PMT Test Facility User Manual

June 13, 2013

1 Introduction

The following is meant to provide sufficient instructions for the user to be able to operate the PMT test facility (hereafter PTF).

2 Motor Control

There are a set of 10 motors for the PTF, 5 for each gantry; they are 3 translation motor, 1 rotation motor and 1 tilt motor.

The motor are driven by a pair of Galil Digital Motor Controllers, which are mounted on electronics rack.

2.1 Limit Switchs

For the three translation axes we have mechanical limit switches. The output of the limit switches are attached to the Galil controller; if a gantry reachs a limit switch motion on that axis (in a particular direction) will be immediately stopped.

For the rotation axis we have a pair of magnetic limit switchs of the rotary table. Again, these limit switches will prevent motion beyond a certain angle.

We do not have any electrical limit switches for the tilt axis.

2.2 Phidget Accelerometer/Magnetometer

In order to be able to calibrate the tilt axis, we bought a separate electronics component that can measure 3 dimensional acceleration (and hence tilt). Specifically, we bought the USB-connected Phidget 1044.[1]

3 MIDAS front-end Programs

There are a number of different front-ends needed to run the PTF. A summary of the front-end is as follows:

- feMotor0[0,1]: This front-end talks to the Galil controller and provides motor control in low-level motor variables.

[1] `http://www.phidgets.com/products.php?product_id=1044_0`

- feMove: this front-end provides the user the ability to control the gantry in meaningful coordinate system, with collision avoidance and initialization. feMove initiates moves through feMotor settings.
- feScan: this front-end provides the ability to do a full gantry scan once a run is started. feScan initiates moves through the feMove settings.
- fePhidget: this front-end logs the information from the Phidget accelerometer/magnetometer.

The source code for all front-ends is under SVN version control. Details are given in following sections.

3.1 feMotor

The first frontend, called feMotor, was written before the first project team began work.[2] Therefore, it contains much of the basic functionality for the control software. By itself, it can be used to set the motor velocity and acceleration of each axis, to send the motors to a relative position on each axis and to monitor real-time variables such as the current position, the current state of the limit switches and whether or not the motors are currently moving. All the feMotor settings are in low-level units of motor steps. and has many options that will rarely be changed; the user should rarely have to change anything in feMotor. The user might occassionally want to check the feMotor variables to check if a limit switch has correctly activated.

There are two different feMotor front-ends, feMotor00 and feMotor01, each of which talks to a single Galil controller. Currently we only use feMotor00.

3.2 feMotor

feMove provides a meaningful control level to the user. feMove does this by only communicating to those values which are necessary for movement. These can be separated into three categories: settings, controls and variables.

- The 'settings' values in the ODB are read into feMove once upon initialization. These settings include velocity and acceleration as well as the additional settings of motor scaling, axis channels and limit positions.
- The control values are used to send commands to the axis motors. Therefore, they are hot-linked to functions in feMove which send the desired commands to feMotor. Hot-linking mean that, if the user changes the control values in the ODB, the function is called. These controls include Start Move and Stop Move as well as the additional control, ReInitialize. Furthermore, an array of destination values for the axes is included in the control values. This, however, is not hot-linked to any particular function and is simply read into feMotor when Start Move is set to true.
- The variables are used to monitor aspects of the Testing Rig in real-time. Like feMotor, they include the current position, the current state of the limit switches and whether or not the motors are currently moving. However, in addition to these, feMove also monitors whether or not a move has been completed, whether or not the gantry destinations have been swapped and whether or not the axes have been initialized.

[2] Originally written by Dave Morris of controls group.

The initialization of the axes is used to convert the relative position in counts that feMotor uses into absolute positions in meters. This is done by defining an origin using the position of the limit switches. Specifically, the initialization routine will drive each motor to its negative limit position; once it gets there the feMove has a reference point for that axis.[3] After the initialization feMove can meaningfully present the gantry positions in meters in an absolute coordinate system.[4]

This initialization is very important for the current usage of the PTF. The initialization must be rerun every time that the feMove program is started (because there is no way to know where the gantry was left after feMove was last turned off). If you try to initiate a move to a particular position before the initialization has been run, then feMove will automatically run the initialization first.

3.3 feScan

The purpose of feScan is to string together the individual movements of feMove into a path. Currently feScan just performs a predefined set of scans around a point at the middle of the PTF area. You initiate a scan by starting a MIDAS run.

If desired, feScan can also take Magnetic Field data by communicating with the fourth frontend, fedvm. fedvm is solely used to send commands to a digital voltmeter connected to a 3-axis Hall Probe. It is being used ”as is” from another project which required similar data. feScan uses the ”run” feature in MIDAS to produce Magnetic Field and position data in ”banks”. The format of these banks can be read by an auxiliary tool analyzer_example.exe, which produces the data tables found in the Results section below.*This feature is currently not working.*

4 Instruction Checklist

The following is a checklist for operating the PTF.

1. Inspect the PTF gantry area; ensure that all obstructions are removed and that gantry motion is unimpeded.

2. Logon to midptf01; username ’midptf’, password ’pmtCalib???’

3. Turn on the power to the Galil motor controllers (left side of electronics rack).

4. Open the MIDAS webpage interface by going to

   `http://midptf01:8081`

5. Turn on the feMotor00, feMove, feScan and fePhidget front-end programs. You can do this from the Programs link off the MIDAS status page or you can start the front-ends from a terminal by doing

   ```
   cd online/src
   feMotor -i 0
   feMove
   ```

[3]As noted above the tilt axis doesn’t have a limit switch; instead it use the Phidget tilt measurement to try to level the axis.

[4]though not necessary the best coordinate system. See final section of manual on known problems.

feScan

fePhidget

The four last commands must be done in separate windows. Starting the commands on the terminal is useful if you want to check additional debugging information.

6. Now you should initialize the gantry position in feMove. You can do this through the feMove ODB Control settings, but it is easier to go to the 'Gantry Control' custom page:

   `http://midptf01:8081`

   and click 'Reinitialize Gantry'.

   The gantry initialization will take a couple minutes (depending on where the gantry starts). You will hear an audible click when the gantry reachs the limit switch of each of the three linear axes. Keep an eye on the gantry to ensure that the limits are all correctly reached.

7. You can then use the 'Destination' fields in the 'Gantry Control' page to choose a new location which to move the gantry; then click 'Move Gantry' to initiate a move. Remember there is no collision avoidance in the feMove code yet! Be careful!

8. Finally you can try running a scan. As long as the feScan program is running, just click 'start' from the MIDAS status page and start a run. The scan sequence takes 15 minutes now.

9. Once you are finished for the day, power down the galil controller. That seems like the safe practice.

5 List of Known Problems and Needed Improvements

Work still to do:

1. The second gantry is not at all cabled. It needs all cable and electronics installed. This will presumably wait until we are fully satisfied with the first gantry.

2. Don't have any idea about the cables for the laser, APD, etc in the housing. Need to ensure that cable runs work for these cables as well.

3. Some of the Galil limit switches seem to intermittently fail; suspect that we need to switch to another set of Galil channels.

4. Need to modify initialization so that we move fully up in Z before initializing in X and Y. Wayne says that this should ensure that we avoid the edge of the water volume.

5. Need to implement the collision avoidance code and give it a sensible set of objects to avoid.

6. Need to define a sensible global coordinate system for the user; for instance, +Z is currently pointing down.