ForceTwo Document, version 1.1
Power Modules Version 2.1
Firmware Version 6.5
2023-06-09

Table of Contents

Overview:

Our ForceTwo controller can control brushed motors, conventional brushless motors, and direct drive motors.

Each motor is controlled by a Texas Instruments dual core Digital Signal Processor, with floating point processor.
In addition to 4 PID's (Position, Velocity, and two current loop PID's), there are 4 notch filters, and 3 low pass filters, along with a deadband filter.
PID stand for Proportional, Integral, Derivative.
The Force Two Telescope Controller has two Power Modules, and one CPU Module (which is a Raspberry Pi 3 and a HAT (Hardware Attached Top).

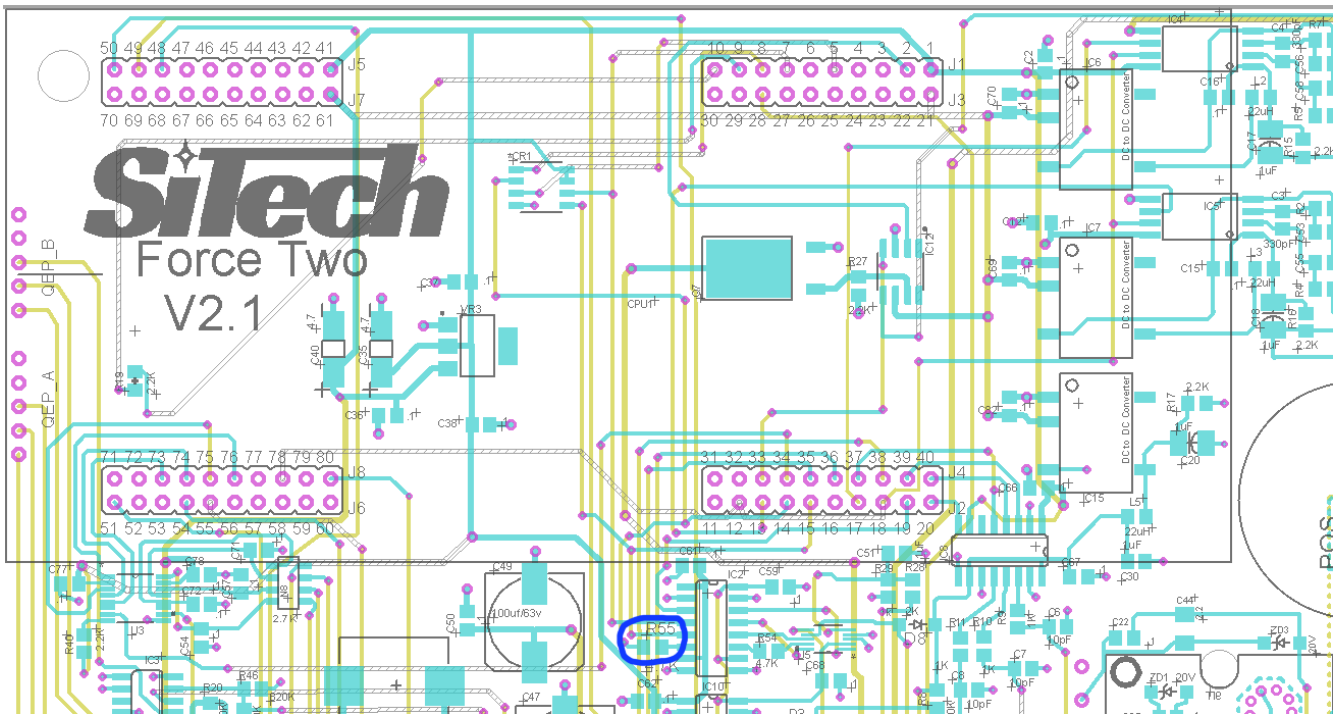
There are 2 high speed incremental encoder inputs, and 1 absolute (BissC or Endat2.2), 4 digital inputs, and 1 analog input on each power board.

The voltage rating of the motor supply is 250 volts, (this is a conservative rating). The continuous current (with a 30 amp bridge driver chip) is 20 amps (this is also a conservative rating).

The 3 phase bridge driver chip is rated at 600 volts and 30 amps continuous (amperage is based on 80 Degs C bridge temperature).

Starting with version 2.1 of the power modules, there is a flash rom which stores the configuration.
If you have an earlier version, there is no flash rom. Instead, upon power up, the configuration is automatically sent to the power modules.

To see if you have version 2.1 power modules, notice that the label is underneath the CPU module of the power module, so it's hard to see. Instead, see if R55 is on your board. If so, it's version 2.1 or later, and you should have a flash ROM chip.



CPU Module:

The CPU Module (the Raspberry Pi with HAT) is connected to both Power Modules via a short USB A to USB mini-B.

Near the CPU module, there is a 3 amp 5 volt supply which supplies power to the CPU via the USB Micro connector.

On the CPU Module "HAT" there is a USB B connector. This needs to be connected to a Linux or Windows computer, in order to run our software, SiTechExe.exe.

There are 3 connectors on the HAT. The one on the left is the handpad connector. The one in the middle is the Slave USB port, and the one on the right is the autoguider.

Power Module:

There is no enclosure for the Power Modules, they need to be mounted in an enclosure.
Allow plenty of space for cooling.
The fan on the back of the heatsink will automatically come on when needed.

The Power Module has a BiSS encoder 9 pin D-Sub connector, which can be connected to Renishaw Absolute 26 bit or 32 bit encoders, or Heidenhain encoders with the Endat2.2 protocol.

Wiring:

See drawings in Appendix C.
The motor power positive should be connected to the circuit breaker on the left side of the Power Module, at the top of the breaker. The bottom of the breaker will already have a red wire that is connected to the motor power input terminal block, on the right hand side of the power module, terminal 2.

The motor power negative should be connected to the motor power input terminal block, terminal 1.

The 3 phase brushless DC motor should be connected to the motor terminal block. Pin 1 is C, pin 2 is B, and pin 3 is A. This terminal block is on the right end of the Power Module edge.

If you're connecting a brushed motor, use the A and C connections.

Adjacent (just to the left of the motor terminal block) is a 4 pin plug-able connector. Pin 1 (on the left) is GND, and pin 2 is +12 volts. This should be a 2 amp minimum regulated power supply.

Pins 3 and 4 of this connector could be connected to a relay with a 12 volt DC coil, which could used to operate a brake. When the controller is operating in auto, this relay will be energized. If there is a motor fault, or if the power module is put in the "manual" mode, this relay will be de-energized.

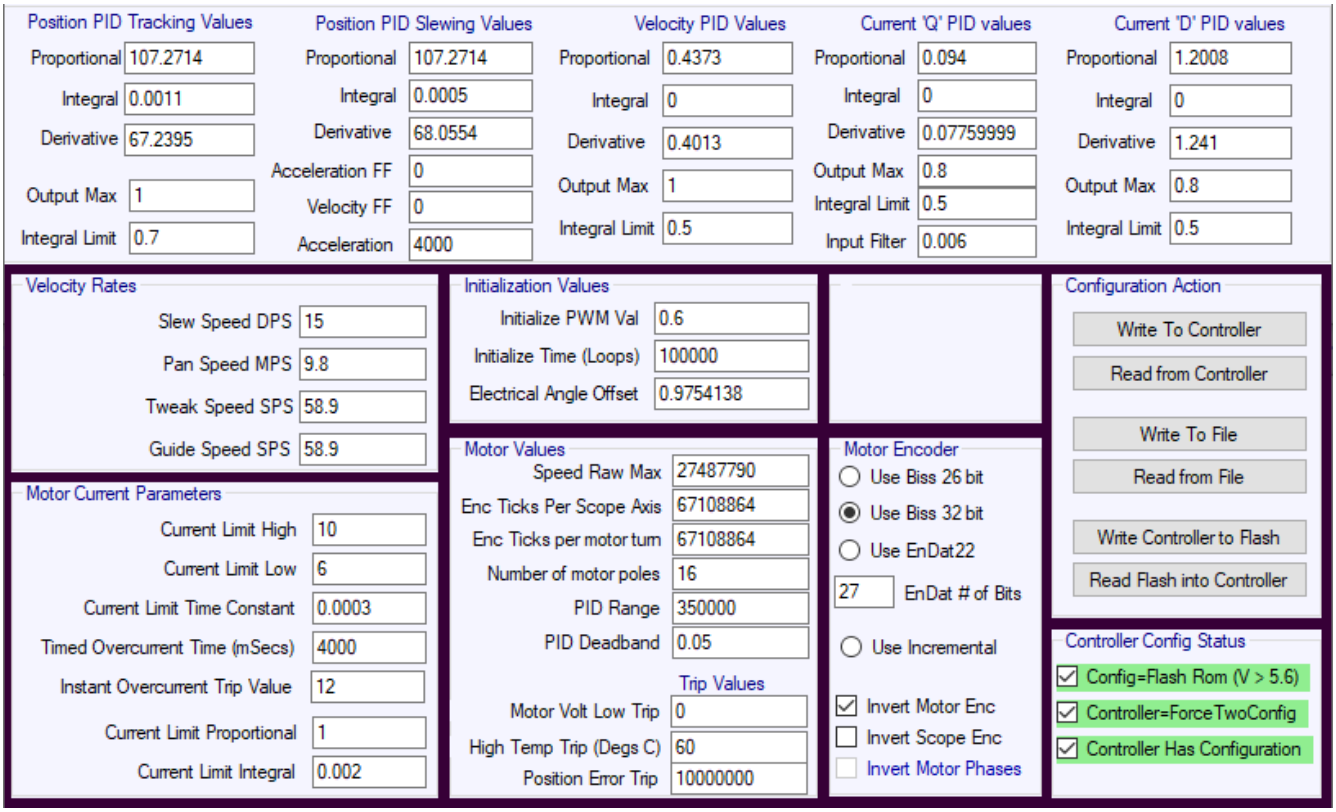
The Absolute BiSS encoder should be plugged into the 9 pin D-Sub connector on the left end of the Power Module edge. If you have a Heidenhain Endat2.2 encoder, you'll have to have an adapter cable.

Plug the Mini-B USB cable into the Power Module (it's just behind the large inductor near the back right corner), then into the CPU Module (the Raspberry Pi).

Do the same for both Power modules. The two USB connectors on the Pi are interchangeable; The software in the CPU module will figure out which power module is which, based on the jumper on the Power Module (See the drawing in appendix C). This jumper should be in place on the Primary Axis (Right Ascension or Azimuth), and should not be installed for the Secondary Axis.

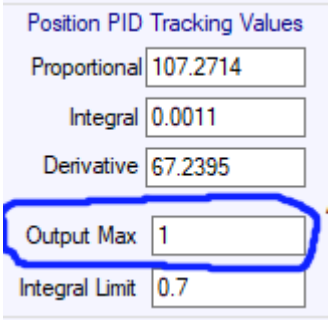
Configuration of the Power Module:

Here is a screenshot of the Configuration Tab:



Don’t be overwhelmed, we’ll go through one parameter at a time.

If this is your first time setting everything up, *make sure there is no voltage on the motor power supplies*. Also, best to set the “Output Max” to zero. This parameter is in the section “Position PID Tracking Values”.



Before configuring, make sure that 12 volts is connected to both of the Power Modules and the mini USB are connected between the Power Module and the CPU module (Pi).

When you power up the Power Modules, the Red LED on the front of the Power Module should be flashing.

You need to find the IP address of the CPU Module. It's set up for DHCP when we ship them.
You can use the free software **Advanced IP Scanner** or a similar program to find it on your local intranet.

Or, you can connect a keyboard and HDMI monitor, and find the IP address using:
ifconfig

In the latest version of the CPU Module, the green LED on the PI will flash the last group of your IP address directly after a re-boot. For instance if your IP address is 192.168.0.206, it will flash 2 times, pause, then 10 times (for the zero), pause, then 6 times.

Now, you need to connect to the CPU Module (Pi). 3 ways to do this….
1. You can plug a keyboard, mouse, and HDMI monitor into the Pi,
2. Use Windows Remote Desktop.
3. On another Linux machine, log in with SSH -l pi -Y 192.168.?.?

Note when using Remote Desktop: If you will run a graphics program such as ForceTwoConfig.exe, or CodeBlocks IDE using the "sudo" command, you need to issue this from the terminal first:
"xhost +".

The user name is **pi.**
The default password is **servo**. For security, we suggest you change it, and document the new password with your IT folks.

There is a program that is automatically running when the CPU Module (Pi) starts. It's called **ForceTwo**.

The configuration software is named **ForceTwoConfig.exe.** This software is installed on the CPU module (Pi).

From a terminal, type **mono ForceTwoConfig.exe**

If it exits with a message about "You'll have to Kill ForceTwo using htop, or run this program using sudo", try again but use **sudo mono ForceTwoConfig.exe**

Again, if you're using 'sudo', you'll have to issue the "xhost +" if you're using remote desktop.

This will KILL the **ForceTwo** program.So after configuring, and before running **SiTechExe** on another computer, you'll have to restart **ForceTwo** or reboot.

To Restart **ForceTwo**, from the terminal, type:
sudo ./ForceTwo

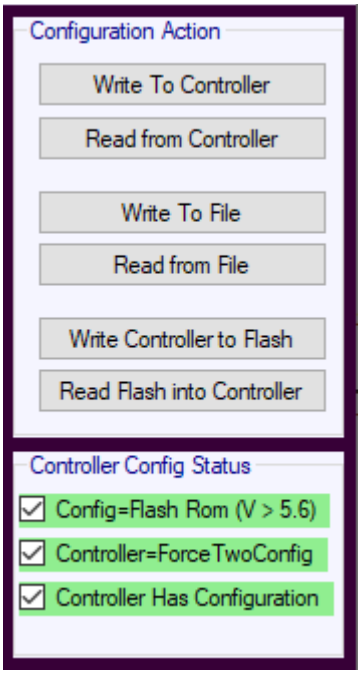
If you have an early version with no flash rom, on power up with 12 volts, there is NO configuration in the power module.
In this case, each time the power module is booted, it needs the configuration sent to it. This normally happens automatically by the "ForceTwo" program.
These area the two files which are loaded to the appropriate power modules:
ConfigFromPri.AxCfg
ConfigFromSec.AxCfg
This means you should always save the current configuration to one of these files as appropriate.

ForceTwoConfig.exe can only configure one Power Module at a time. You can select the other power module by choosing the other **ttyUSBn** port near the upper left. Once you have communication, you'll see which axis you're communicating with, on the window label. The **Primary** axis is the right ascension or the azimuth, the **Secondary** is the declination or the altitude.
If you want to configure both at once, you can simply run two instances of ForceTwoConfig.exe.

IMPORTANT!:
Make sure you know which module (primary or secondary) you're configuring, and make sure you know which configuration files go with which module. The filenames must have a **"Pri"** in them for primary, and "**Sec"** in them for secondary. Spaces in filenames is a bad idea.

Ok, let's get started…..
A note about where configuration information is stored:
There are 4 places where configuration is stored.
1. A file on the CPU Module
2. While running ForceTwoConfig.exe, it's stored in the text boxes on the Configuration tab and the Filter Parameters tab.
3. In working memory of the ForceTwo Power Module,
4. In version 2.1 of the power boards, you can store the configuration to Flash Rom.

Here's a screenshot of the configuration load and save area:



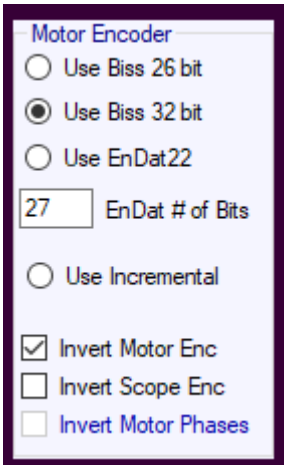
If the Controller Config Status is all green, then ForceTwoConfig.exe is equal to the working memory configuration in the Power Module, and is also equal to the Flash Rom in the Power Module.
The buttons are fairly obvious. If you make a change in ForceTwoConfig.exe, you can click the "Write To Controller, and the current configuration in ForceTwoConfig.exe will be sent to the working memory in the Power Module. When testing and configuring, that's the simplest way to see if it helps or hurts, etc. After you're happy with the configuration, save to a file and also to flash rom (if V2.1 power modules).

First, let's start with a default configuration.
If you have flash rom (V2.1 of power boards) you will already have the default configuration from our factory.
If not, go to the **Configuration tab**, and click on **Read From File**

If you're connected to the primary axis, select the file named **ConfigFromPri.AxCfg**
If you're connected to the secondary axis, select the file named **ConfigFromSec.AxCfg**

Setting the Motor Encoder Direction:

First, check the **Use 26 bit Encoder** for a 26 bit BiSS encoder, check the **Use 32 bit encoder** for a 32 bit BiSS encoder, check Use Endat22, or check the **Use Incremental** radio button.
If you're using an Endat2.2 encoder, you'll have to put the number of bits in the textbox labeled **Endat # of bits**.



Click on **Write To Power Board**, to save this value (note, it's only saved in the working memory, if you reboot, you'll loose these values).

Take a look near the bottom of the window, you'll see the **Enc=, motor encoder, the posErr, MotVolts, Amps,** etc. This is called the **Drive Info Area**. This is visible from any tab, and is important information. When I refer to the **Drive Info Area**, the information will be visible there.

Notice the **Loc=**, the 1st value on the 2nd line. This is the motor encoder number in raw ticks.

NOTE!: Keep in mind that if it's a negative number, the number should be a smaller negative number while *increasing*.

Primary axis:
While watching the motor encoder location, move the primary axis by pushing the telescope axis by hand (or turning a geared system motor by hand).

Alt/Az Mount:
The motor encoder should increase when moving clockwise (birds eye view).

GEM Mount:
The motor encoder should increase when moving West.

Fork Equatorial Mount:
The motor encoder should increase when moving West.

If the motor encoder value is moving the wrong way, change the checkbox labeled **Invert Motor Encoder.** Remember, if it's checked, you need to uncheck it, and vice versa. Then, click on **Write To Power Board** to save to the working registers.

Secondary axis:
While watching the motor encoder location, move the secondary axis by pushing the telescope axis by hand.

Alt/Az Mount:
The motor encoder should increase when moving toward the zenith.

GEM Mount:
Before starting, make sure the counterweight is pointed to the west.
The motor encoder should increase when moving NORTH.

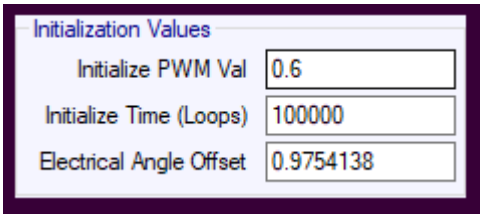
Fork Equatorial Mount:
The motor encoder should increase when moving NORTH.

If the motor encoder value is moving the wrong way, change the checkbox labeled **Invert Motor Encoder**. Remember, if it's checked, you need to uncheck it, and vice versa. Click on **Write To Power Board** to save.

If you have a flash rom (V2.1 of Power Module) since you've got the motor encoder directions, you should save to flash.

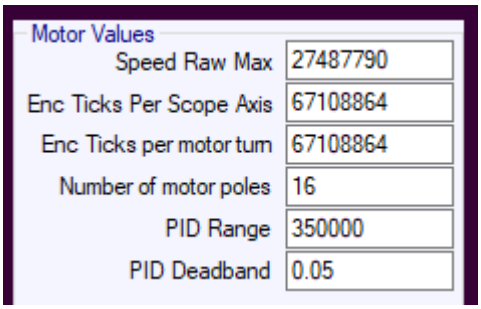
OK, now that we have the motor encoder direction proper, we must save the config file. Click on the **Write To File** button. Maybe at this point, you should select a different file, so you can always get back to the original. Once everything is working properly, make sure you save it to the **ConfigFromPri.AxCfg** or **ConfigFromSec.AxCfg**, so the **ForceTwo** software will load the proper configuration. When you save it to an alternate filename for a backup, make sure you have **"Pri"** in the filename for the primary axis, and **"Sec"** in the filename for the secondary axis. Remember, no spaces in filenames.

Setting the Initialization Values:



Set the **Initialize Current** and the **Initialize Time** (Servo Sample Loops).
The initialize current should be set at a higher current value that the motor normally runs when accelerating. This value is 0.0 to 1.0. 1.0 is 100%, so will apply full voltage to the motor windings.
Set the initialize time value to be at least 100000. This is the number of sample rate ticks, which is 6.666KHz. The main idea, the mount has to settle down in the initialization time period.

Setting the Motor Values:



Enc Ticks per Scope axis.

If you have an Absolute BiSS encoder (26 OR 32 bit), the Ticks Per motor turn should be 67,108,864. If not, you should calculate this based on your gear ratio.

Enc Ticks per Motor Turn.

In the case of a direct drive, it will be the same as the Enc Ticks Per Scope Axis. Again, if you have an absolute BiSS encoder, or a Heidenhain Encoder, the number should be 2^26 or 67,108,864 (even if you have a 27 bit – 32 bit encoder).

Number of poles:

Important Setting: Now enter the number of motor poles. If you're using a brushed motor, this should be zero.

PID Range

The default "PID Range" is zero. If this is zero, then the position loop position error is full scale output + or – when the position error reaches the encoder ticks per electrical cycle. This number is the "enc ticks per motor turn" / (Number of poles * 2).

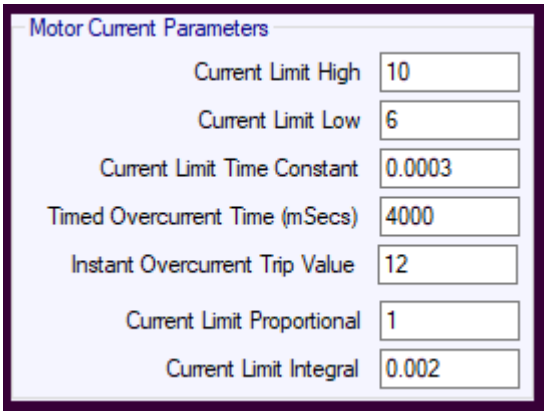
If the "PID Range" is not zero, the PID range will be based on this number. For instance if you have 50,000 in this textbox, when the position error is 50,000, the position PID loop will have 100% output with a Proportional of 1.0 and integral of zero. You can increase the overall gain of the PID's by decreasing this number.

If you have a direct drive brushed motor (yes, there is one Brushed Direct Drive in New Mexico), you will need to make this number somewhere around 1,000,000.

PID Deadband:

When the PID output switches from positive to negative (or vice versa), there's a slight portion where there is no effect on the motor current. This setting will pass over that value instantly, providing tighter control.
Start with 2% which would be 0.02.
If this is too high, you'll get a high pitched hum when there is zero torque on the motor.
When you first go to the Auto mode, theoretically, there will be zero torque on the motor. So try setting this with zero torque and stopped.

Setting the Motor Current Control Values:



First item is the **Current Limit High** This will depend on your motor, and required current for motion, acceleration, imbalance and friction. In no case should it be more than 24.

Next is **Current Limit Low.** This is the value that the motor current setpoint is lowered to over time. I like ¾ of the **Current Limit High** value, but use what you think is best. We can always change this later.

Next is **Current Limit Time Constant**. This is an arbitrary number, but it effects how long it takes for the motor current to roll back from the **Current Limit High** value to the **Current Limit Low** value.

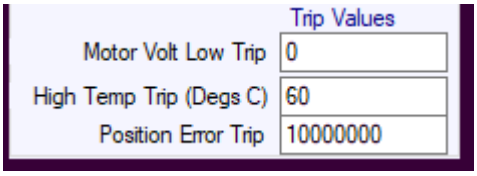
Next is the **Timed Overcurrent Time**. This is set in mSecs, and if the motor current is at the **Current Limit Low** setpoint for this amount of time, it will trip the drive with a **Timed Overcurrent** error.

Next is the **Instant OverCurrent Trip** Value. This should be no more than 30.

There is a simple PI (Proportional Integral) loop that keeps the current below the current setpoint. The default proportional is 1.0, and the default integral is 0.002. You should probably not change these, unless you have an unusual amount of motor winding inductance or motor winding resistance.

Click on **Write To Power Board** and also, **Write to File** and save it again.

Setting the Trip Values:



Motor Voltage Low Trip should be set probably about 50% to 70% of your supply voltage.
High Temperature Trip (Deg's C) should be set at about 60 deg's. If you have nuisance trips, you could set it higher, but no more than 75 deg's.
Position Error Trip should be set as you see fit, but you should not have nuisance trips. It's set for motor encoder ticks. The default is 1,000,000 ticks.
Once set, click on **Write to Power Board** button, and then click on the **Write To File** button.
Also, if you have flash rom, write to flash.

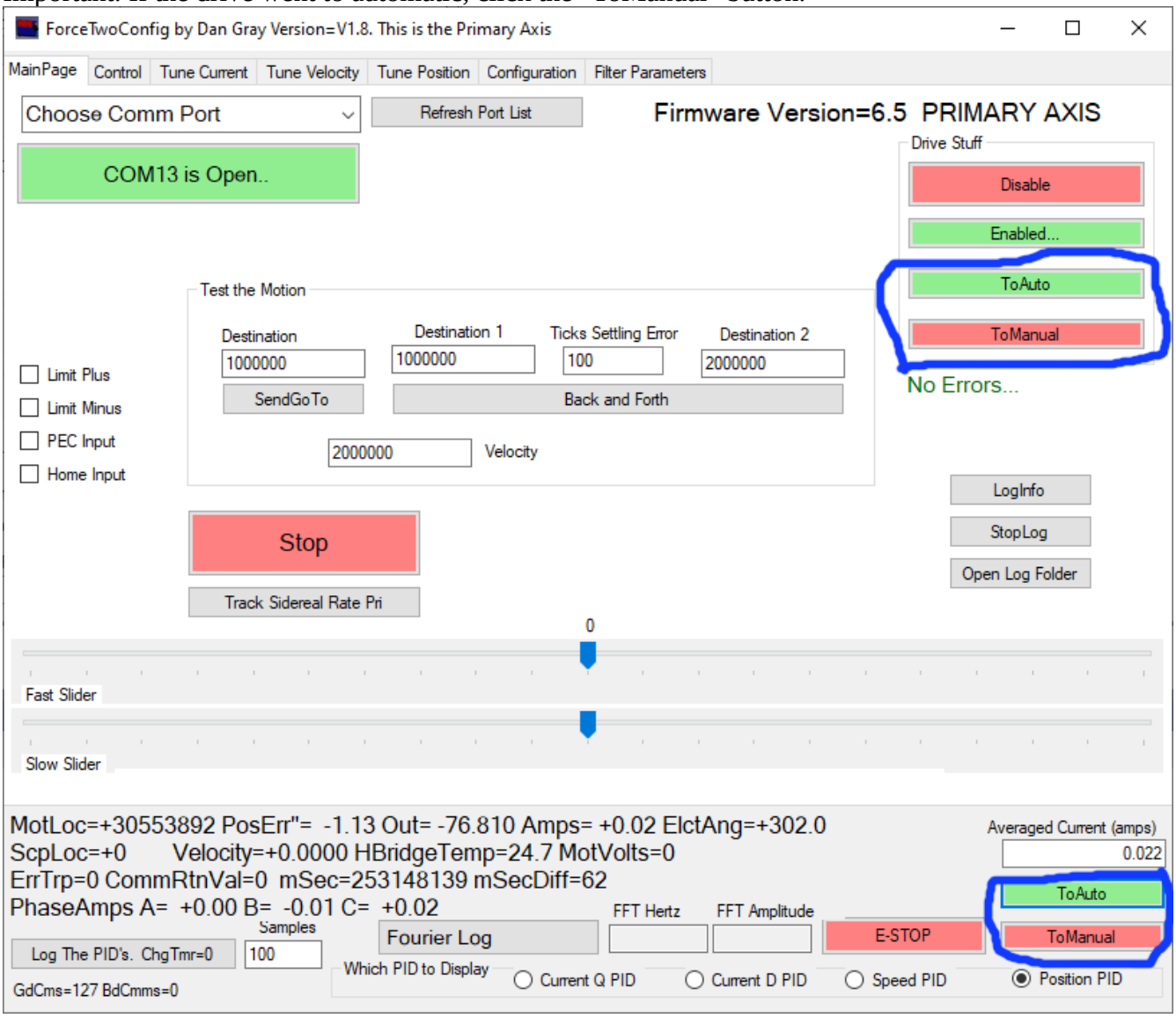
Checking the motor direction:

We've already set up the encoders to move the proper direction. We've set up most of the motor parameters. Before testing in "Auto", we need to make sure the motors turn the proper direction in relationship to the motor encoders.

Go to the **MainPage** tab again. The volts should read near zero, i.e., less than 4 or 5 volts.
NOTE: For safety, before turning on the motor power for the first time, set the Output limit to 0. The default is 1.0 (100%). This is in the "Position PID Tracking Values" section of the Configuration tab.

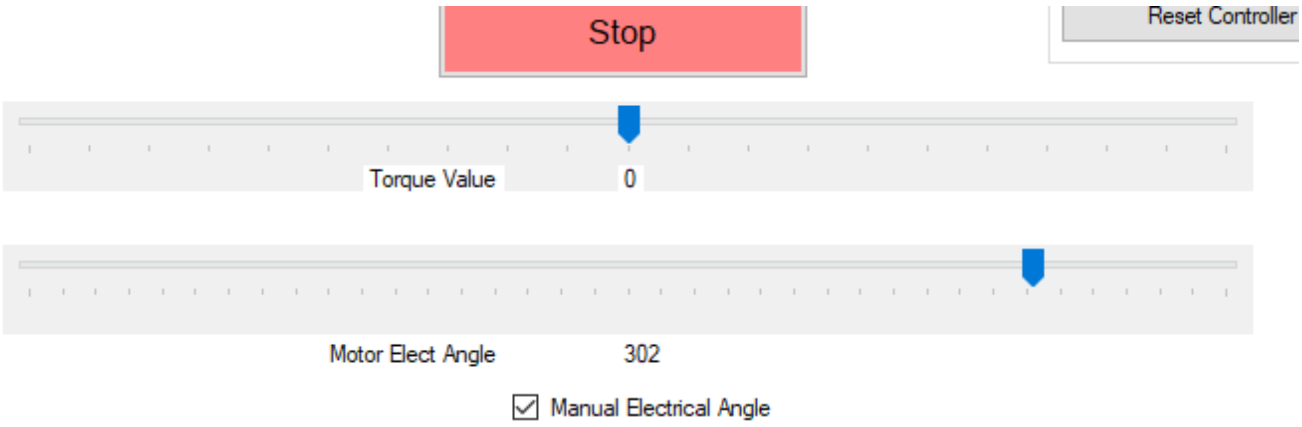
Turn on the supply voltage. Watch this voltage in the **Drive Info Area**, it should read near the motor power supply voltage that you have.

Important: If the drive went to automatic, click the “ToManual” button.



Under the **Drive Stuff** container (top right), the errors are listed. These errors are “latching”, so it means they were there before now. Hopefully, you don’t have any errors. When you click on the pink button above this labeled **Disabled. Click to Enable**, It should turn green, and the errors box should say **No Errors...** If you still have an error, we need to deal with that before proceeding.

If all is well, click on the control tab. There’s two sliders, the **Torque Value** slider and the **Motor Elect Angle** slider.



Normally, the motor electrical angle that you see near the bottom of the screen (it's labeled **ElctAng=**) comes from the data that you sent it earlier, the number of poles and the ticks per motor electrical angle.

To check the motor direction, you must first check the checkbox labeled "Manual Electrical Angle". This will be grayed out (disabled) if you're in the Auto mode.

If you're in the Manual Electrical Angle mode, when you move the bottom slider (**motor Elect Angle**), the Power Module will use the slider as it's motor electrical angle, instead of the actual calculated motor electrical angle. When you slide this slider, the motor will respond by acting sort of like a micro stepper motor.

Anytime during this test, if something bad happens, you can always click on the **Disable** button near the top right, or from ANY tab, you can always click on the **E-Stop** button near the lower right. This will force the Power Module into the **Disabled** mode.

Step one:
If you set the output limit to 0, you must raise it. I would suggest about 0.5 (50%).
To proceed with the test, move the **motor Elect Angle** slider to the left, to zero.

Step two:
Move the **Torque Value** slider slowly to the right (Remember that zero torque is the center). Watch the telescope axis AND the motor current. The telescope axis should move. When it moves, increase the motor current another 25% or so (to make sure you overcome any mechanical resistance).

Step three:
Slowly move the **Motor Elect Angle** slider clockwise. Watch the telescope axis, and also the **Loc** number near the bottom of the screen.
The axis should move in the same direction as determined when you set up the encoder direction. The motor encoder (the **Loc** number) should be INCREASING. Remember, if this is a negative number, it should move to a smaller negative number.

If this is backwards, move the **Torque Value** slider to the center (near zero torque), or click the stop button.

Click on the **E-Stop** or the **Disable** button. Turn off the power to the motors. Watch the motor voltage value on the **Drive Info Area**, and when it is below a safe value, proceed as follows:

A small inconvenience here, there is a "grayed out" checkbox on the **Config** tab called **invert Motor Phases**. This doesn't work for now, so you'll have to swap any two leads of the motor phases, then repeat step one to step three.

IMPORTANT! *Once you're finished with the motor direction, then don't forget to move the* ***Torque Value*** *slider to zero and uncheck the "Manual Electrical Angle" checkbox.*

Finding the motor electrical angle zero point:

Direct Drive with absolute encoder

On any new motor installation, and assuming it's an absolute encoder, at some point, you'll have to "find" the motor electrical angle zero point or offset.

Set the **Initialize Current** and the **Initialize Time** (mSecs) on the **Config** tab.
The **initialize current** should be set at a higher current value that the motor normally runs when slewing. Set the **initialize time** value to be at least 100000. This is in the sample rate loop count, at 6.666Khz. The idea is that the axis needs to settle down during the initialization time period.

The **Initialize Offset Angle** is calculated by the initialization process.

Select the **Control** tab.
If the **Torque Value** slider is not at (or near) zero, move it to the center.
If the **Motor Elect Angle** slider is not at -1, move it to -1 (all the way to the left).
If the drive is not enabled, click on the **Disabled. Click to Enable** button
Now click on **Initialize Motor Electrical Angle** button, which will start the initialize cycle.

Watch the motor amps (on the **Drive Info Area**), it should ramp up to the set value. The axis should move to it's natural magnetic field. Sometimes it's helpful to "help" the axis by moving it slightly back and forth, then release before the initialization period times out.

Important! Once it's done initializing, you need to read the new value, then save it to your current configuration file. To do this, go to the **Configuration** tab, click on the button **Read from Power Board**. Note that the value **Initialize Offset Angle** will change. Now save your configuration to file, by clicking on the **Write To File** button. If you have a flash rom chip, save this to the flash rom as well.

Direct Drive with Incremental Encoder:

Directions are similar to above, except this will need to be done on every power cycle.

Conventional Brushless Motor with incremental encoder and encoder index:

We intend to make this automatic, but for now, it's manual. You'll only need to do this one time though.

1. Reboot the Power module (this should set the motor encoder ticks to zero), then Enable the power module, and put in manual.

2. On the "Control" tab, click on the Manual Electrical Angle button. Apply some positive torque with the Torque Value Slider. Watch the motor current. It's important to not exceed the continuous current setting for your motor, but should be close to it.

3. Using the Motor Electrical Angle Slider, slide the slider to 265 degrees. Now click on the slider line to the right of the slider "handle". Each time you slide it, the angle will increase a small amount. Click until it's 270 degrees, or the closest number to 270. Start over if you go to far.

4. Note the motor encoder position in the "Drive Info Area". We'll call this position "A".

5. Now move the Motor Electrical Angle Slider to 275 degrees, and click to the left of the slider handle until it reaches the closest to 270 degrees.

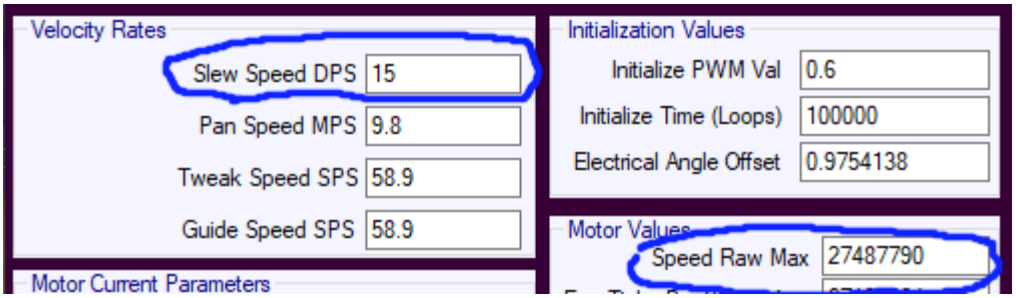
6. Note the motor encoder position, again. We'll call this position "B"

7. In the "Electrical Angle Offset Configuration Item, put in this number: 0 - (A + (B – A) / 2).

8. Send to Controller, then Save to file, and Save to flash.

Next time you turn on the power module, the motor should turn slowly until it detects the index, and will initialize the encoder to the value of 0.

Tuning your axis:



The first thing to do is decide on a slewing velocity.
There area two text boxes, one of them is in degrees per second units, and the other is speed raw units.

When you change one, the other changes, and the textbox background changes for a second or two.

This is an important setting. When you move the Fast Slider, it will limit the velocity to your setting. Also the velocity PID is based on this setting, IE if you have the velocity setting for 10 degrees per second, the velocity error will be based on 10 degrees per second. I would suggest start with a slower slew speed, when everything is tuned and working, you can always speed it up.

Inside the Power Module, there are 4 PID's.
First is the position loop. There is always a motor location setpoint in the controller, based on the latest velocity and position command. This setpoint is sent to the position loop, and the position is sent as a "process". The PID calculates an output (-100.0% to 100.0% or the output limit you set).
This position loop output is sent to the Velocity loop as it's setpoint. The d/dv velocity calculation output (again, -100.0% to 100.0%) is sent to the current loop as it's "process". The output of the velocity loop is the setpoint to the current control loops, which take into account the motor electrical angle, which is then sent to the PWM outputs. See figure 1:

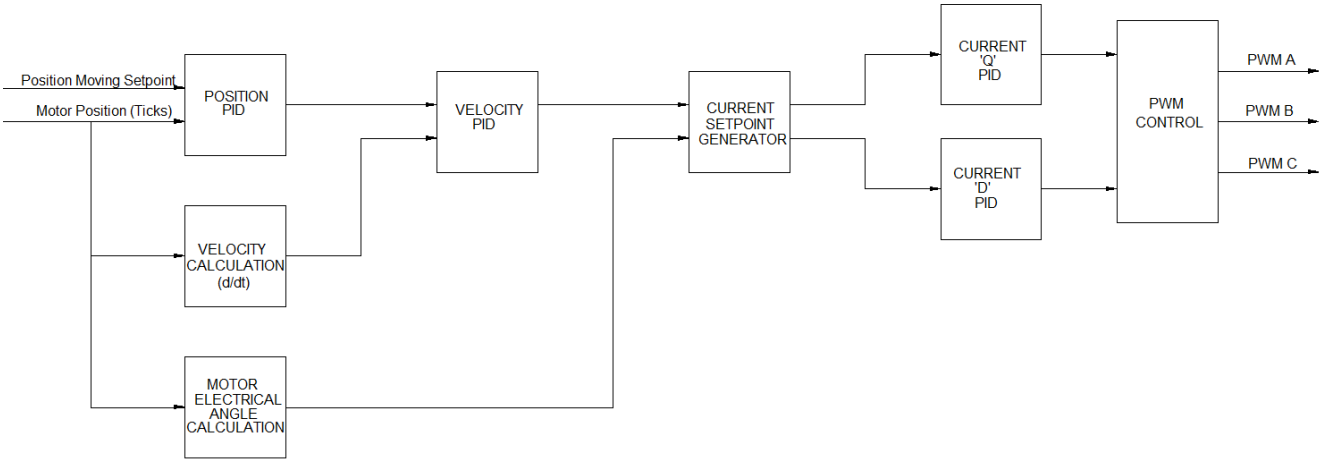


Figure 1.

The PID's should be "tuned" in the following order:
Current 'D'
Current 'Q'
Velocity
Position

Current 'D' tuning:
Internally, it's setpoint is always zero, making it easy to tune.
The default is:
Proportional 6
Integral 0.2
Derivative 0
Output Max 0.9
Integral Limit 0.8;
Hopefully, you won't have to change these.

Current ‘Q’ tuning:
Start with:
Proportional 2.5
Integral 0.0
Derivative 30
Output Max 0.9
Integral Limit 0.5
Input Filter 0.0

Velocity tuning:
Start with:
Proportional 1.0
Integral 0.0
Derivative 1.0
Integral Limit 0.5

Note: There are two sets of position loop PID settings, Tracking settings and Slewing Settings. Normally, you need a tighter PID control when tracking or moving slowly. The raw velocity threshold for going from the tracking settings to the slewing settings is set for a starting point raw speed of 100,000, and will be fully engaged to the slewing PID parameters 600,000. When de-accelerating, it will slowly integrate to the tracking settings when below 100,000.

Position PID Slewing Values Primary Axis:
Start with:
Proportional 50.0
Integral 0.0001
Derivative 120.0
Acceleration and Velocity FF should start at zero for now.
Acceleration should be set low enough so the controller does not go into current limit during acceleration or de-acceleration. Start with 1000, and move up from there. It is in Ticks per loop / 65536.0. The loop sample rate is about 6.666KHz.

Position PID Tracking Values Primary Axis:
Start with:
Proportional 80.0
Integral 0.0001
Derivative 120.0
Output limit should be 1.0.
Integral Limit should be about 0.5

Position PID Slewing Values Secondary Axis:
Start with:
Proportional 25.0
Integral 0.0001
Derivative 160.0

Position PID Tracking Values Pri Secondary Axis:

Start with:
Proportional 20.0
Integral 0.0001
Derivative 200.0
Output limit should be 1.0.
Integral Limit should be about 0.5

Once you have your starting values, go to the **Config** tab, and **Write to Power Board**, and also save the file by clicking on **Write to File**. You can also save to flash if you have a flash rom chip.

There are three tabs for adjusting the PID's with sliders. The sliders are cubed, to make then slow at the bottom end, and fast at the top end.
When you make an adjustment, if you don't like it, you can always click the **Restore** button.

IMPORTANT: When you're happy with your adjustments, you have to realize that the new values are ONLY stored in the controller. To get them into the **Config** tab you MUST click on the **Read From Power Board** button, then **Write to File** button. If you don't, you'll loose your adjustments. Also if you have a flash rom chip, you should write to the flash rom.

At the bottom of the screen, you can click on the radio button of the loop you're tuning, and click on the **Log the PIDs** button, to sample a bit, then a graph will be displayed.
Use your mouse inside the graph, and you can read the loop time, and the values for all of the items on the readouts on the right hand side of the graph.

A few tips. A slow oscillation is usually caused by too much integral. A medium oscillation is usually caused by not enough derivative, or too much proportional. A musical note that's fairly low frequency (usually perceived as a musical note in the 100's of hertz range) is caused by too much derivative in the position loop. A high frequency or "hash" is usually caused by too much derivative, or too much proportional in the 'Q' current loop.

OK, let's get started.
Let's assume you have initialized your motor electrical angle, all your **Current Control Values, Motor Values, Trip Values, and Initialization Values** have been set.
Mentioning one more time, on the **Control** tab, make sure the **Motor Elec Angle** checkbox is unchecked.

Let's go to the **MainPage** tab, and click on the **ToAuto Button**. If you have an oscillation, go to the **Tune Position** tab, and slide the **Integral Gain Tracking** to 0.0 (for now). If you still have oscillations, lower the **Proportional Gain Tracking**.
If it's playing a low note (in the 100's of hertz), turn down the **Derivative Gain Tracking**.

If there's hash, or a high note, click on the **Tune Current** tab, and turn down the **Proportional Gain Current 'Q'** until it stops. If necessary, turn down the **Derivative Gain Current 'Q'** as well.

When things stead out, you're ready to tune the current loop 'Q'.
Click on the **Tune Current** tab.

We'll leave the **Integral Gain 'Q'** at zero.
Slide the **Current Derivative** to the left and to the right. You'll probably have a low frequency note (in the 100's of hertz) if you remove all the derivative, and a high frequency note with too much derivative. If necessary play around with the **Current Proportional 'Q'** and **Current Integral 'Q'** until it's quiet enough and things are stable. So far, my experience is the derivative should be about 2 to 4 times the proportional.

OK, let's assume we're done with current 'Q', and we assume the default values are OK for current 'D'. Click on the **To Manual** button, then go to the **Tune Position** tab, and restore the 3 tracking values.
Now we click on **Read From Power Board**, and then **Write to File.**

Next, let's assume the **Tune Velocity** is OK with the default values (Proportional Gain 1.0 and Derivative Gain of 5.0 or so). You can play with these settings if you need to improve your velocity change position error.

So, go to the **MainPage** tab, and give it a speed of about 100,000 with the **Slow Slider**.
On the **Drive Info** Area you'll see the location change and you'll see the **PosErr** in arc seconds. The position error should be 0.0n with a possible 0.1n once in a while. It probably will need some help.

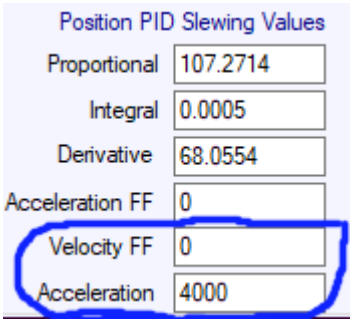
Turn the **Derivative Gain Tracking** up until you hear the musical note, then back it off until it's quiet.
Turn the **Proportional Gain Tracking** up until you can hear (or see) it transition to unstable. Back it off until it's steady. You can try lowering the proportional and increasing the derivative, or vice versa, to see which creates the least amount of position error. You can click on the **Position PID** radio button, then click on the **Log the PIDs** button, and view the graph. The position error is in arc seconds. When you're happy with it, make the slewing values be the same, except:
Make the proportional gain slewing be about 20% less.
Make the integral gain slewing be about 50% less
Make the Derivative Gain Slewing be 20% more.

Typically, the slewing proportional should be less than the tracking proportional, the slewing integral should be less than the tracking integral, and the slewing derivative should be more than the tracking derivative.

Ok, go to the **MainPage** tab, and click the large **Stop** button.
Now we'll play around with the slewing position PID values.
Move the fast slider until it moves the mount visibly. If you have low frequency error (visible on the mount axis), lower the integral gain slewing, or increase the derivative gain slewing. If you have medium frequency error, lower the proportional gain, or try increasing the derivative. If a medium to higher frequency hum, lower the derivative gain slewing. The idea is, after the acceleration period, the position error should be about an arc second or so, RMS.

It could be that you could revisit the **Tune Current** tab, and raise the derivative, so you can use a higher proportional gain in the position.

“Feed Forward”



“Acceleration Feed Forward”:

Don’t set this until you have everything working, with the final weight and balance. When moving slowly, watch or log the output, and take an average. Watch or log the output during acceleration. Take an average , and subtract the moving slowly output from the acceleration output. You should do this in both directions. Take the lower of the two directions, and put this in the Acceleration Feed Forward configuration item.

“Velocity Feed Forward”:

Don’t set this until you have everything working, with the final weight and balance. When moving slowly, watch or log the output, and take an average. Move at slew speed, and watch or log the output when it’s finished accelerating. Take an average, subtract the moving slowly output. You should do this in both directions. Take the lower of the two directions, and put this in the Velocity Feed Forward configuration item.

When you’re happy with it, don’t forget to:
IMPORTANT:
You’ve made changes to the sliders. The new values are ONLY in the Power Module. You must:
1. go to the Config tab, and click on **Read From Power Board**.
2. Click on **Write To File**.
3. If you have a flash rom chip, click on the “Write Controller to Flash” button.

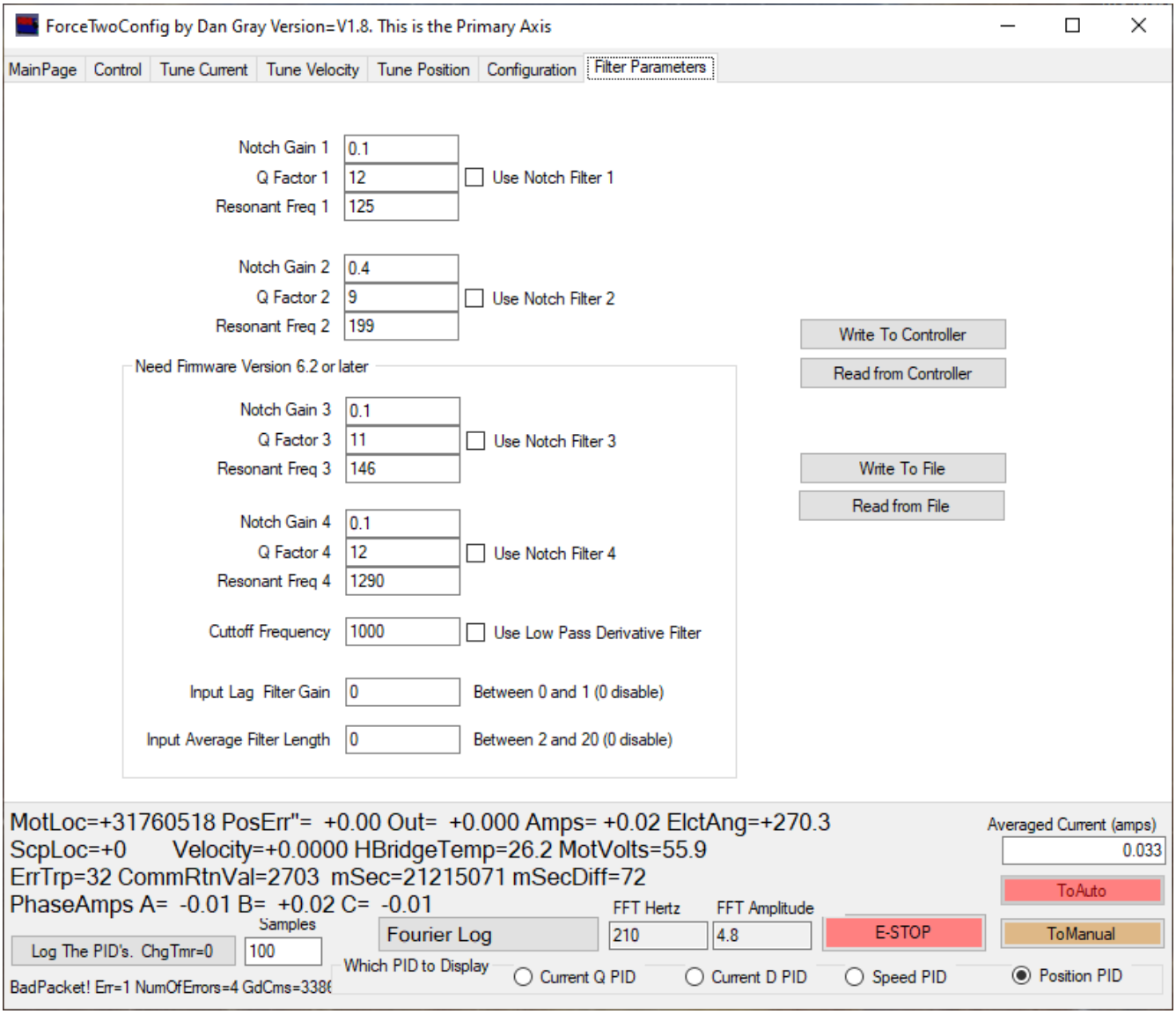
Once you’re happy with the tuning, save it to the filename:
/home/pi/ConfigFromPri.AxCfg for the primary axis, or
/home/pi/ConfigFromSec.AxCfg for the secondary axis.
Don’t mix them up!
These two files will be automatically loaded and sent to the Power Modules on startup of the CPU Module (Pi). Also, if you have a flash rom, save to the flash rom.

Setting the filters:

These filters are a powerful tool for setting the PID gains higher, for tighter control of the position.

If you have a direct drive motor, your telescope mount will probably have some mechanical resonant frequencies. In the prior section we tuned the PID's for the best stable control we could, so the pid's are probably the highest gains possible without filters.

Here's a screenshot of the "Filter Parameters" tab.



Notch Filters:

There are 4 notch filters.
Each filter has 3 settings:

1. Notch Gain
2. Q Factor
3. Resonant Frequency

Notch Gain:
This setting is how far to reduce the output if the frequency is centered on the resonant frequency. For best results, make this as high as possible and still eliminate the active resonance error.

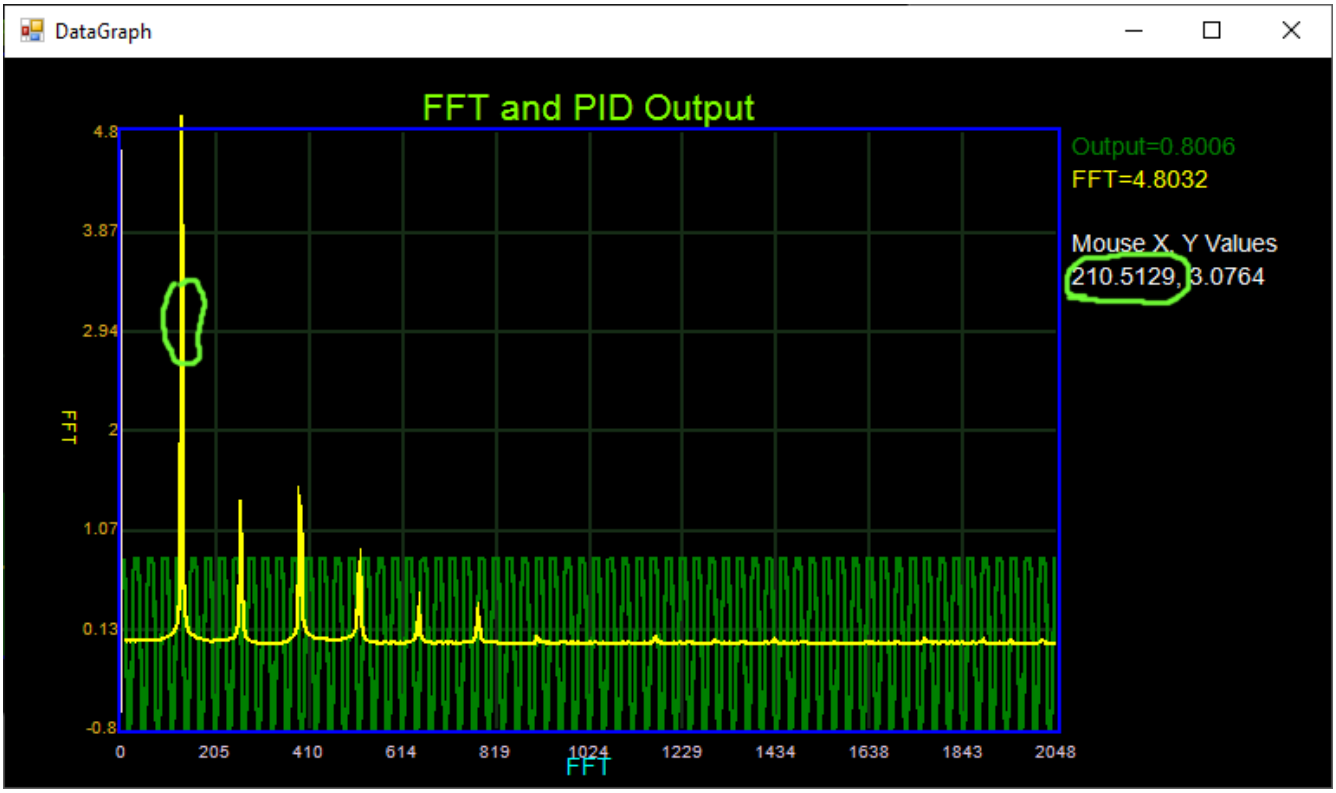
Q Factor or Notch Width:
In the notch filter I found online, they called it "Q". The documentation says "Q factor, Higher Q gives narrower band".
You may have to play around with this. I like to start out somewhere between 8 and 20. The higher the number, the better results, especially if using more than one filter.

Resonant Frequency:
Enter the resonant frequency here.

Setting the notch parameters.
Start with a Q factor of 15, Notch gain of 0.5.

1. First you need to find the first resonant frequency.
Make sure you're tracking at somewhere around the sidereal rate. There's a button on the "main page" tab for this.
2. Now using the tracking Derivative slider (on the "Tune Position" tab page), slowly increase until you hear a musical note. This will be your first resonant frequency.
3. While you hear the "note", near the bottom center of the Drive Info Area, click the button labeled "Fourier Log".

4. After a few seconds, a graph will appear, the green circle is the highest magnitude resonance. Wherever you put the mouse, the circle on the right will have the resonant frequency.
You can zoom in on this area by holding down the left button to the top left of the resonant frequency, and drag to the lower right. This will give you a more accurate resonant frequency.



Also, the textbox to the right of the Fourier log has the highest amplitude resonance frequency in it. You can copy this, and paste it in the Resonant Frequency text box.

5. Doing this Fourier log will put the Power Module in the safe mode. You'll have top re-enable from the Main Page.

So you've put the resonance frequency in the textbox, and have the 0.5 on the gain, and 15 for the Q factor (notch width).

6. Now put it back to auto, and start tracking again. You should hear the musical note again. While you hear the note, enable the notch filter you're working with, and click on "send to controller". The musical note should stop.

7. If this was a help, maybe save to file and flash.

For your second notch filter, try the same, but this time increase the tracking proportional. Hopefully, you won't need more than two. If so, 3 and 4 should be set similar to the above steps. When using more than two filters, it's important to keep the gains high, and the notch width narrow.

Input Filters:

Butterworth Low Pass Derivative Filter

The simplest filter is the “low pass derivative” filter. This will filter out noise on the derivative component. I wouldn’t go lower than 1Khz, because it could dampen the control. This filter is a Butterworth filter.

To set this filter, increase the tracking derivative as described above. When you hear a high note, you can play around with the “low pass derivative” filter, to eliminate the high note. As already mentioned, don’t go below 1KHz.

“Input Lag Filter”:

This is a filter based on the integral of the position error.
If the input lag filter Gain is 0, it’s disabled. 0.99 would be a very weak filter. 0.01 would be a super strong filter, and probably wouldn’t work. You could try this in some instances.

“Input Average Filter Length”:
This is a moving average filter. It averages the last N motor encoder locations. This could help in some instances.

Interpreting the LED’s:

Power Module Main Board:
There are two LED’s adjacent to the motor terminal block.
The Green LED will be on steady when the controller is in **AUTO**.
The Green LED will be flashing when the controller has no errors, but is in **MANUAL** (or Torque mode).
The Green LED will be OFF, and the Red LED will be flashing if there is a problem with the Power Module. The fault(s) will be displayed on the **MainPage** of **ForceTwoConfig.exe** OR will be displayed on the **Features / Controller Stuff** in **SiTechExe.exe**.

Power Module CPU:
On The Power Module CPU, there are a blue LED and a red LED on the extreme right side. If the red LED is flashing fast, there is communications being sent from either **ForceTwo** or **ForceTwoConfig.exe**.
If the blue LED is flashing fast, the Power Module CPU is responding to the commands from **ForceTwo** or **ForceTwoConfig.exe**.

There is a Green, Red, then Blue LED about 3 CM from the right hand edge of the Power Module CPU. The Green and Red should be on steady. The Blue should be flashing, and indicates the program is running.

CPU Module (Pi) HAT:

The LED near the top right labeled **Primary** is the indicator for the Primary Axis Status. On Steady it's in **AUTO**. Flashing, it's in **MANUAL** (Torque Mode).

The LED near the top right labeled **Secondary** is the indicator for the Secondary Axis Status. On Steady it's in **AUTO**. Flashing, it's in **MANUAL** (Torque Mode).

When the LED labeled **USB** is flickering, there is communication between **SiTechExe.exe** and the CPU module (Pi).

The Red LED labeled **Slew** will be on when the CPU Module (Pi) is in the "Slew" mode (Fast Speed).
The Yellow LED labeled **Pan** will be on when the CPU Module (Pi) is in the "Pan" mode (Medium Speed).

The Green LED labeled **Guide** will be on when the CPU Module (Pi) is in the "Tweak" mode (Slow Speed).

Handpad Operation:

For the handpad to work, the **ForceTwo** software must be running on the CPU Module (Pi). You can always reboot the CPU Module (Pi) to start it up.
The **SPD** key will switch between **Slew**, **Pan**, or **Guide** (Tweak) mode.
To put the controllers from manual (torque mode) to Auto, you can press the **ESC** and **RTN** buttons simultaneously.
To move the scope, press the arrow keys. The **up/down** is for the secondary axis, and the **left/right** is the primary axis.

Upgrading the software on the CPU Module (Pi):

Note about versions… Never downgrade a piece of software, unless directed by SiTech support staff.

There are two pieces of software for the ForceTwo system on the CPU Module (Pi).
1. **ForceTwoConfig** (Current version is 1.2a).
2. **ForceTwo** software (Current version is 1.2)

To upgrade the **ForceTwo** software:
there are two parts to this software, the source code and project, and also the executable.
To recompile, just don't do it!
But if you must, and you know what you're doing, make a full backup of the directory before starting.
The directory is here:
/home/pi/C/ForceTwo

Here's what I do to run codeblocks:
Codeblocks MUST be run using sudo, if you're going to be debugging. In order to do this, at the terminal, you may have to run the command:

xhost +

then at the terminal type:

cd /home/pi/C/ForceTwo
sudo codeblocks.cbp

Be Careful and Have Fun!

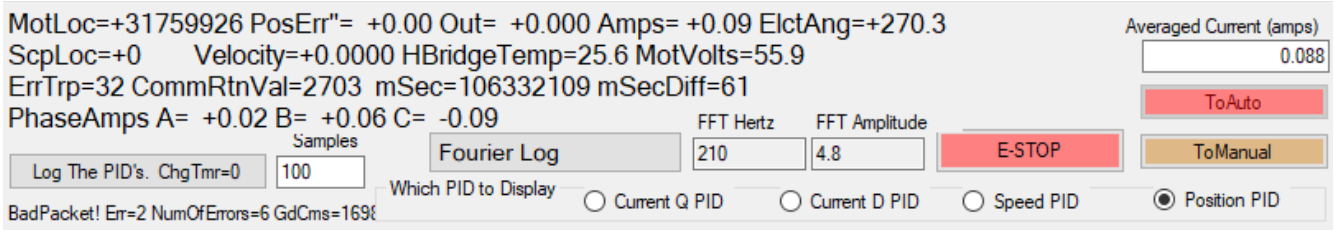
The Executable must be in the **/home/pi/**
director. If you compile the source code using codeblocks, the executable will be in
/home/pi/C/ForceTwo/bin/Debug/
You'll have to copy it to **/home/pi/**
so when the CPU Module (Pi) reboots, it will run the proper program.

More than likely, it will be a change initiated from our side.
In order to upgrade:
1. If you choose (it may be a good idea), make a backup of the source and executable files.
2. Replace the source files with the files we provide. The current source files are here:
/home/pi/C/ForceTwo/
3. Replace the executable file **ForceTwo** in **/home/pi/** with the new version.

To upgrade the **ForceTwoConfig.exe** file:
1. Make a backup of the current file in **/home/pi/**
2. Replace it with the provided **ForceTwoConfig.exe** file from us.

Appendix's

Appendix A Description of Drive Info Area,:



Line 1:
Loc= is the motor encoder location, raw value.
PosErr= is the deivation in arc seconds between the desired motor position and the actual motor position.
Out= is the output in percentage, -100.0 to 100.0
Amps= is the absolute highest phase amperage.
ElctAng= is the calculated motor electrical angle, based on your configuration, and the motor encoder.

Line 2:
ManVal= is the manual value (Torque Value) in -100.0% to 100.0%. It will be zero if in Automatic.
Velocity= is the motor velocity, based on raw ticks.
HbridgeTemp= This is the temperature (Deg's C) of the 3 phase H Bridge driver chip. The trip value is set by the configuration parameter labeled "High Temp Trip (Degs C)". The default is 60. The cooling fan will turn on at 37 Deg's C and turn back off at 35 Deg's C.
MotVolts= is the voltage at the motor power supply at the input terminals.

Line 3:
ErrTrp= is the binary equivalent of any errors. See below for a description.
CommRtnVal= When this software or the ForceTwo software sends a command that doesn't require a return value, this value is set to the version number. Otherwise, it returns the value that is asked for. Displayed in this value is the last value that was asked for, in other words, the version number is never displayed (except on the top right of the "MainPage" tab);
mSec= The Power Module has a built in mSec clock. On reset, it starts at 0. This will always change on the display, unless there is no communication.
mSecDiff= is the communication time difference in mSecs.

ErrTrp= description. The bit is the binary AND equivalent value:
bit 001=Gate Volts Low
bit 002=Instantaneous OverCurrent Hardware Trip
bit 004=Instantaneous OverCurrent Firmware Trip
bit 008=Motor Volts Low Trip
bit 016=Over Temperature (Hbridge) trip
bit 032=Drive is not enabled
bit 064=Positive direction limit switch activated
bit 128=Negative direction limit switch activated
bit 256=No configuration in the Power Module yet.
bit 4096=Timed OverCurrent Firmware Trip
bit 8192=Position Error Trip
bit 16384=BiSS Absolute Encoder Error Trip
bit 32768=Checksum Error (between Power Module and this software or ForceTwo software)

Appendix B, Specifications:

Controlled by two Texas Instrument DSP processors, running at 200 Mhz (one per axis).

PWM frequency is > 20 Khz, above the audible range.
Sample Rate 6.6666Khz
PWM resolution 12 bit, (+/- 2048)
Maximum motor supply voltage is 250 Volts DC
Maximum Peak Current is 30 amps.
Maximum Continuous current is 20 amps.
Typical RMS jitter with a direct drive motor at tracking speeds, < 0.05 arc second
Typical RMS jitter with a direct drive motor at LEO satellite speeds, < 0.5 arc second

Hardware inputs for each axis:
1 Renishaw BiSS Absolute Encoder (26 bit or 32 bit) or Heidenhain Endat2.2
2 High Speed Increment Encoder inputs (RS422)
Limit Plus
Limit Minus
Periodic Error Worm Sensor
Homing Switch Sensor
1 0-5 volt analog input

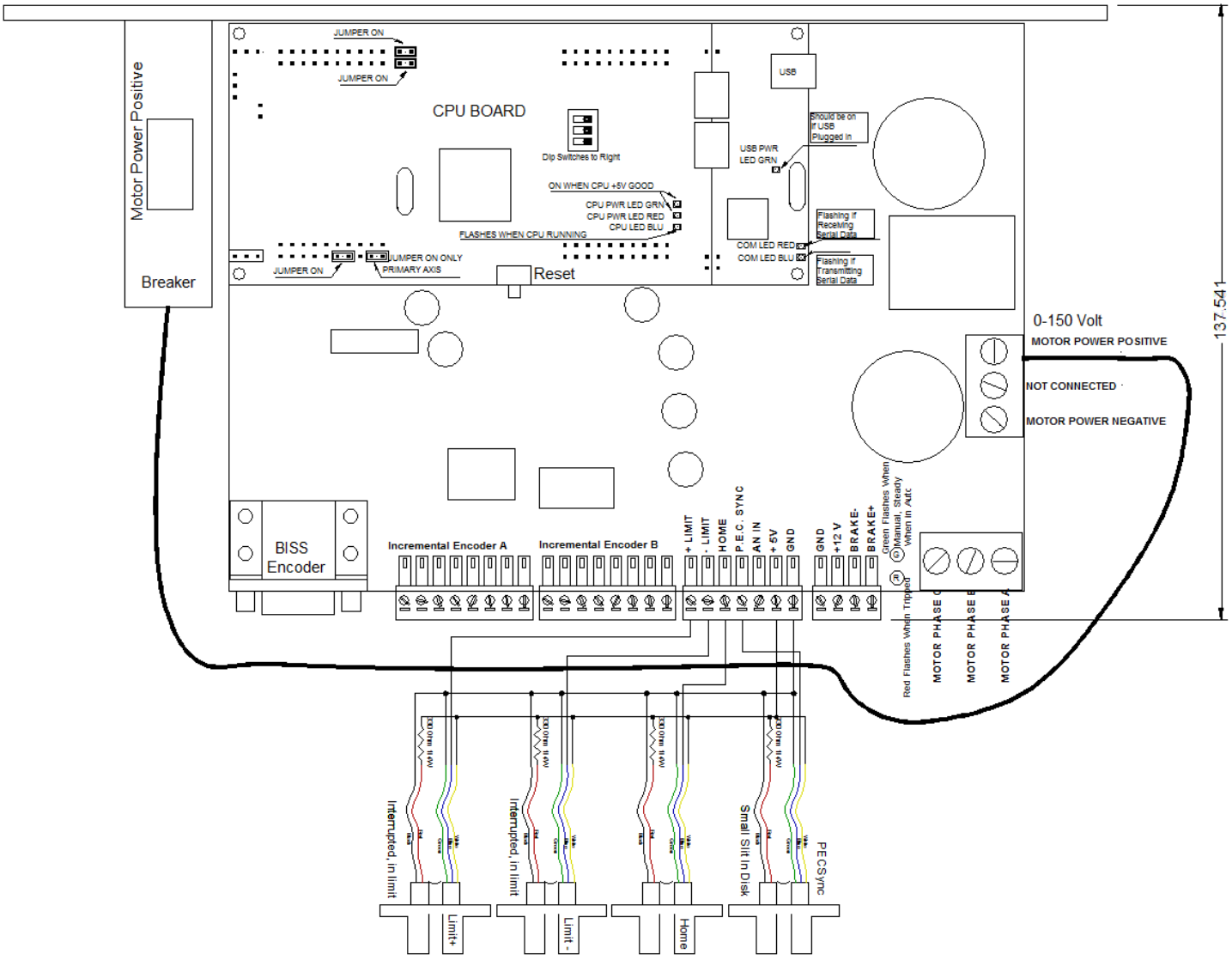
Hardware outputs:
Brake control
Motor output (single phase or 3 phase)

PID Loops per axis:
Position Loop
Velocity Loop
Current Loops (2)

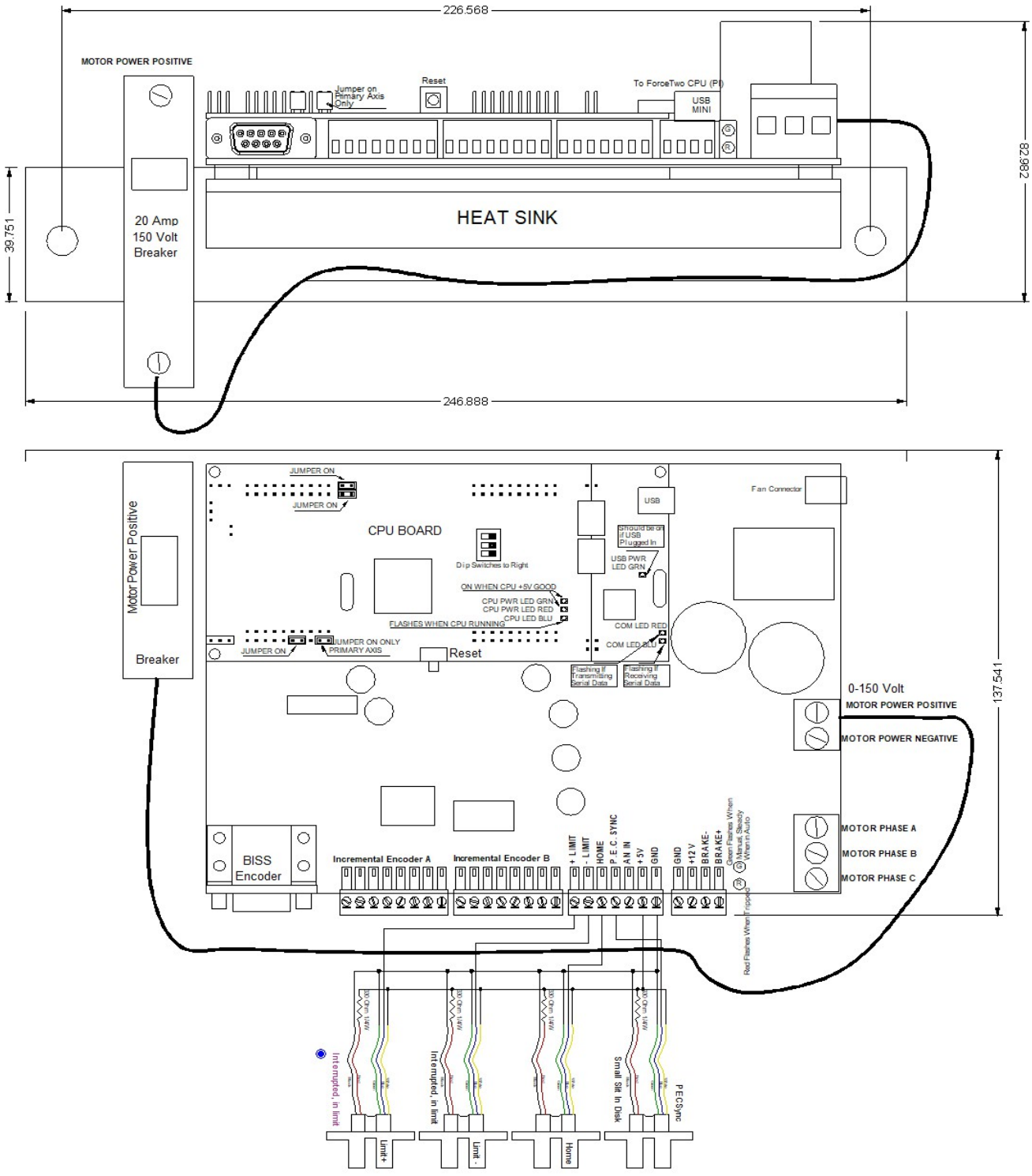
Position Loop Filters per axis:
Notch Filters (4)
Low Pass Filters (3)
DeadBand Filter (1)

Appendix C: ForceTwo Drawings

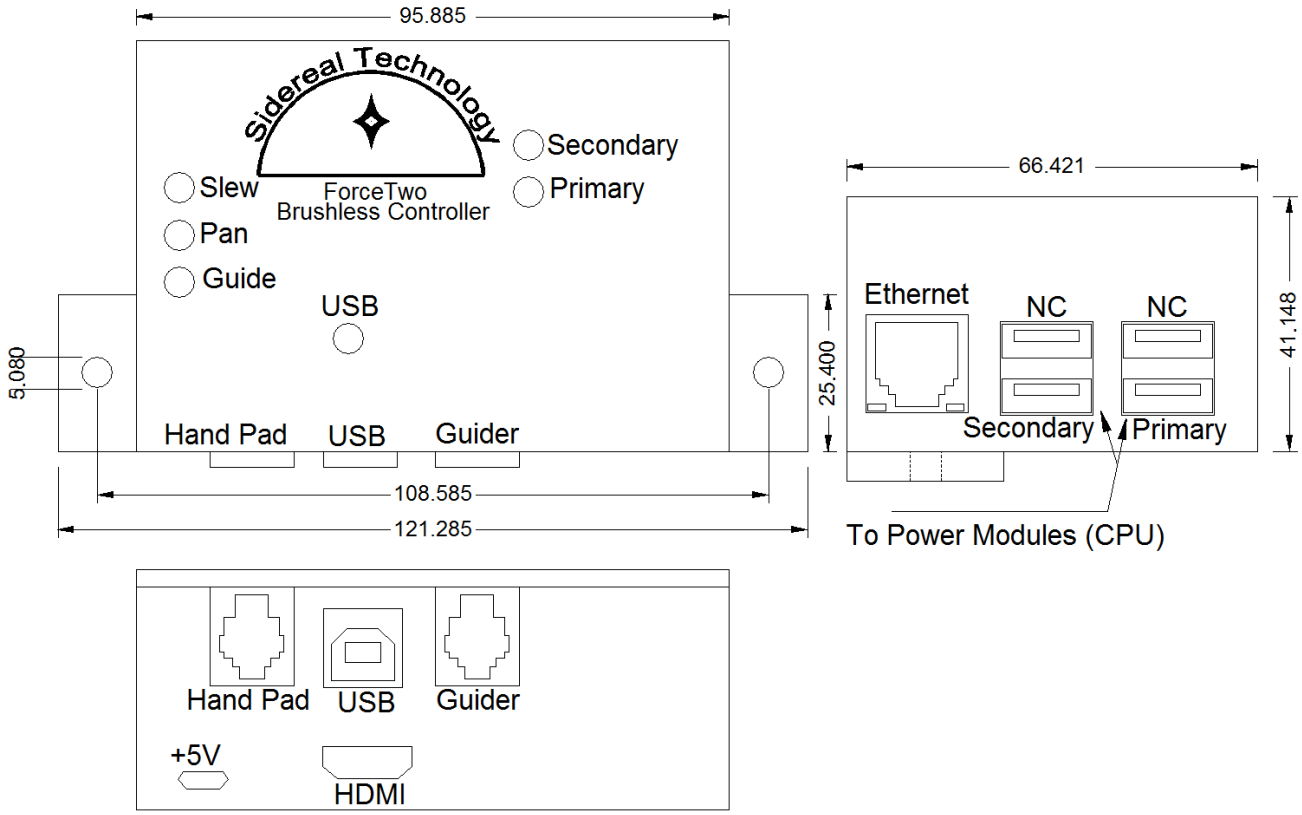
Power Module (two per system) (Dimensions in millimeters): Version prior to Version 2:



Version 2 Power Module:



CPU Module (Raspberry Pi)(Dimensions in millimeters):



Appendix D Upgrading the firmware:

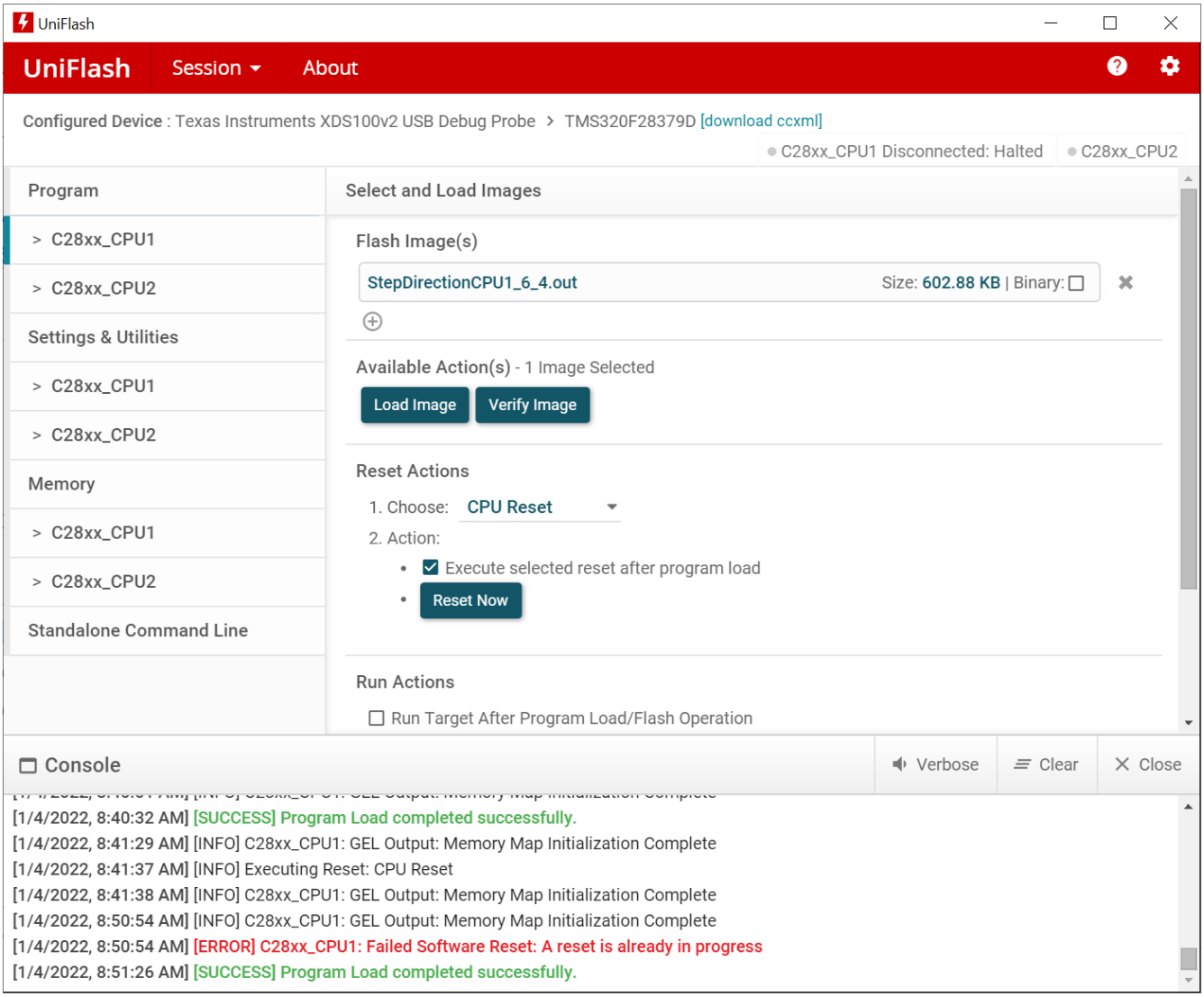
Important! You need to be in communication to our office to receive the correct firmware files.
There are firmware files for the CPU1 and the CPU2 on the ForceTwo Power Module CPU.
If you install the wrong firmware, it could be possible to destroy the FET Hbridge Chip.

This needs to be done using a Windows computer.
For the USB to serial driver, you need to download and install this:
https://software-dl.ti.com/ccs/esd/documents/xdsdebugprobes/emu_xds_software_package_download.html

Plug the USB to cable between the powered up Power Module CPU and a USB port on the Windows computer.

First, you need to download and install Uniflash:
http://software-dl.ti.com/ccs/esd/uniflash/uniflash_sl.5.0.0.2289.exe

Run Uniflash. It looks like this:



Under the "Choose Your Device" select "TMS320F28379D"
Under "Choose Your Connection" select "Texas Instruments XDS100V2 USB Debug Probe"
Click the start button.

IMPORTANT: Make sure you don't put CPU2 firmware in CPU1 or vice versa.
Updating CPU2:
Under program, click on C28xx_CPU2.
Now over on the right side, click on the "Browse" button, and select the CPU2 firmware image.
Now click on "Load Image"
Watch the status at the console, near the bottom.
Once successfully done, it should say with green text "[SUCCESS] Program Load Completed successfully"

Updating CPU1
On the right side, click on the "Browse" button, and select the CPU1 firmware image.
Now click on "Load Image"
Watch the status at the console, near the bottom.
Once successfully done, it should say with green text “[SUCCESS] Program Load Completed successfully”

Power the board down, then back up.
That should do it.