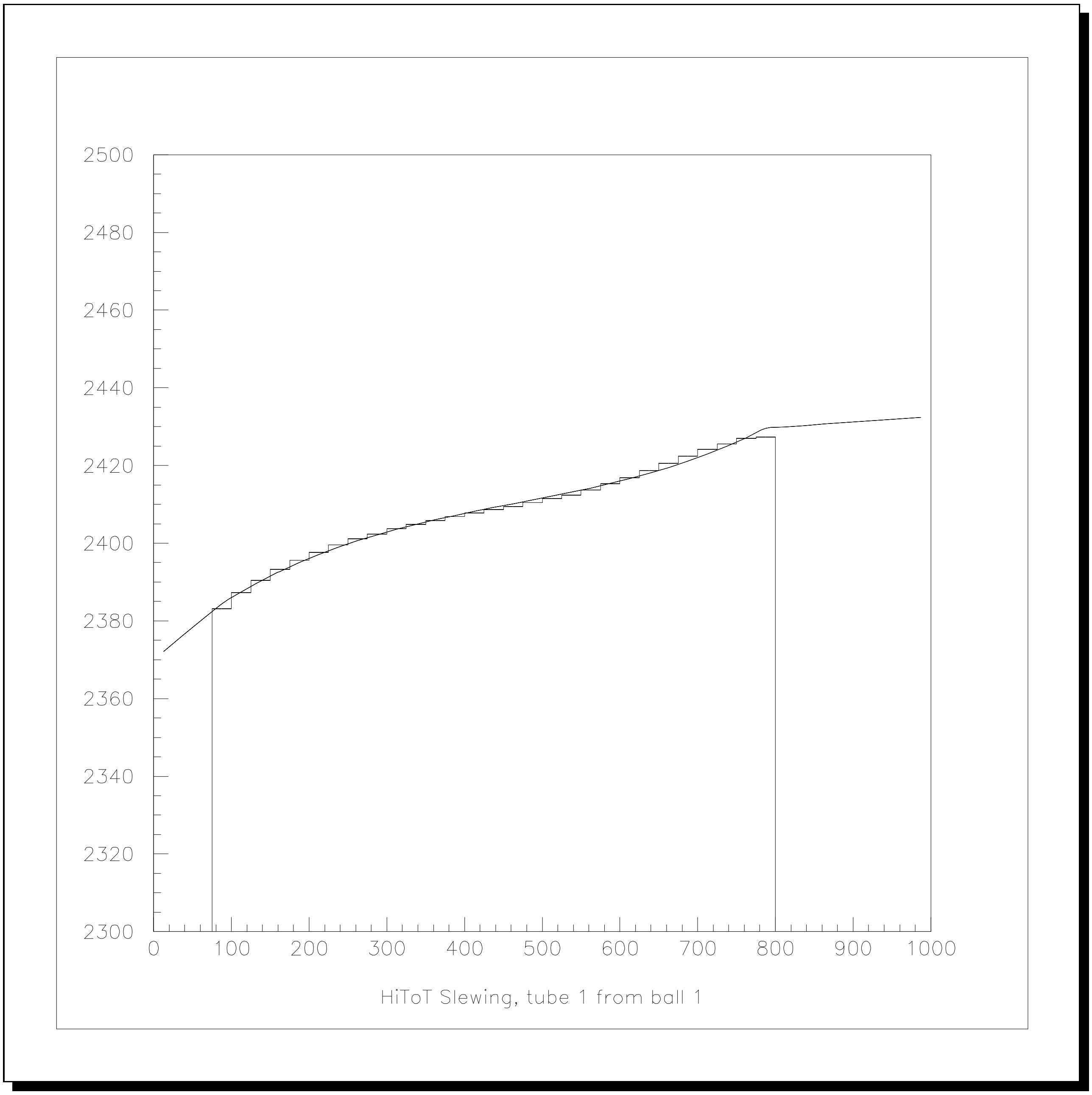
The calibration constants are saved in a text file, one line per PMT. The record format is as follows:

int	int	int	int	int	float	float	...
pmt	ball	Juldate	Left	Right	a0	a1	...

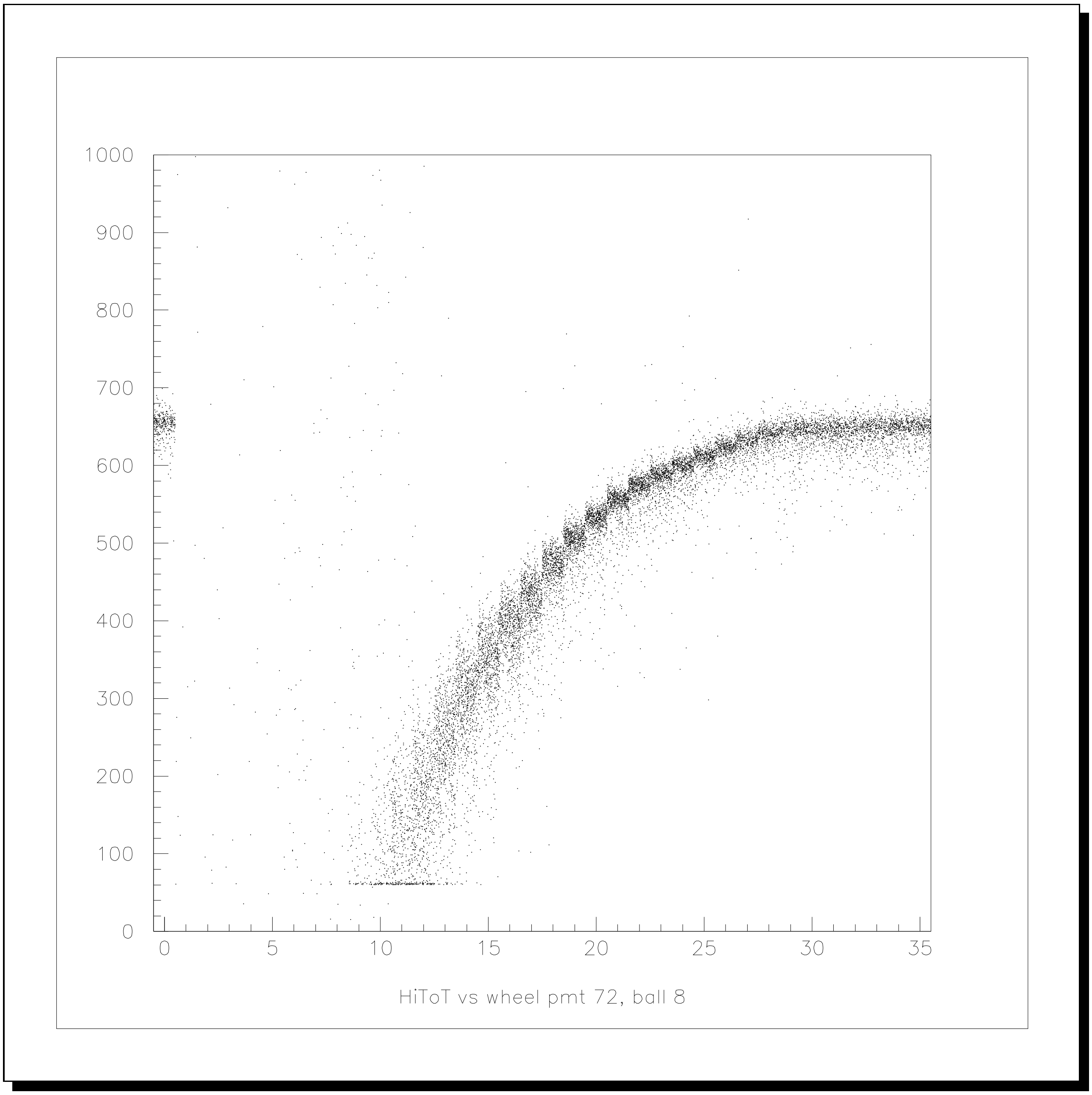
The meaning of the first three columns is obvious, and is needed for bookkeeping purposes. “Left” and “Right” represent the bounds of the fitted region. “a0”, “a1”,... are polynomial coefficients. Last two columns are treated as slopes of linear extrapolation to the left and to the right of the fitted region respectively. The format is flexible in the sense that degree of the polynomial to be used is determined according to the number of coefficients in the record. Different PMTs may have polynomials of different degrees.

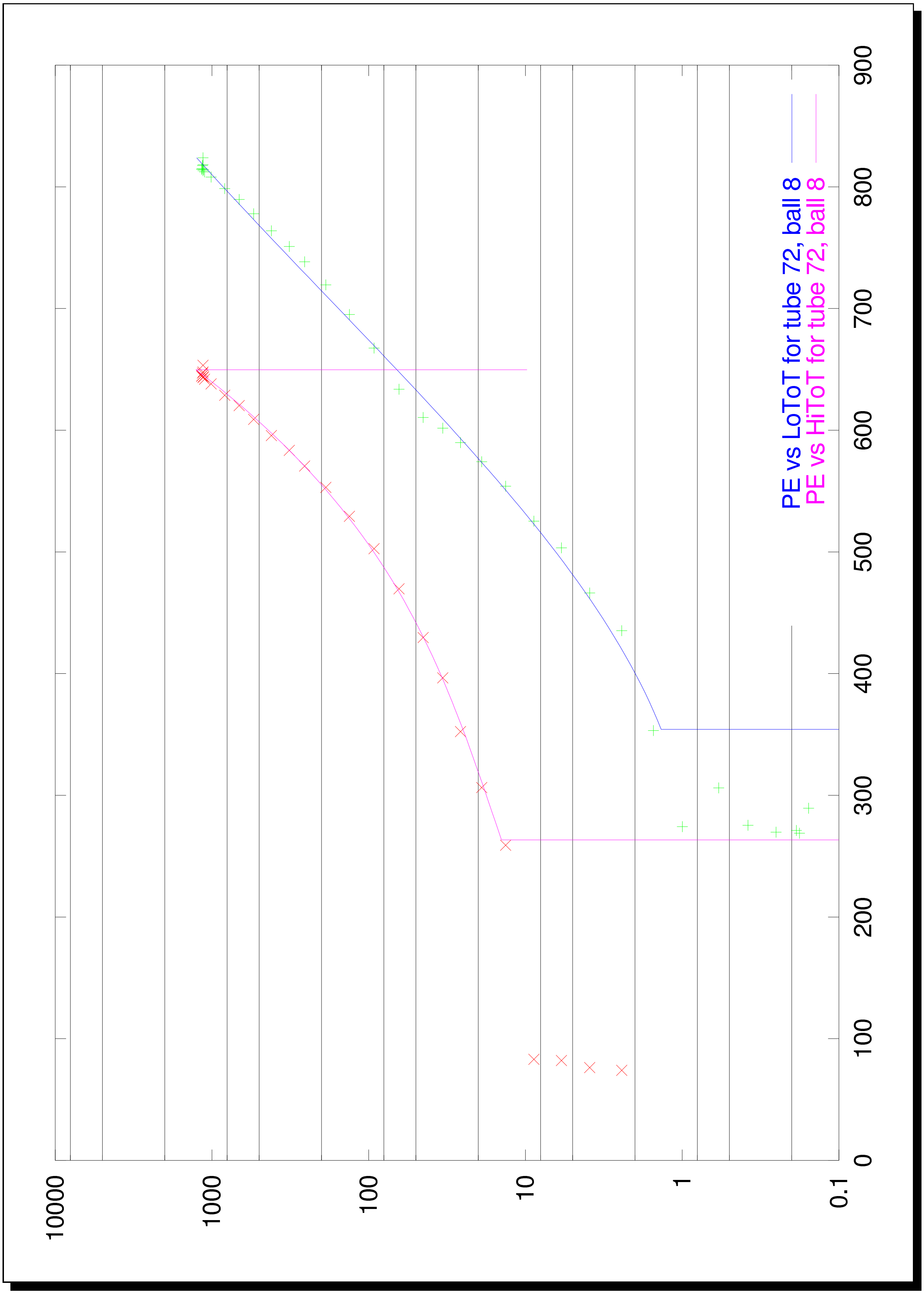
F Example of slewing.




2500
2480
2460
2440
2420
2400
2380
2360
2340
2320
2300
0
100
200
300
400
500
600
700
800
900
1000
HiToT Slewing, tube 1 from ball 1

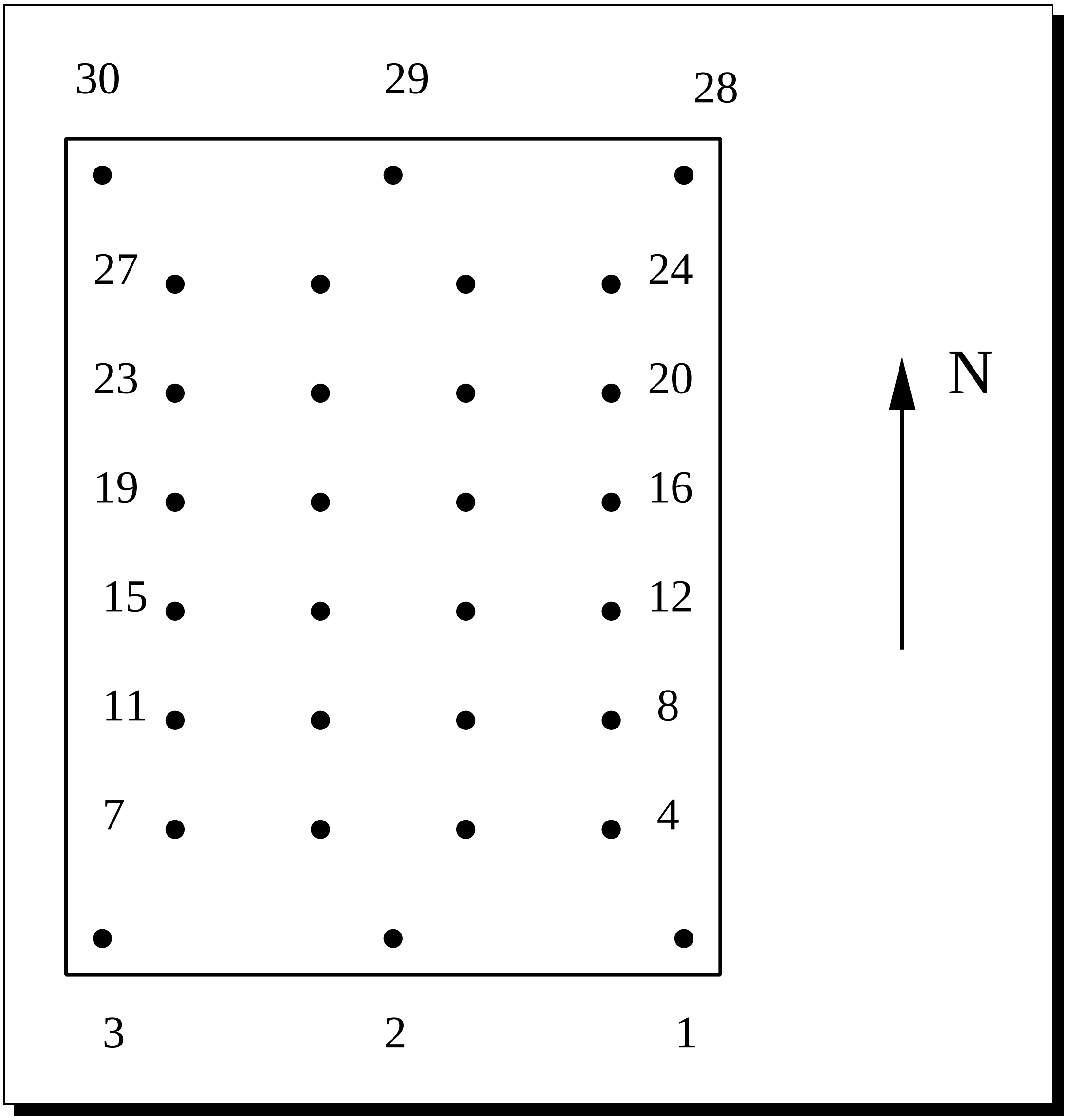
G Example of PE calibration.





PE vs LoToT for tube 72, ball 8
PE vs HiToT for tube 72, ball 8
10000
1000
100
10
1
0.1
0
100
200
300
400
500
600
700
800
900

H Laser ball locations.



I Company list

National Instruments

tel: (512) 794-0100, (800) 433-3488. fax:
ESP-488 Source Code Routines for GPIB-ENET p/n 777059-01
ESP-488 Software Reference Manual for the GPIB-ENET p/n 320910A-01

Newport

tel: (800) 222-6440 fax: (714) 253-1680 tech.support (949) 253-1840 tech@newport.com

Motorized Rotation Stage 495B-A
Motion Controller MM3000

Laser Photonics, Inc

tel: (407) 281-4103 fax: (407) 281-4114
`Steve Qualls (president) 12351 Research Parkway Orlando, Florida 32826`

J References

References

[1] National Instruments GPIB Tutorial

[2] DiCon FiberOptics Facility Switch. Operation & Control Manual

[3] Newport MM3000 User's Manual

[4] Stanford Research Systems, Inc. Digital Delay/Pulse Generator Manual

[5] ESP-488 Software Reference Manual for the GPIB-ENET ftp://ftp.natinst.com/support/manuals/320910a.pdf

[6] TNT4882 Programmer Reference Manual ftp://ftp.natinst.com/support/manuals/320724.pdf

[7] Private communications with Eric Singer from National Instruments.

[8] Fleysher, L. et al., Proc. 26th ICRC (Salt Lake City, 1999), OG.4.4.03

[9] Lazar Fleysher, et al., Slewing Extrapolation. Milagro memo, 4/20/2000.

[10] Lazar Fleysher, Roman Fleysher, An Invitation to Random Functions. Milagro memo, 5/7/1999.

[11] Lazar Fleysher, et al., Time Pedestals. Milagro memo, 7/3/1997.

[12] Lazar Fleysher, Roman Fleysher, and Peter Nemethy. Time Pedestals II (Enhancements). Milagro memo, 11/1/1997.

[13] Lazar Fleysher, Roman Fleysher, and Todd Haines. Methods of Photo-Electron Calibration. Milagro memo, 6/19/2000. See also, Isabel Leonor, TOT-PE Conversion using Occupancy Method. Milagro memo, 4/22/1998.

[14] Andy Smith, Calibration (TOT to PEs) Status. Milagro memo, 3/23/1998. See also, Isabel Leonor, TOT-PE Conversion using Occupancy Method. Milagro memo, 4/22/1998.