# **Using External Mode with the Pixhawk PSP**

# **What is External Mode?**

Host PC

Communication Channel

Simulink Model

External

Mode

Client

(ExtMex)

External Mode

Server

Target

Generated Code from Simulink Model

External mode is a way in which one can examine execution and “tune” the generated code from your standard desktop environment. This feature comes with Embedded Coder and can greatly enhance debugging capabilities or interactive testing through tuning of model parameters. For instance, suppose you wanted to know the output value of a certain signal at each time step of the simulation. One could implement this by modifying the generated code to execute a “printf” of that signal value manually or create a block to do this (we actually provide this for you), however, this can be cumbersome. External mode can alleviate this by not only displaying the value of the signal from the generated code in a scope in the Simulink environment, but it also enables you to log that signal and record it to your host PC as a MAT file as well. One way to think of external mode is that your Simulink model now becomes a “user-interface” that interacts with the generated code during execution, instead of the play button playing a simulation on your host machine the play button now begins execution of embedded code.

This mode is not to be confused with Hardware-in-the-Loop or Processor-in-the-Loop.

External mode works by establishing a communication channel between the host PC and the embedded system you are targeting. MathWorks provides all the necessary source files for both TCP/IP and Serial communication – in the case with the Px4FMU we are using serial communication.

The generated code and the Simulink infrastructure have the necessary source files and shared libraries to do this for you without having you manually modify the code itself.

For more information, please see:

<http://www.mathworks.com/help/rtw/ug/external-mode-communication_f1028013.html>

## **Some important considerations:**

External mode can be a potentially taxing process since it needs to transfer data back and forth to the PC as well as wait for user commands to start/stop execution of the model as well as tuning of parameters. Because of the limited resources on the Px4FMU, it is highly recommended to NOT use external mode for flight tests. Due to limited RAM on the system, the amount of data transferred back must be reduced considerably at each execution time-step, therefore, some special modifications to some user settings must be made. More detail will be covered on this down below.

Note that if you are just interested in logging data rather than viewing data as the system is running and tuning parameters then you may want to consider data logging. A Simulink block is being developed for this in the Pixhawk Library browser.

**Important Note**: Recent testing has confirmed that external mode does not operate correctly with the telemetry radios. Data comes in very sporadically and long time intervals. We are looking into this and will provide an update to address this, in the mean-time, please use USB for serial connection for external mode.

# **External Mode Tutorial Part 1:**

# **Viewing Signals from the generated Code**

### **Serial port setup:**

Before running External Mode, you should be familiar with the serial ports offered by the hardware. The Px4FMUv2 comes with a variety of different options. To see which serial ports you may want to use consult these pages here:

<https://pixhawk.org/users/wiring>

<https://pixhawk.org/dev/wiring>

Note: some serial ports are reserved and cannot/should not be used, such as ttyS0 which is a serial debug console port. Please consult above links for information on this.

Suppose you wanted to use the standard USB programming cable for external mode. In this case, ttyACM0 becomes a free serial port to be used by external mode and ttyS6 be a serial console shell. Here is one example of one would modify their rc.txt file to do this:

#sercon - in the latest release sercon already gets called

usleep 1000

uorb start

usleep 1000

#nshterm /dev/ttyACM0 & #Disable the USB serial console

usleep 1000

px4io start

usleep 1000

#commander start

#usleep 1000

#mavlink start -d /dev/ttyS1 -b 115200

#usleep 5000

#dataman start

#usleep 1000

#navigator start

#usleep 1000

sh /etc/init.d/rc.sensors

usleep 1000

#sh /etc/init.d/rc.logging

#usleep 1000

#gps start

#usleep 1000

#attitude\_estimator\_ekf start - the latest release does not use this anymore

ekf\_att\_pos\_estimator start

usleep 1000

#attitude\_estimator\_so3 start

#usleep 1000

#mavlink start -d /dev/ttyS3 -b 115200

mtd start

usleep 1000

param load /fs/mtd\_params

usleep 1000

rgbled start

usleep 1000

#px4\_simulink\_app start #disable automatic starting up of the model

nshterm /dev/ttyS6 & #this will make ttyS6 the serial console

exit

In the above example, ttyS6 (serial port 4) and the USB port we typically use to access the serial console is now a free serial port we can use to establish data exchange in external mode. Note that in the example I am doing in this tutorial, I have ttyS6 connected to a USB FTDI connector which will allow my Windows machine to access the serial shell over USB. Other options could include using the telemetry radios to establish external mode which would be different serial ports than the ones mentioned here.

### **Model Configuration Setup:**

The next step after choosing a serial port is to now configure Simulink to use the port of choice. Open up the model

ext\_mode\_intro.slx

## 

To configure Windows to talk to the Px4FMU, click on the interfaces section in Code Generation, then click on Interface, then under MEX-file Arguments you want to type in the following syntax

1 <COMPORT#> <BAUDRATE> <TIMEOUT>

The ‘1’ value in the front is to indicate that the generated code will wait for the host before starting execution. This is highly recommended.

<COMPORT#> - this is the COM port of the USB connection which connects the PC and the Px4FMU

<BAUDRATE> - Specify a baud rate here. Note that radios are set to run on 57600

<TIMEOUT> - Specify the amount of time it takes to wait for establishing a connection before timing out. 5 seconds should be sufficient

Next, we want to configure the generated code to use the correct serial port on the firmware. In this example, this is ttyACM0

## 

Click on the Coder Target menu and then select external mode options. Click on the Baud Rate Select button drop-down to select a baud rate and specify the data exchange serial port as ‘ttyACM0’

### **Running the example model**

The next step is to run the model. This involves a couple of steps:

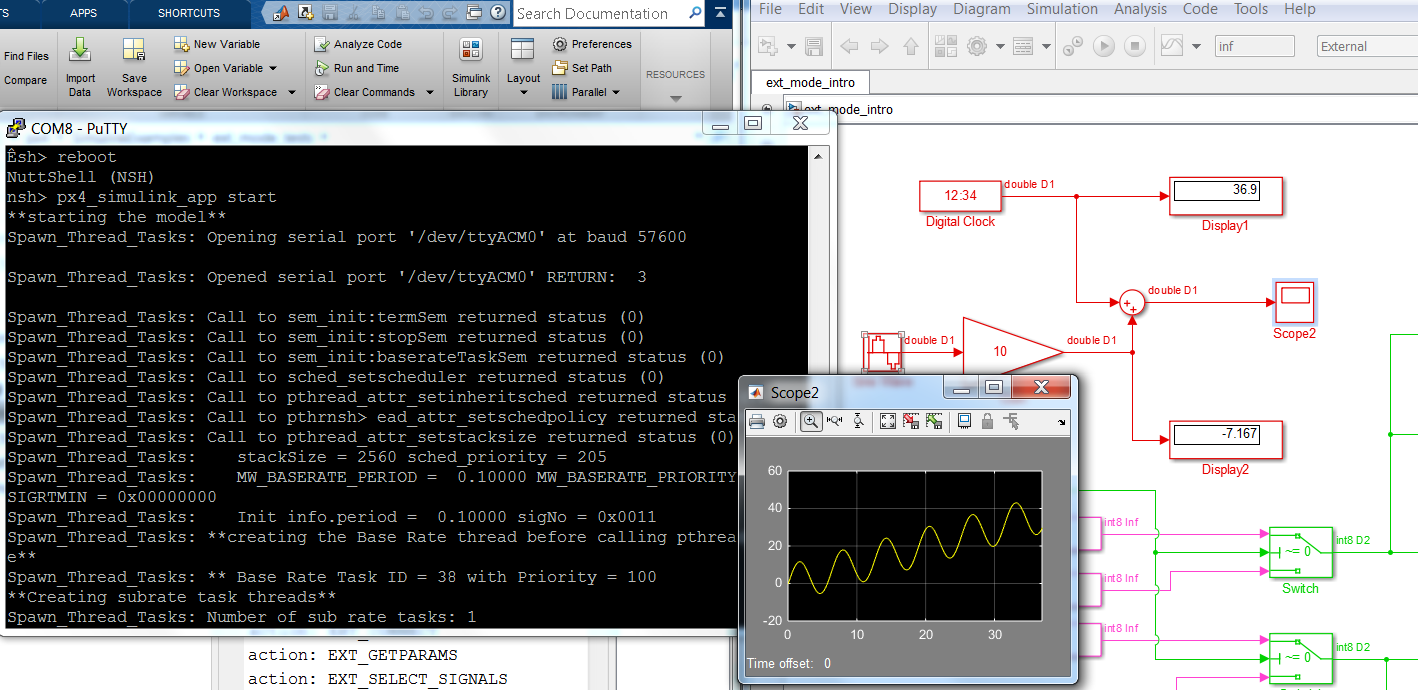
1. Pressing the build button (ctrl+B) to generate code, compile it, and then having it transferred to the PX4FMU
2. Running the model. Open up the serial shell to access the command line and start the model by typing ‘px4\_simulink\_app start’

The model will not actually begin execution yet until receiving a start packet from external mode, this is done in the next step

1. Ensure that the drop-down menu is set to ‘External’



After, click on the “Connect To Target” button on the left of the above diagram. This will establish connection after a few seconds. Finally, press the green play button to the right. You will now be able to see data appear in the Simulink model in the generated code.

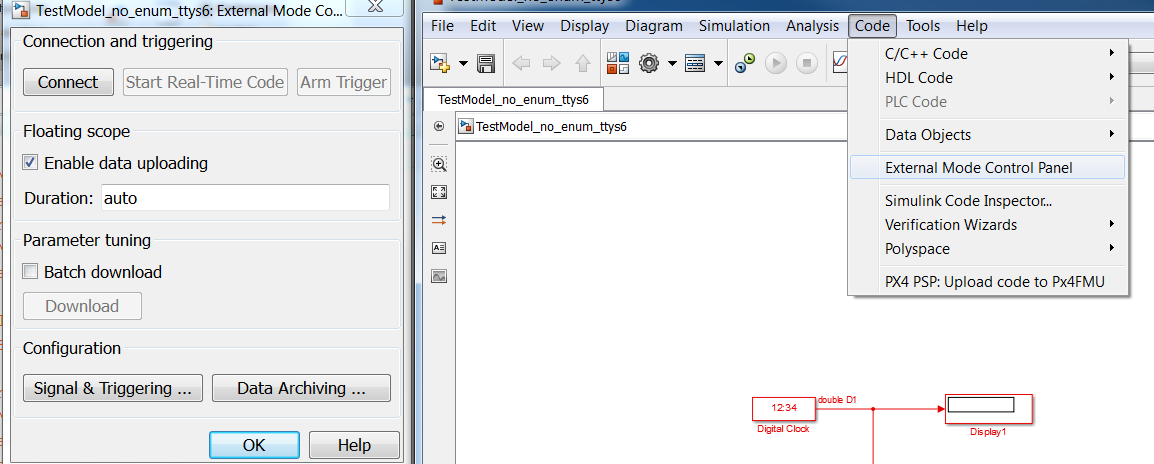


1. To stop the generated code, press the stop button in the model. **If you wish to run external mode again, type ‘reboot’ at the serial console as well**. There is currently a limitation in which external mode code has some issues being treated as re-entrant. This will be fixed in the future – for now, **please reboot the Px4FMU in-between external mode runs.**

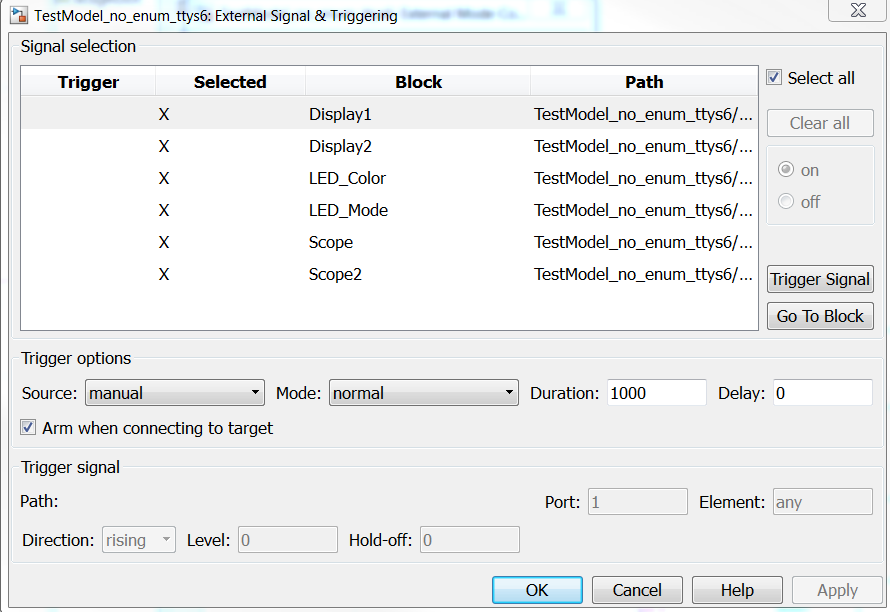
## **External Mode Tutorial Part 2: External Mode Control Panel**

As mentioned earlier, the PX4FMU runs on a resource constrained environment. External mode can be process intensive at times depending on the circumstances. For instance, if you have a model which contains many (10-30) scopes the amount of data transferred can impact real-time performance. To assist with real-time execution of code, external mode has been automatically configured to run as a back-ground task by default. This option appears in the External Mode Control panel and doesn’t need to be changed by the user.

Due to limited memory, the amount of data that can be transferred back needs to be adjusted. This setting dictates a buffer size on the Px4 for each signal you are logging.



When clicking on the “Signal & Triggering” button you get access to all the signals that are attached to display ports. Signals marked with an X are available for viewing in external mode which you can enable/disable with the “ON” and “OFF” radial button on the right. The diagram below has the “select all” option enabled so all display/scopes will be viewable.



The ‘Duration’ value specifies the buffer size to send back to Simulink. The default value of 10000 is rather large, so in this example we have scaled it back down to 1000. A large value can impact performance of external mode – the real-time code can still run but external mode may not report back on time. Since external mode runs on a lower priority task the generated code from Simulink takes more precedence and can potentially delay external mode’s transfer of data to the host machine. To avoid this, choosing a low duration value such as 1000 or 500 samples can help greatly.

Duration value dictates how many samples you’ll see at each time step in the Simulink model. Duration value does not apply to all sample times, since it is based on the base-sample rate. In other words, choosing a value of 1000 means one will receive 1000 data ‘ticks’ of base-rate data. Here is an example:

Suppose we have a model and its fastest-sample time is 250 Hz (Ts = 0.004 seconds) and a slower rate 25 Hz (Ts = 0.04 seconds)

Suppose we choose a duration value of 400:

This means if we hook a scope up to a 250 Hz signal we will see 400 sample points appear at a time in external mode. Or in other words, (400\*0.004 = 1.6 seconds) 1.6 seconds of data will appear at a time.

For the lower sample rate (25 Hz), 1.6 seconds of data yields 40 data points (1.6/0.04 = 40 ) at a time rather than 400

One should be careful to not choose a duration time that is too low. For instance, suppose I chose a duration time of 1. This means:

250 Hz: Any scope hooked up to this will give you one sample at a time

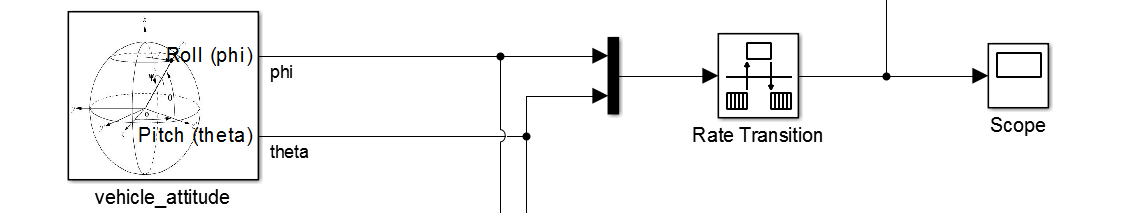
1\*0.004 = 0.004 seconds of data

0.004 / 0.04 = 0 (round to floor)

A duration size must be selected which is big enough such that each time external mode delivers data back to the PC it has enough sample points for signals running on a lower sample-rate. If we chose a duration value of 1 only the 250 Hz signal will display data and the other signals will show no data (default value of zero)

### **Tips for logging fast-rate signals**

Sometimes, depending on the model, there will be a noticeable delay when receiving data. This can occur if the signal if you are trying to log is running at a fast sample rate. External mode may deliver data back to the PC at a slower than desirable rate. One way to get-around this is to use a rate transition block to view a signal running at a fast rate at a slower rate. You will get less data back but this will help alleviate slow-downs – considerations and trade-offs will need to be made depending on the situation



In this example I specify a sample rate of the sensor block to 250 Hz.

If you just want sensor data, you can just hook up a scope and start logging away, however, keep in mind that you will not get real-time performance. This means that the Simulink timer assosicated with the base-rate ticks will not keep up with real-time. Data coming in from the sensors will still update/react to physical stimuli but the time stamp of data will not be in real-time (ie: 2-3 times slower). Other sample rates will suffer as a result and not run on schedule. Performance is affected at the cost of getting more data back into the host PC.

Adding in the rate transition to slow down the sampling will help greatly as shown above.

## px4demo\_attitude\_system\_multi\_task.slx

### **Background Task vs Normal EXT Mode Behavior**

With background tasking, the base-rate task is not encumbered by having to run external mode tasks at the same time, instead, EXT mode is now running in the background on another task being paced at 10 Hz. This can greatly help with real-time performance. Sometimes you may want to use the default behavior if you only have a single sample-time running at a slow rate - if you want data to come in faster or as fast as the model's sample-time then you'll want to use default behavior. On the other hand, if your model is running with multiple rates at varying speeds, you will want to pick the back-ground task option

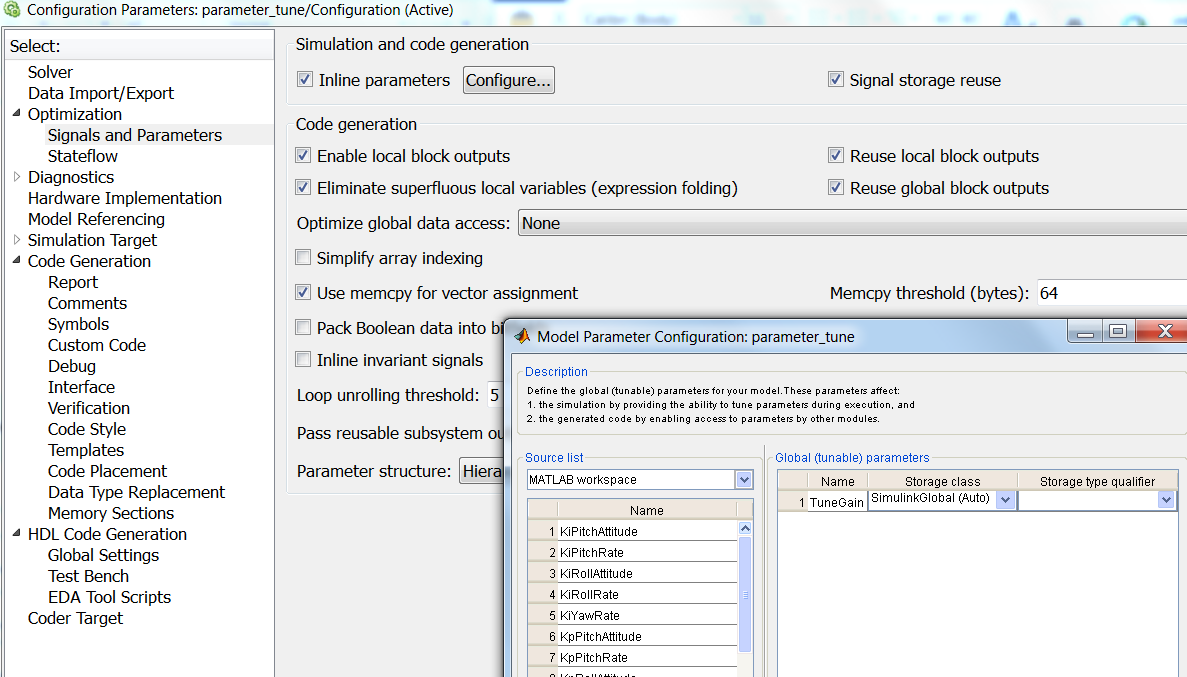
## **External Mode Tutorial Part 3: Tuning parameters**

Tuning parameters is one way to adjust values of the generated code Simulink while execution. In order for this to be possible, parameters must be set to tunable. Tunable parameters allow parameters to be exposed in the generated code as a global value such that an external program can have more control over it rather than having it inlined.

Due to resource constraints in global memory, parameters all Pixhawk PSP models are set to inline and will error out if you disable the inline parameter option. You can, however, select certain parameters to allow for tuning. This is done by clicking on the configure button in this menu

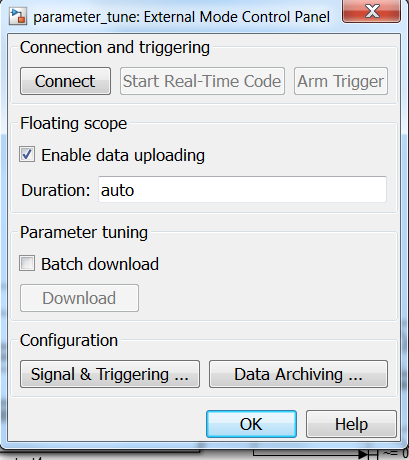
Configuration Parameters -> signals and parameters -> inline parameters -> configure

The example model for this section is : parameter\_tune.slx

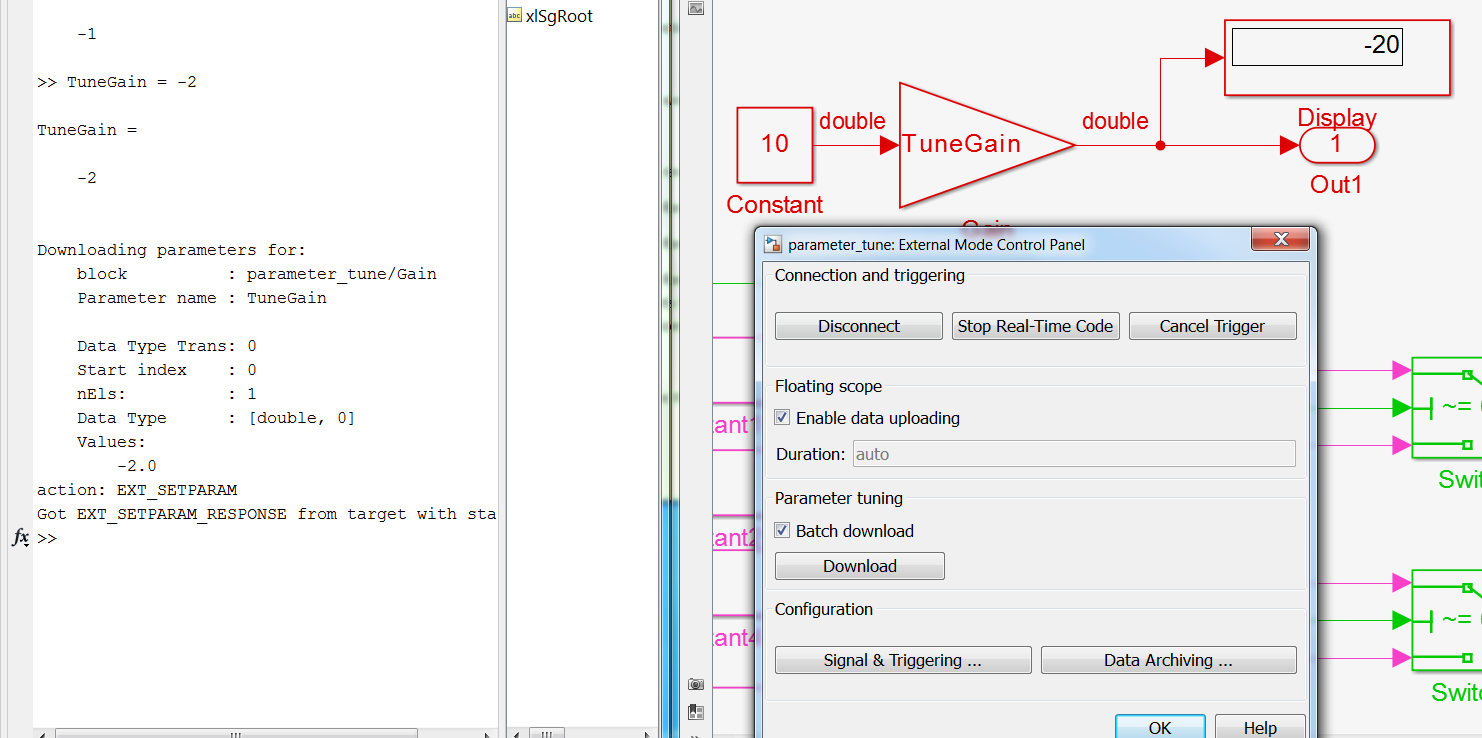


From this menu, you can now select parameters you wish to tune. Note that a parameter that requires tuning must have a parameter which exists in the base-workspace. In this example, the parameter is ‘TuneGain’ which has been selected.

To change the value of this parameter while the Simulink model is running, change the value in the base MATLAB workspace, to update the model and then bring up the External Mode Control panel



Now, press the “Download” button to push updated parameter values onto the generated code with “Batch download” selected



If you have multiple parameters that change, clicking the download button will also update those parameters in one-shot. In the above example I change the TuneGain from -1 to -2 and the generated code receives this update immediately afterwards. **Note a faster method is to also update the model diagram by pressing CTRL+D after changing a parameter in the MATLAB command prompt.**