Real-Time Windows Target, Simulink, and the ECP Model 210

The Long Version

Every station in C-116 should be set up to run the ECP Model 210 from Simulink using the Real-Time Windows Target software. However, it often takes a little effort to make sure all components are communicating with each other. This is a brief guide to help you get started.

Step 1. First of all, *you must be using Matlab 6.5.1 to use the real-time windows target with the ECP systems.* Once you have started the correct version of Matlab, we need to get some communications started.

Step 2. To install Matlab's Real-Time Windows Target, type

rtwintgt –setup

Matlab should respond with

You are going to install the Real-Time Windows Target kernel.
Do you want to proceed? [y] :

Since the whole point is the use the real time windows target, you should type **y** and hit the enter key.

At this point, Matlab will respond with

The Real-Time Windows Target kernel has been successfully installed.

The real-time windows target usually remains installed, so after you type
rtwintgt –setup you may get the message

The current version of the Real-Time Windows Target kernel is already installed.
Do you want to reinstall it? [n]

You can just type **n** (and enter) if you get this message, since you do not need to reinstall it.

Step 3. The real-time windows target utilizes Microsoft's Visual C++ compiler. We first need to tell Matlab to use this compiler. To do this, type

mex -setup

Matlab will respond with the message:

Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n?

You need to type **y** (and enter)**,** since we want Matlab to identify the available compilers

Matlab then responds with

Select a compiler:
[1] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft Visual Studio
[2] Lcc C version 2.4 in C:\MATLAB6P5\sys\lcc
[3] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler:

You need to type **3** (and enter)**,** since we are going to use Microsoft's Visual C++ Compier

Matlab then responds with

Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files\Microsoft Visual Studio

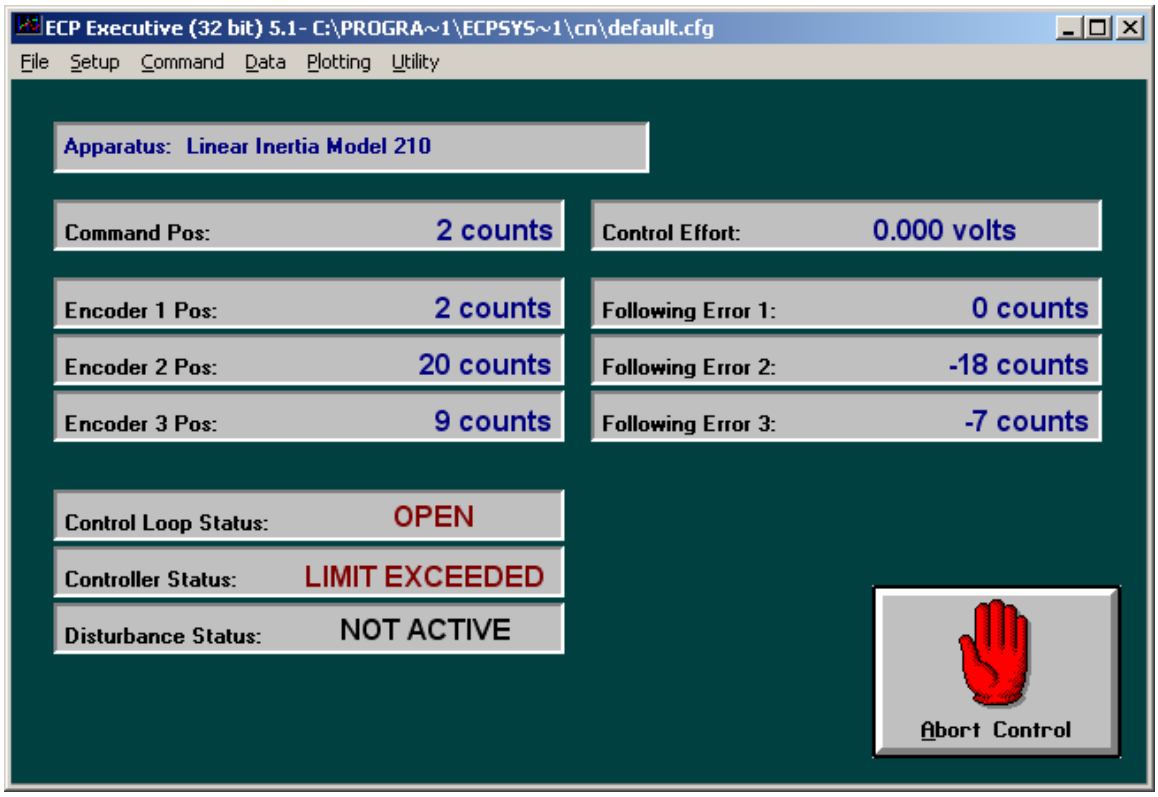
Are these correct?([y]/n):

And you should respond with a **y** (and enter)

At this point Matlab responds with a long-winded reply which basically means it's done.

Step 4. Now we need to inform the ECP system that we will be using Simulink and the real-time windows target. To do this, click **Start -> Programs -> ECP**

The ECP window should pop up, something like the window shown below:



First select on **Utility-> Download Controller Personality File.**

Then select **C: -> Program Files -> ECP Systems -> cn**

Finally select **m210_rtwt_3.pmc** and click on **open**. Wait for the ECP system to load the personality file.

Step 5. Close the ECP Executive **(click on the X)** ***Do not just minimize it!!!!***

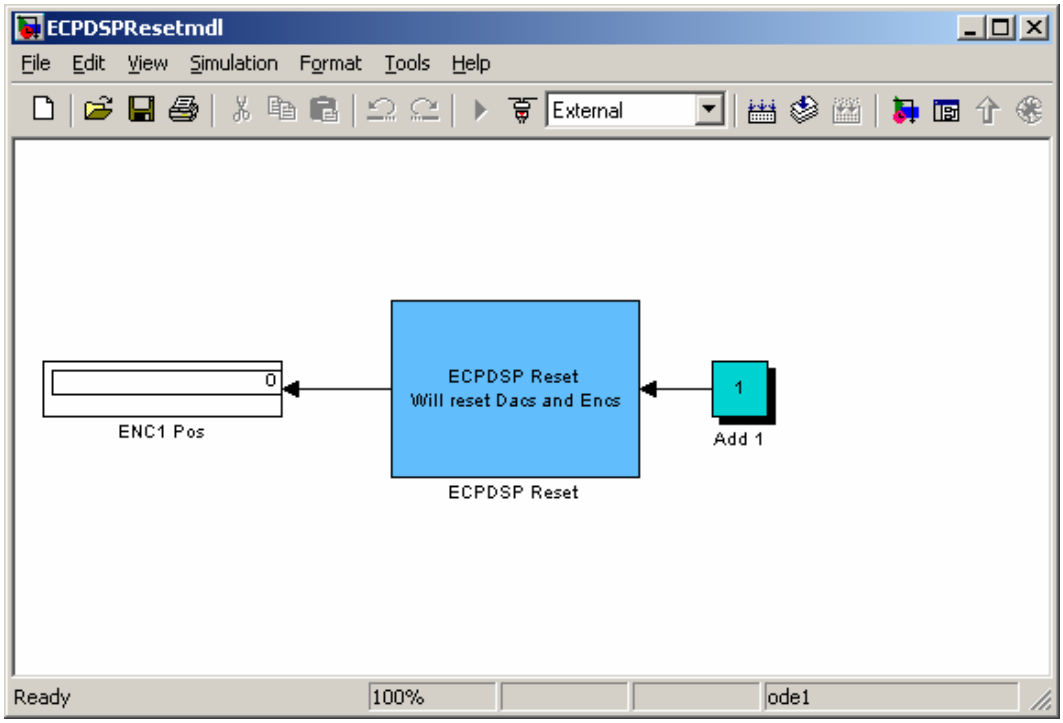
Step 6. Set the Matlab **current directory** to the folder where all of your files are located.

Step 7. Start Simulink by typing **Simulink** in the Matlab window.

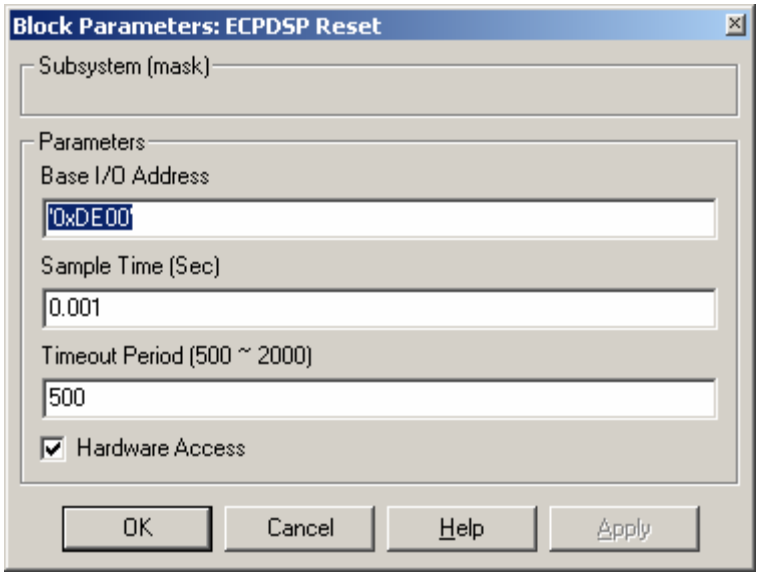
Resetting the System Using Simulink

We often have to reset/reinitialize the system. In order to do this from Simulink we use the **ECPDSPresetmdl.mdl** Simulink model file. This model is our friend, you will probably use it a lot.

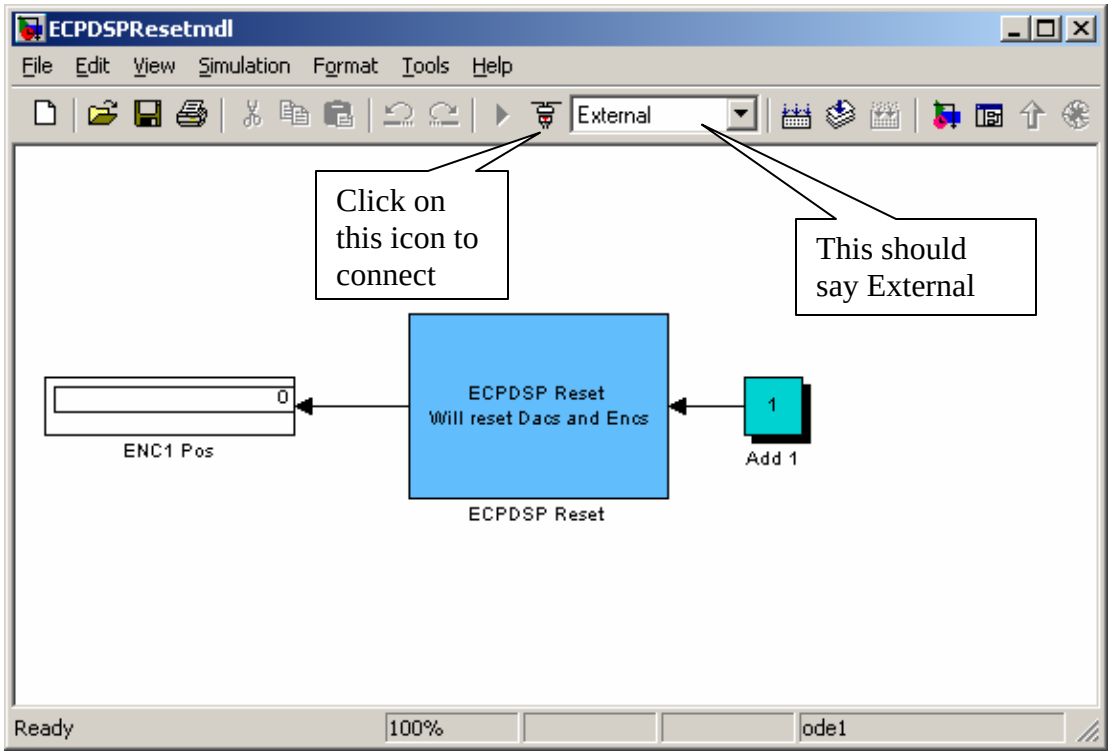
Step 1. From Simulink select **File->Open->ECPDSPResetmdl.mdl** The following window should pop up.



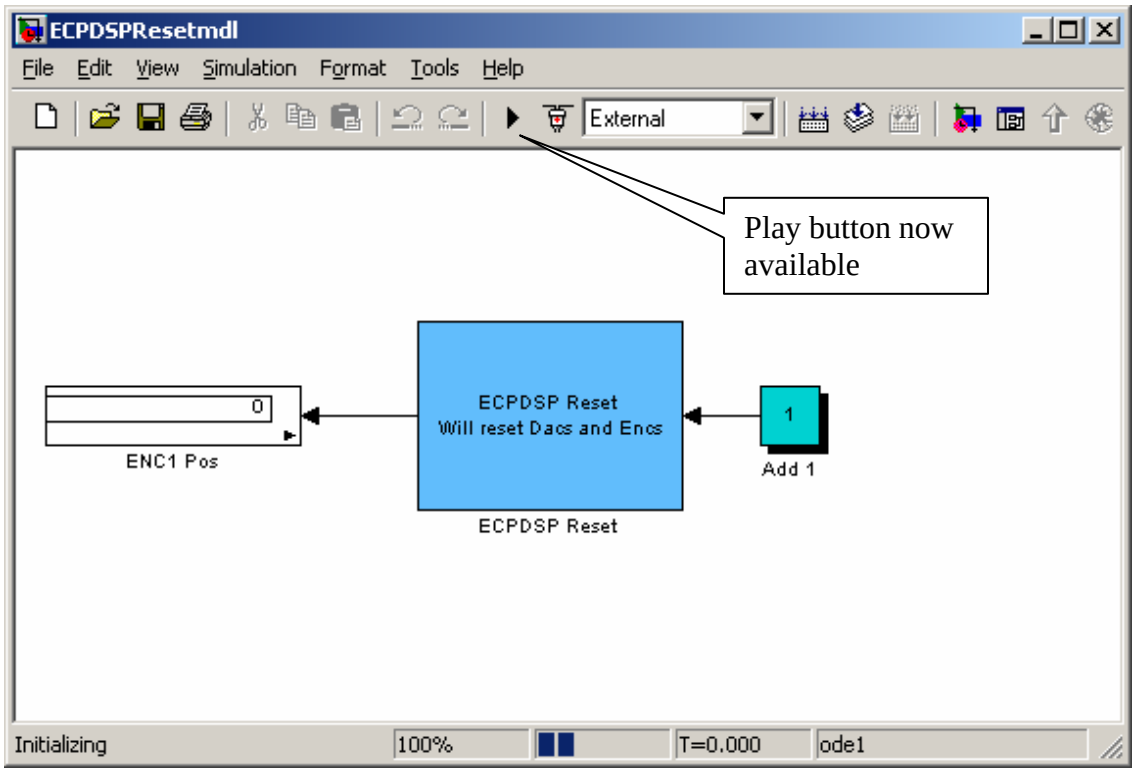
Step 2. Now in order to make sure communications are OK, we need to be sure the address for the signal processing card is where we think it should be. To do this, double click on the ECP Reset block. You should get the following:



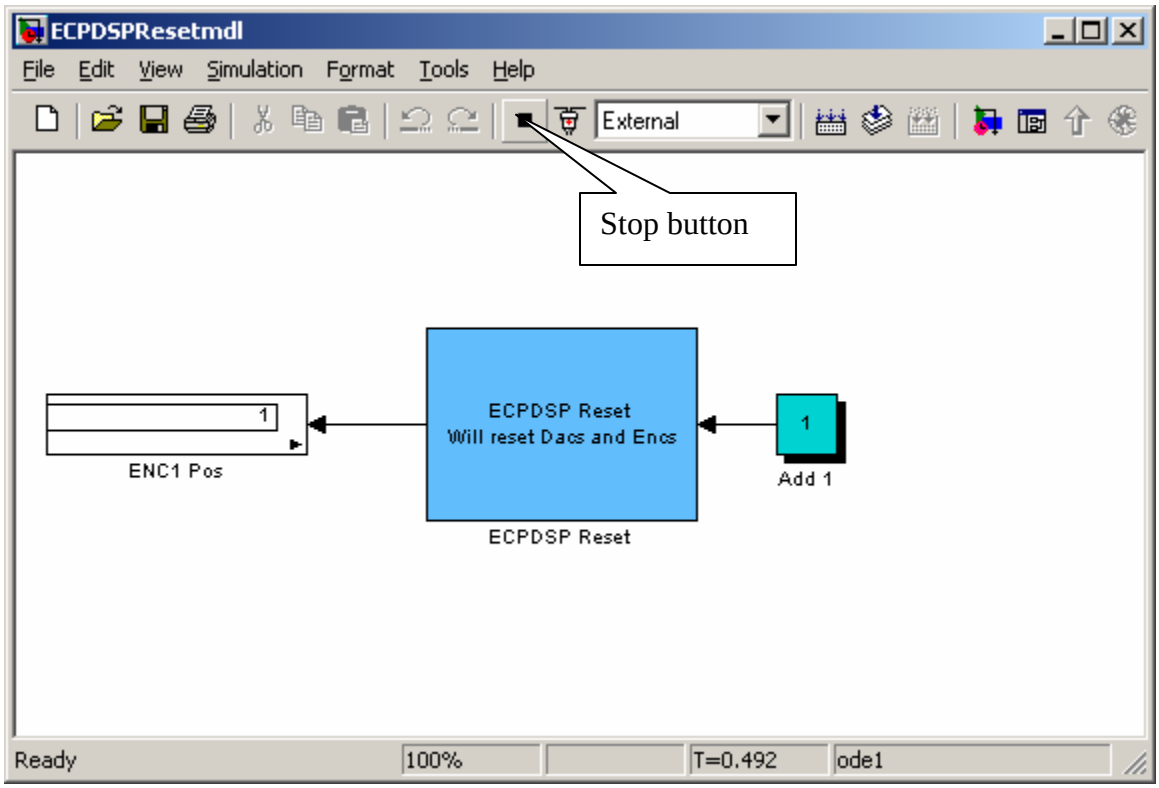
Step 3. Now we need to connect this model to the real system. To do this, first make sure the word External is showing in the box, and then click on the icon next to the box, as shown below.



The play button should now be available, as shown below. In the Matlab window, you should get the message *Model ECPDSPResetmdl loaded.*



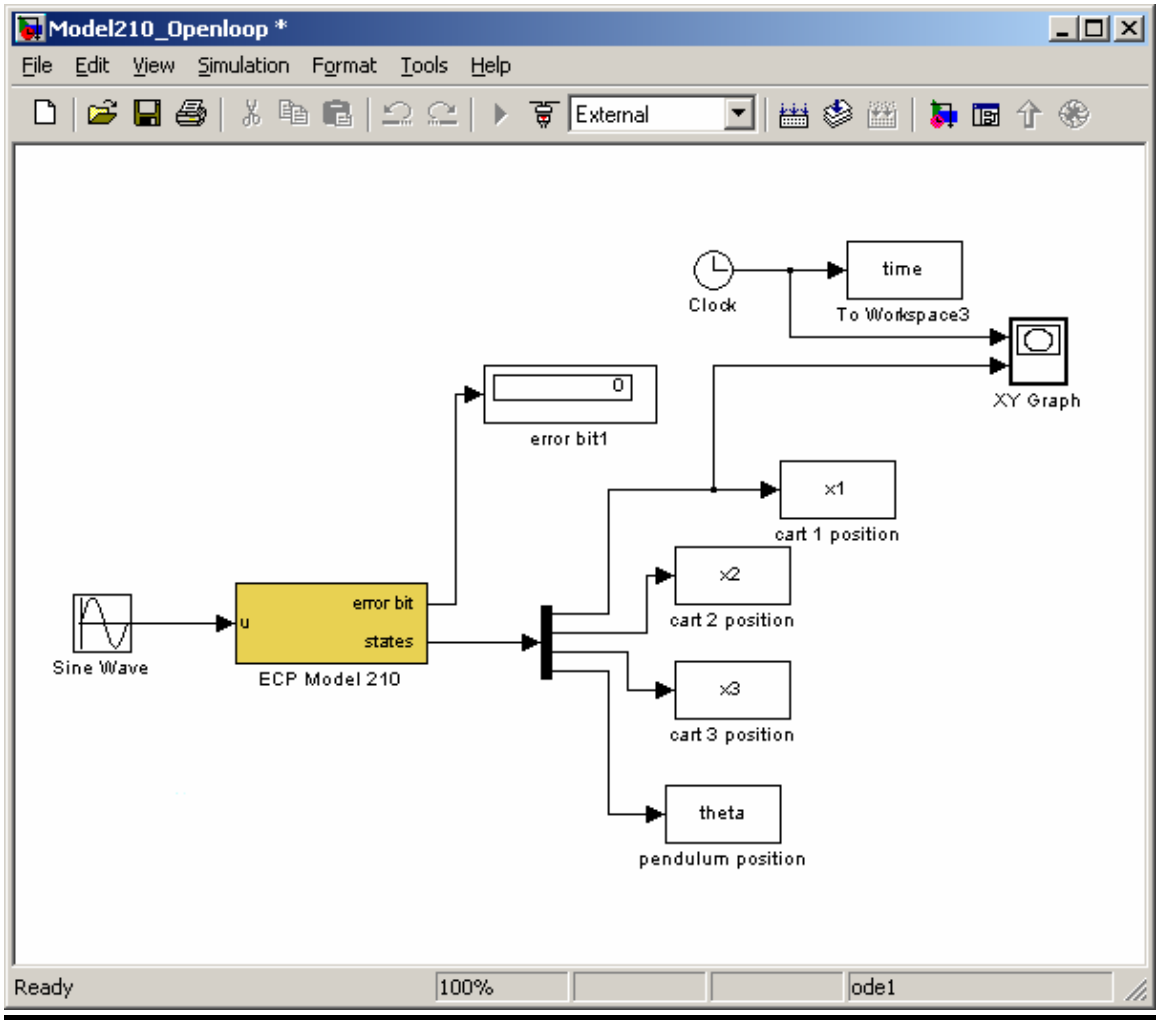
Step 4. Now click on the play button. The window should look briefly like the following. The black box (where the play button was) is the stop button, you can use this to stop the simulation at any time.



If you've done everything correctly, after the system has run in the Matlab window you should get the message *Model ECPDSPResetmdl unloaded.* This routine is just used to initialize and reset values, the ECP system will not move.

Model210_Openloop.mdl

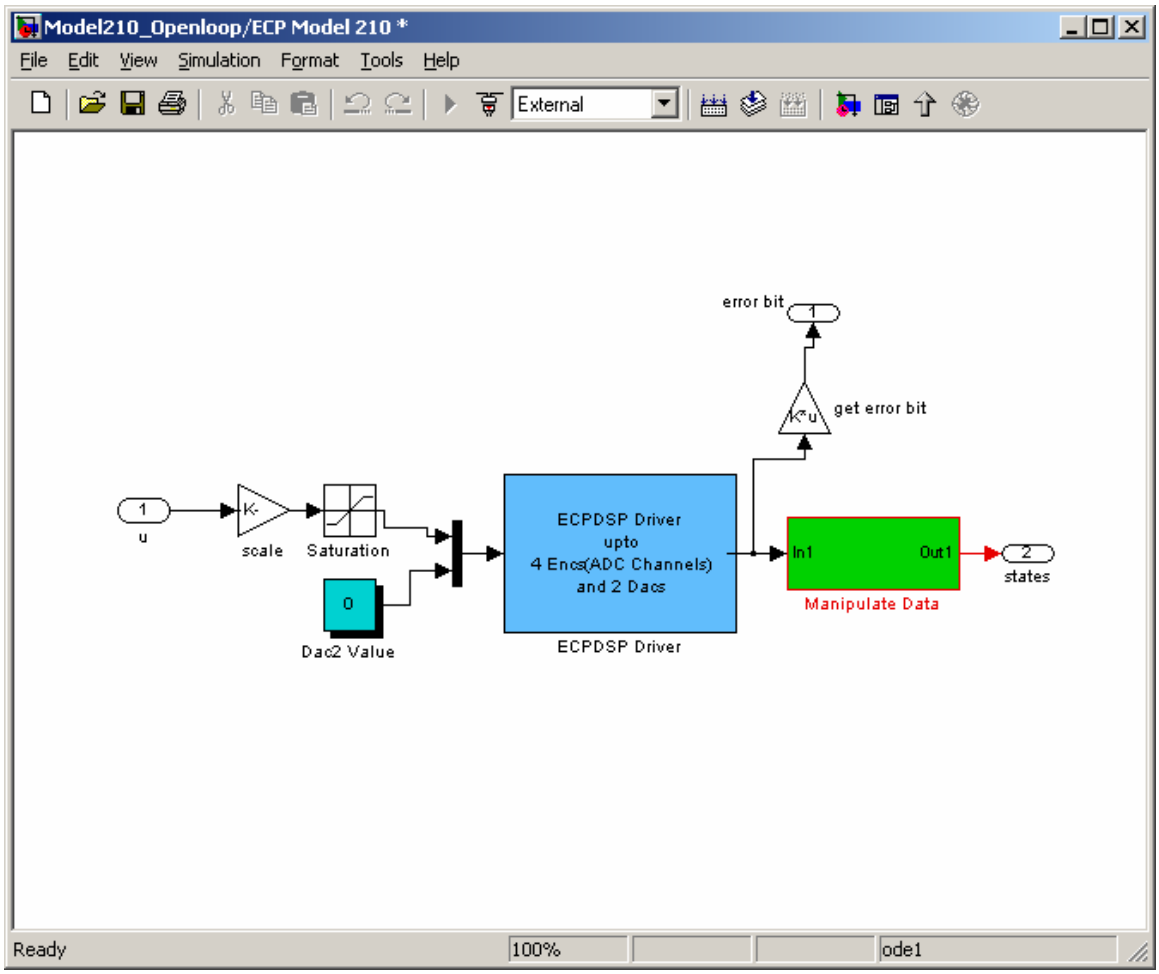
The first thing we usually need to do is to identify the system by constructing a Bode plot and fitting the measured frequency response to the expected frequency response of the transfer function we expect. To do this we will usually use the Simulink file **Model210_openloop.mdl**. This is shown below:



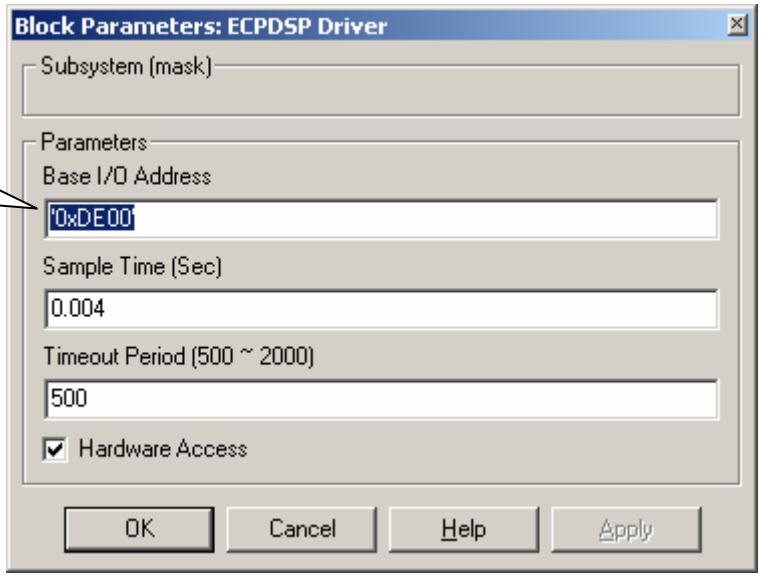
This particular implementation has the input signal in cm and the position of the three carts measured in cm. The output of the pendulum attachment is shown in radians. For this implementation, the position of the first mass is plotted as a function of time as the ECP system runs (the default is to plot for the first 20 seconds).

The first thing we need to do is be sure the Simulink is communicating with the signal processing board correctly. This means we need to be sure the Simulink model thinks the

signal processing board is at the correct address. To do this, first click on the **ECP Model 210** block. You should get the following:

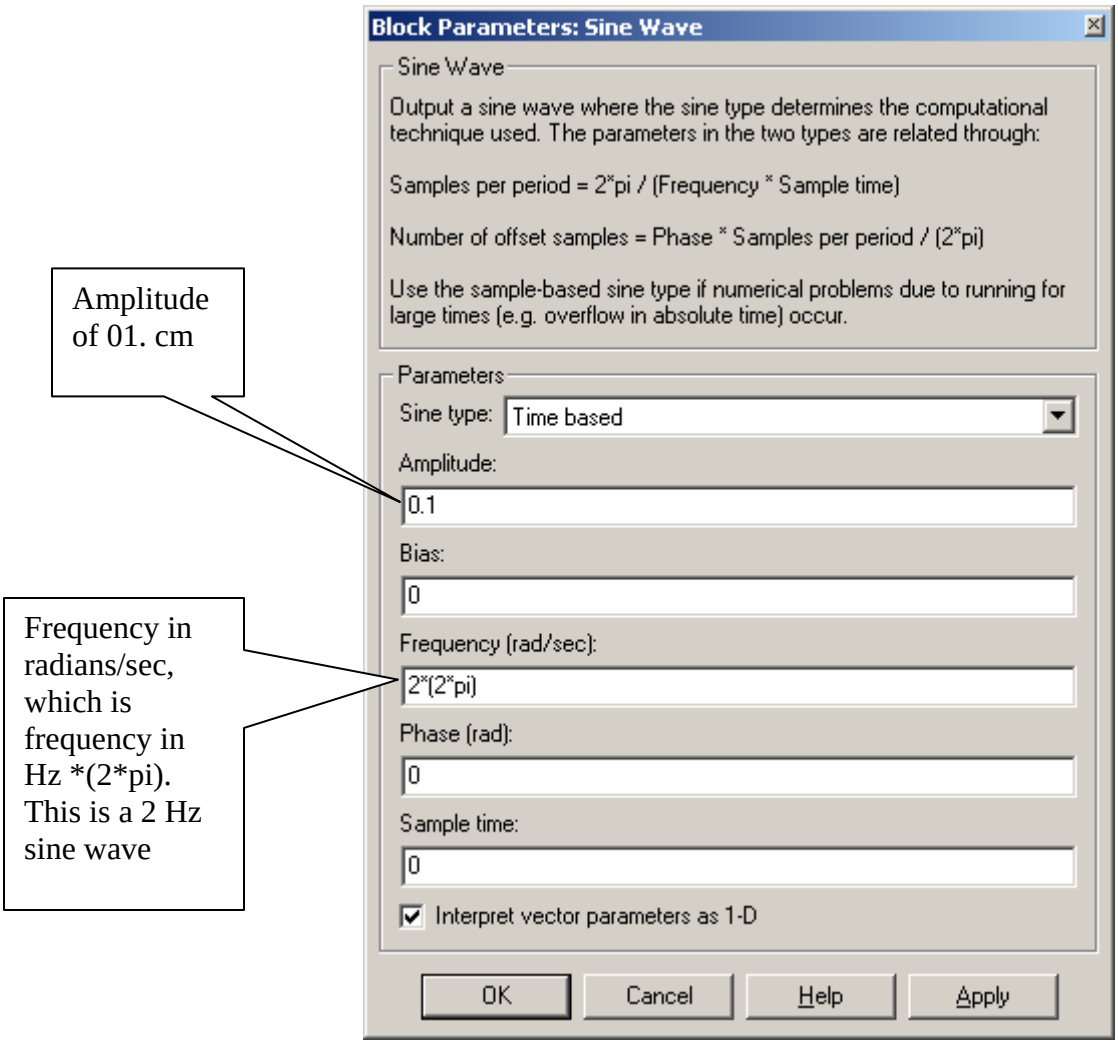


This basically shows us that the **ECP Model 210** block is made up of subsystems, most of which you don't have to worry about. To check the address of the DSP card, clock on the **ESPDSP Driver** block. You should get the following:

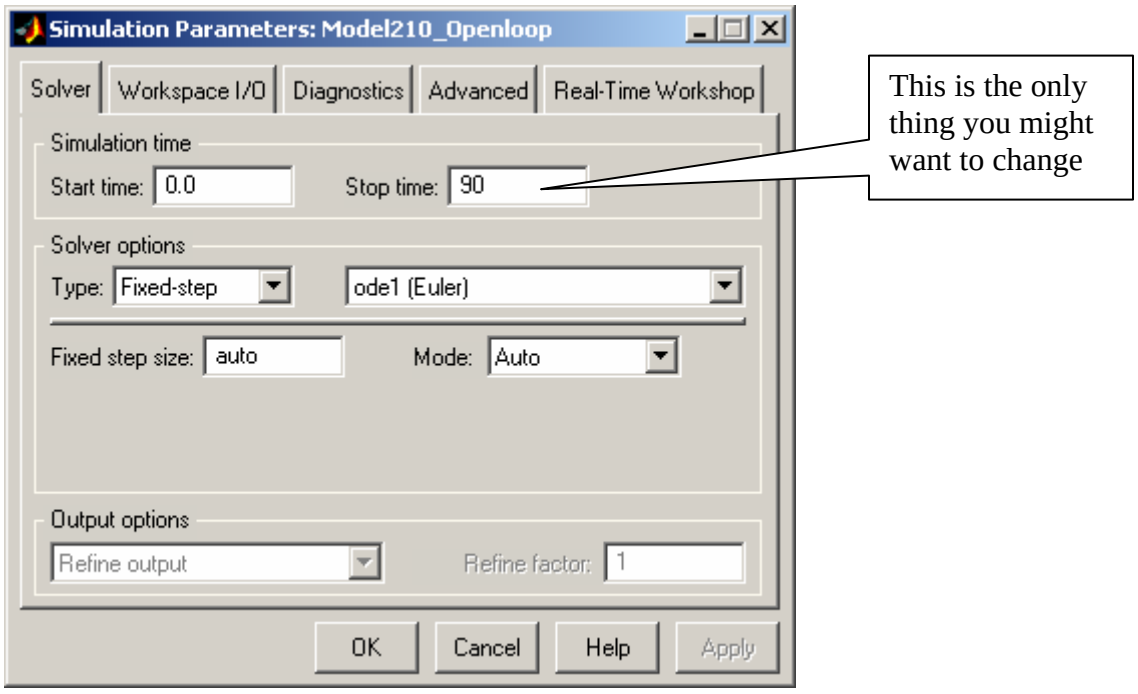


Be sure this address is correct!

In order to plot sine waves of different frequencies, you need to click on the **Sine Wave** block, and both the input frequency and perhaps the input amplitude. The result of clicking on the Sine Wave block is shown below

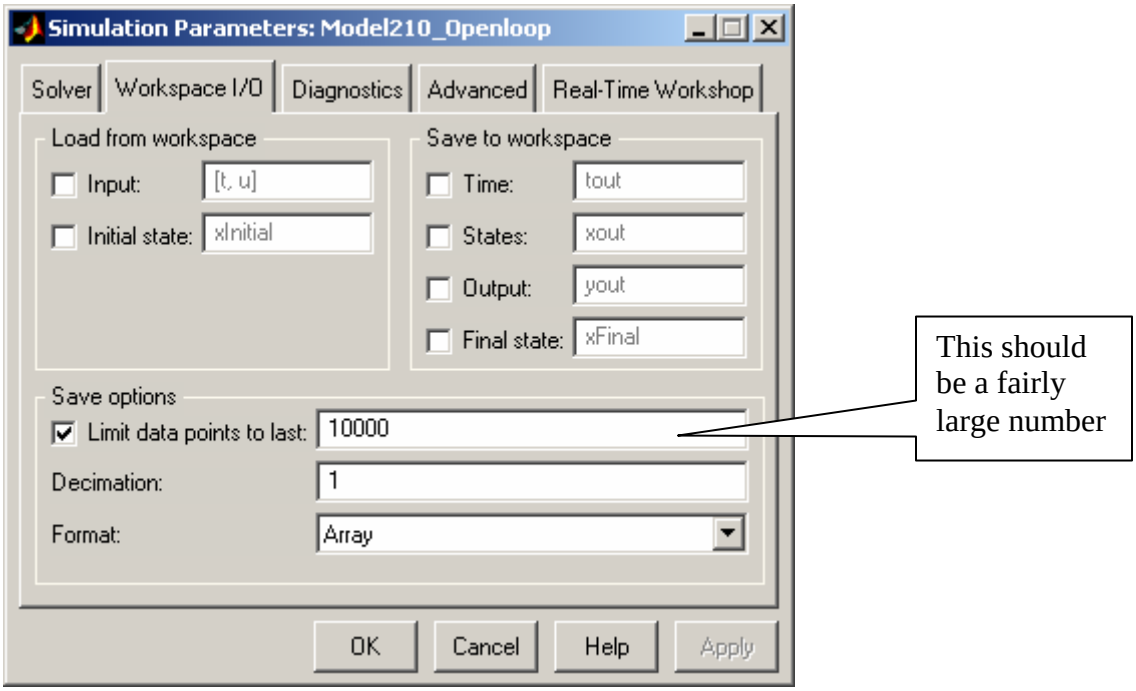


Next we need to set up some simulation parameters. Select **Simulation -> Simulation Parameters.** You will get a screen like the following:

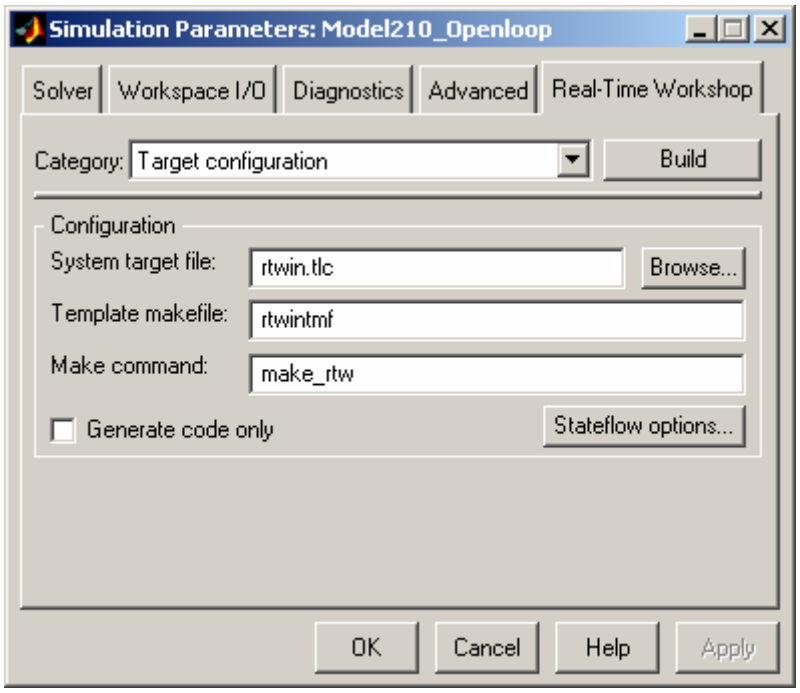


Next, you might want to select Workspace I/O and modify it to look like the following:

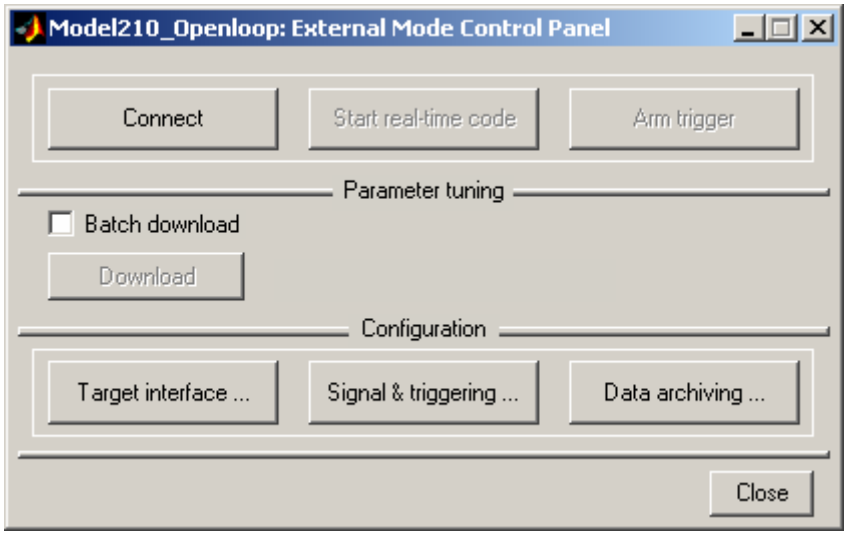
Finaly



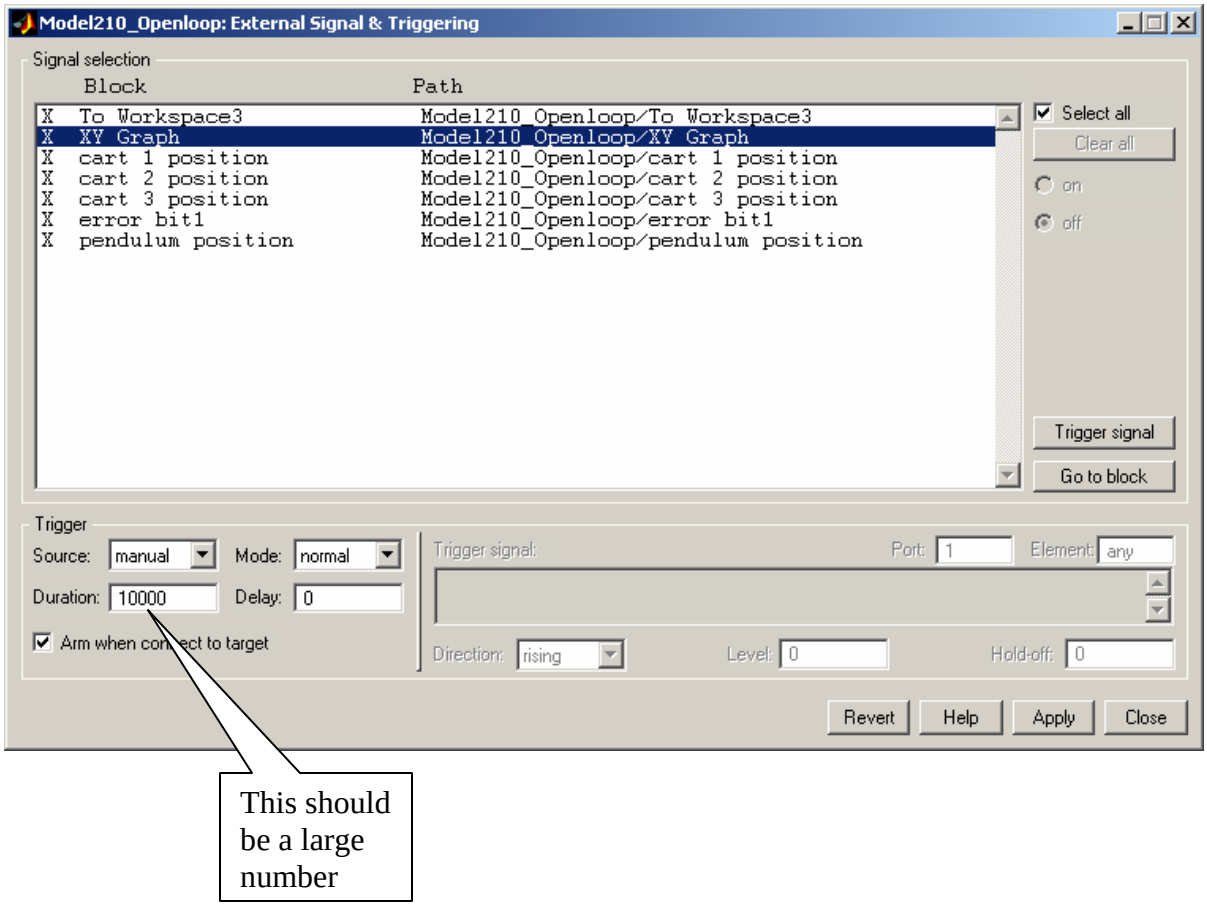
Finally, if you select Real-Time Workshop, it should look like this:



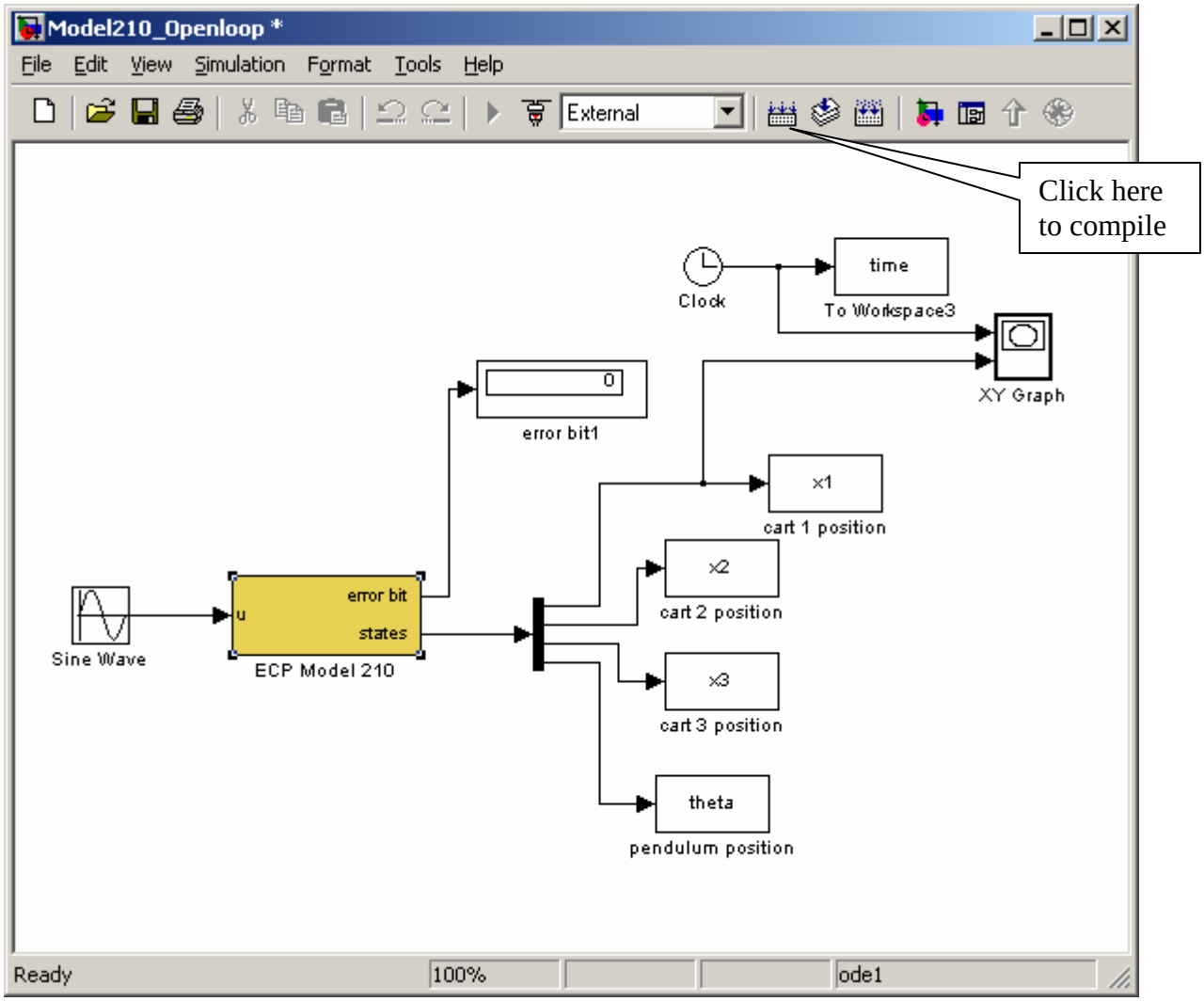
Sometimes, if the system is not saving enough of your data, you will need to select **Tools->External Mode Control Panel** and get the following display



Select **Signal & triggering** and get the following:



At this point we a re ready to compile and run **Model210_Openloop.mdl.**

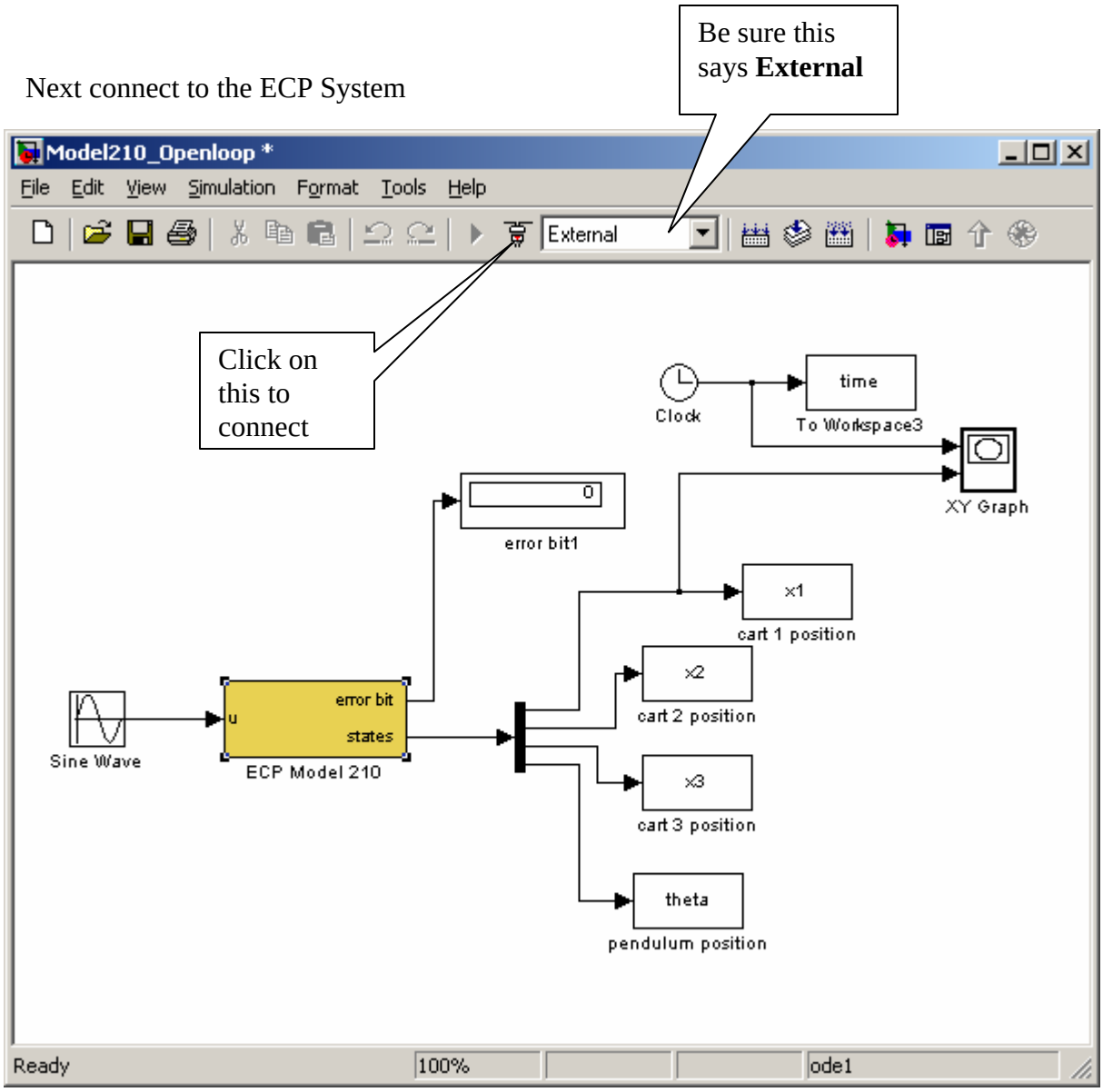


After compiling, you should get the message

Successful completion of Real-Time Workshop build procedure for model: Model210_Openloop

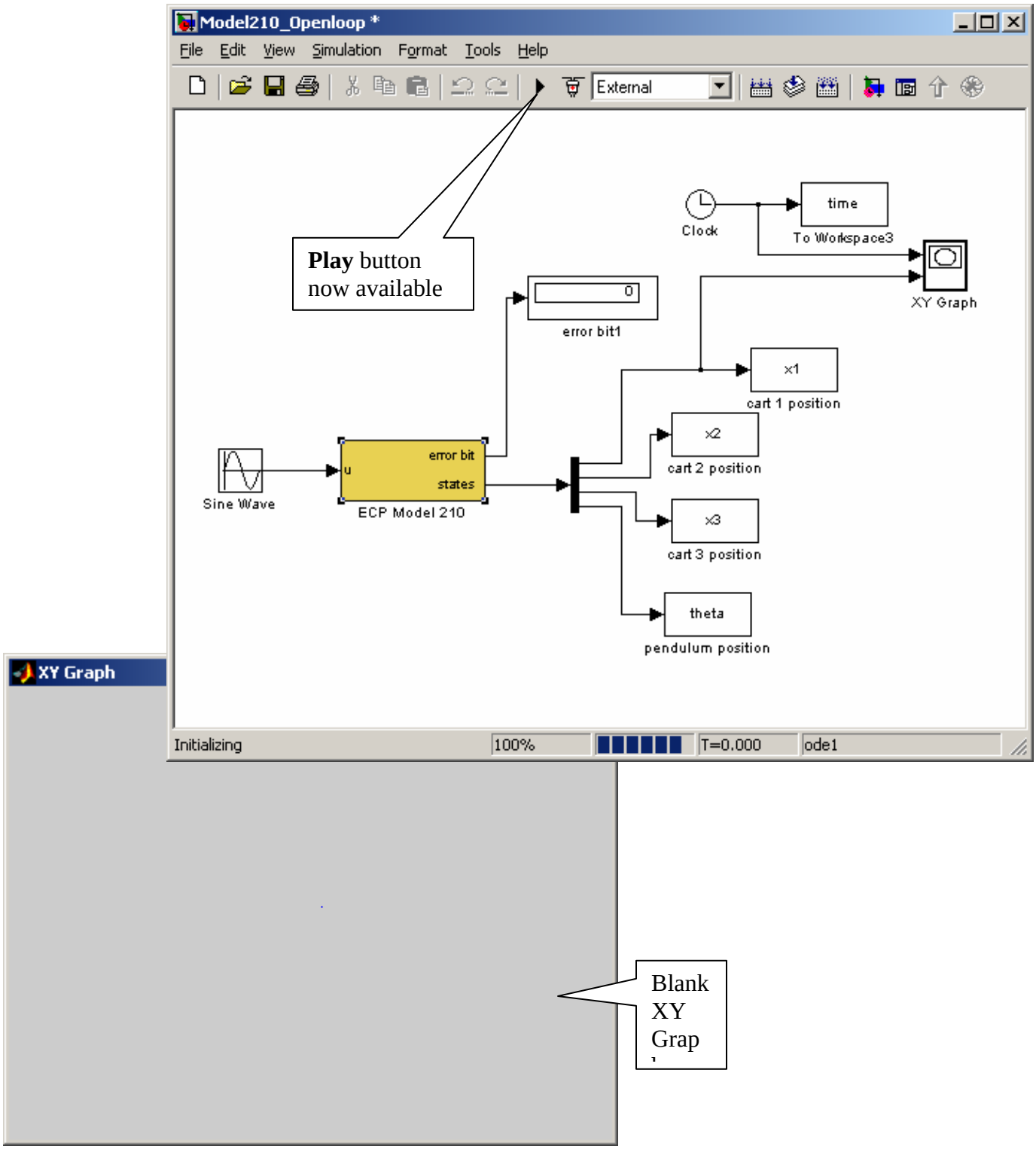
In the Matlab window. Sometimes you get a number of warnings, and if you compile it twice these go away.

Next connect to the ECP System

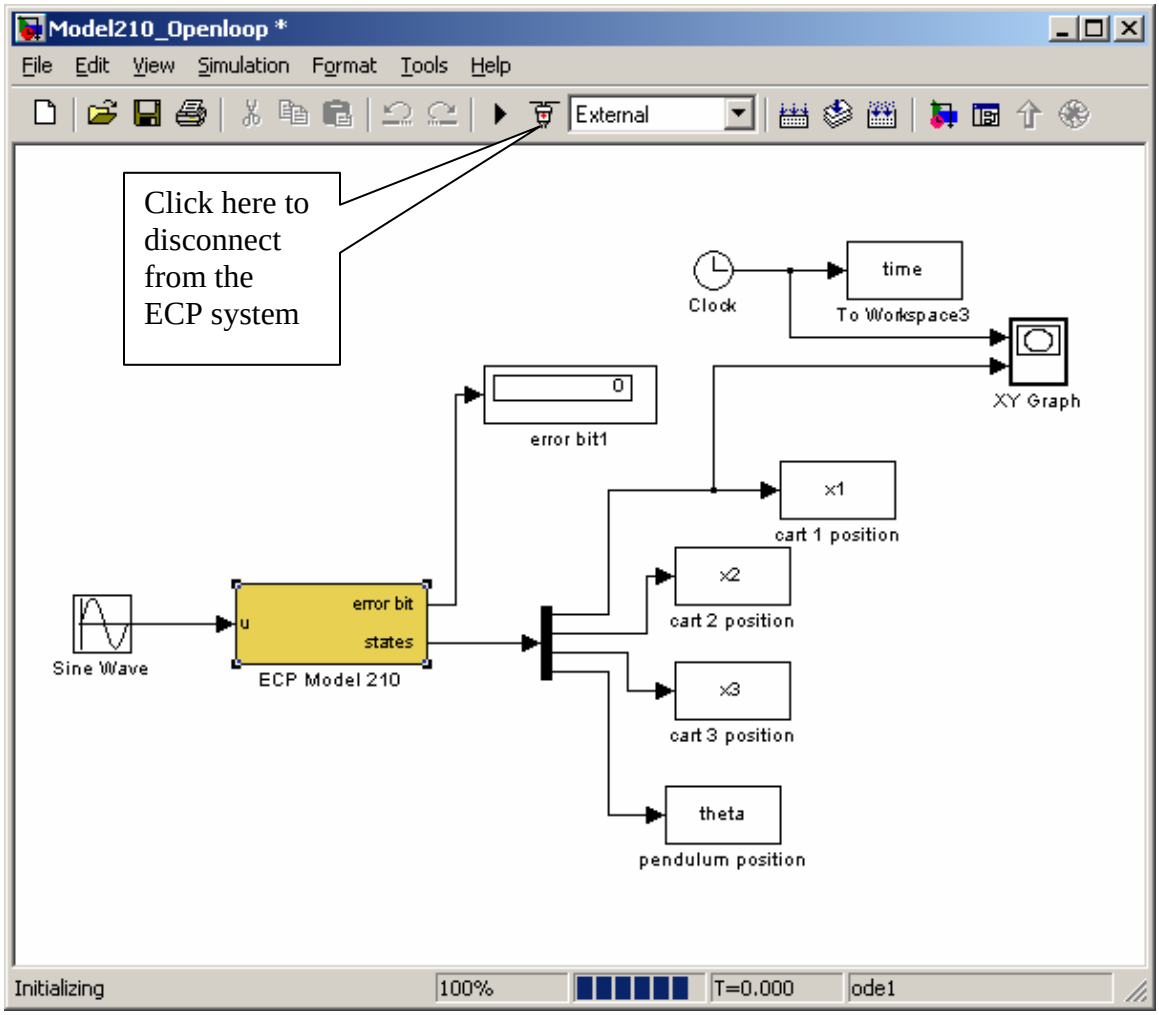


Sometimes when you try and connect you get an error, even though there is nothing wrong. Just clear the window that comes up and try again.

The play button should now be available, and a blank XY graph should appear .

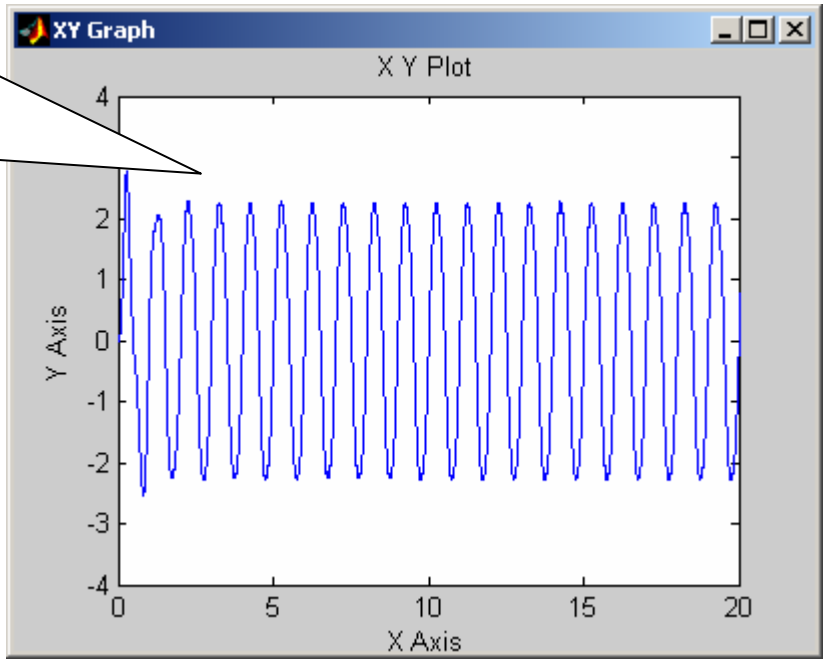


Note that if you want to disconnect from the ECP system, we click on the button next to the **play** button.

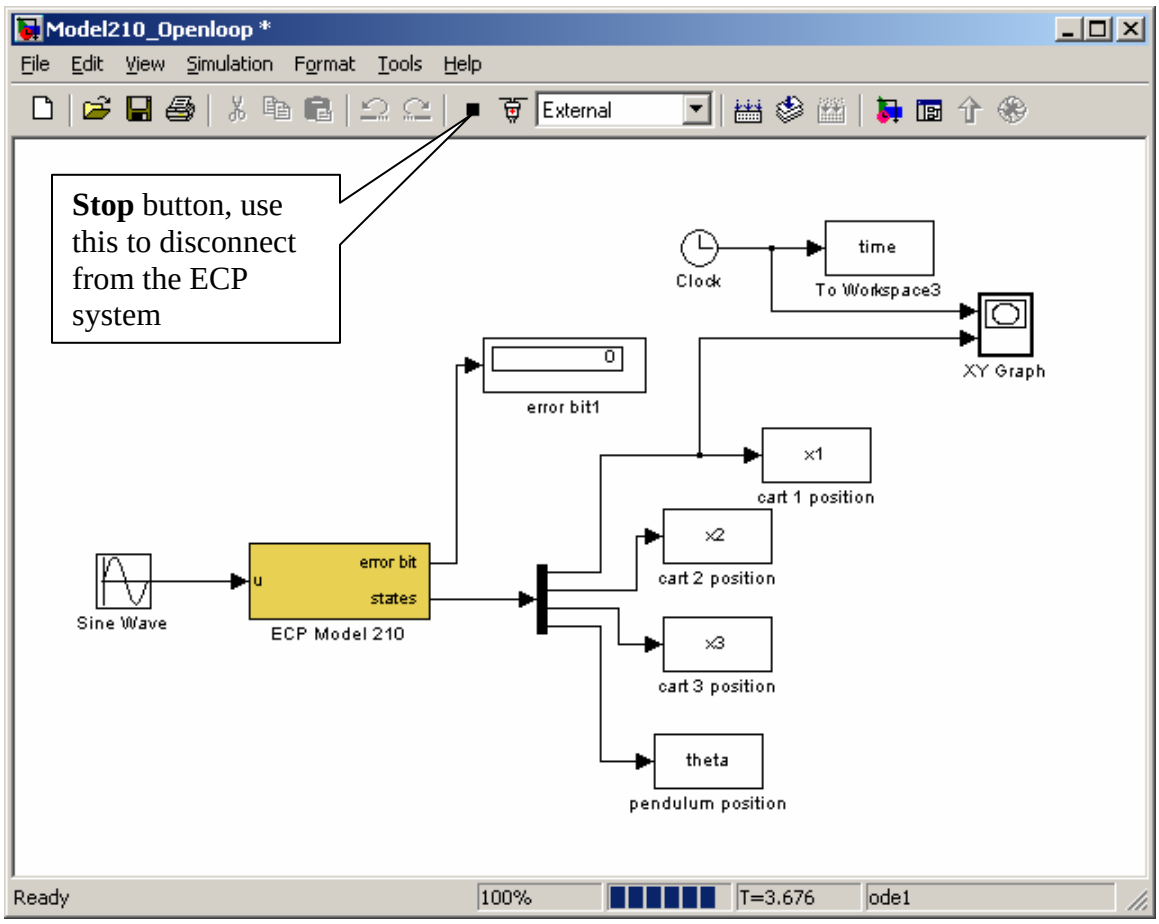


Finally, we can run the system, the XY-graph (for the model I used) will look like

If your cart moves more then 3 cm in either direction, something is wrong with the communications between Simulink and the ECP system. Ask for help!

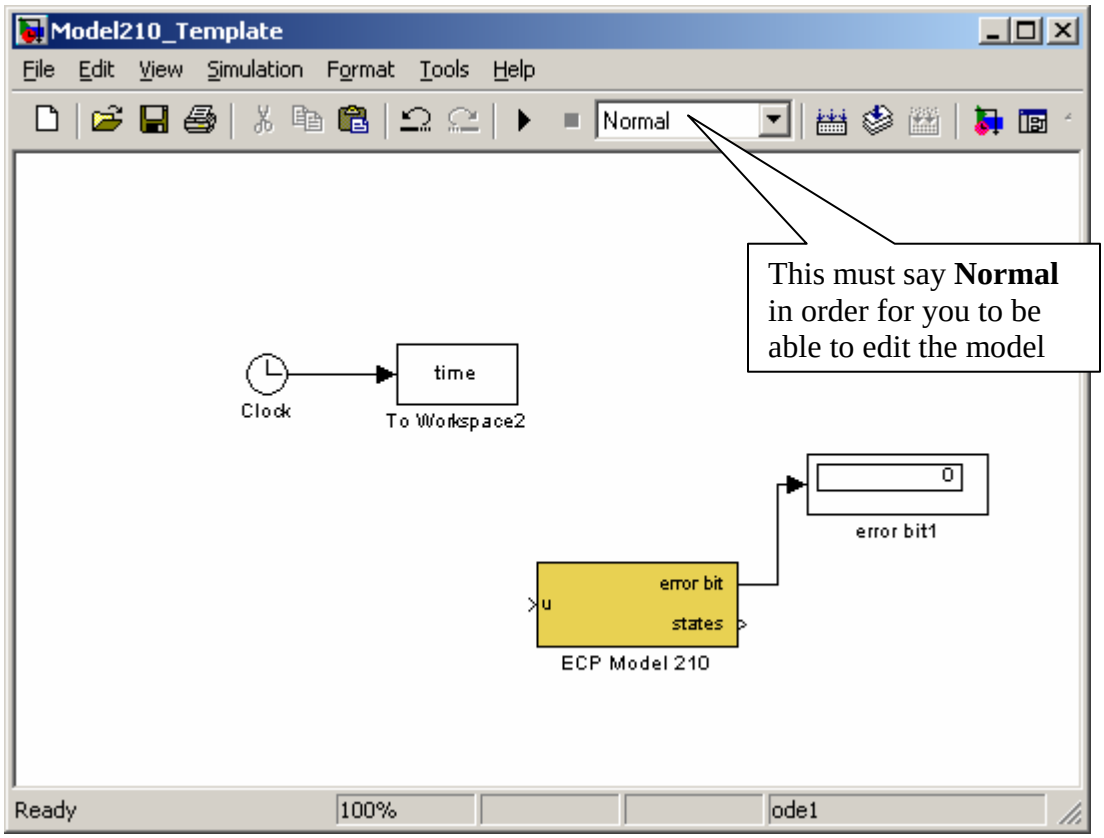


Once we have started running the ECP system, we can disconnect by clicking on the **stop** button. Any data that has been collected will be saved in the workspace.

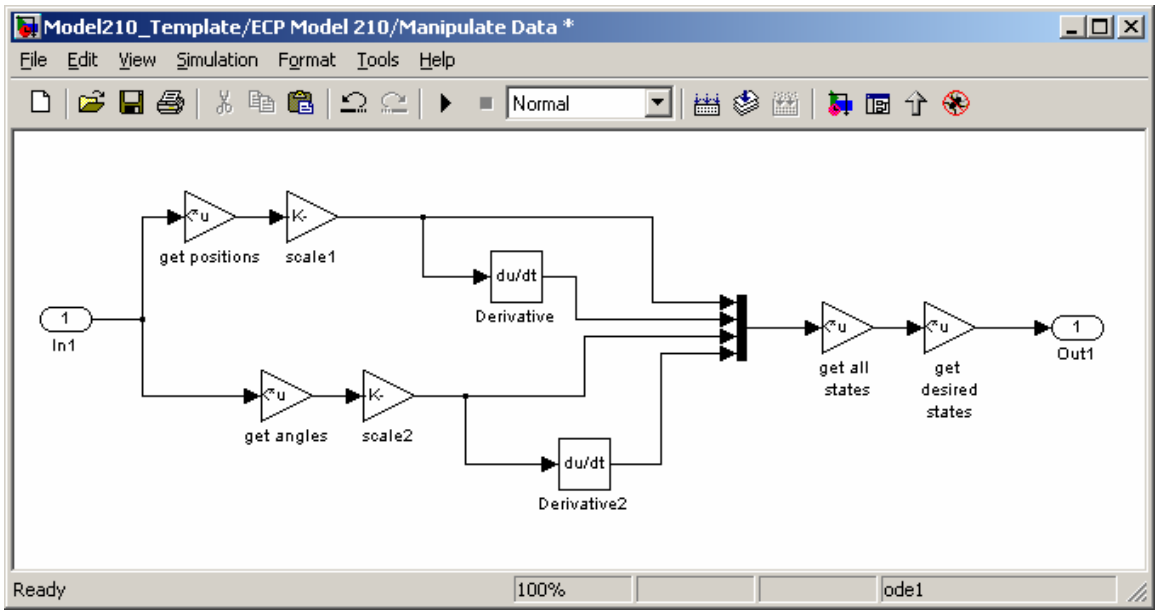


Model210_Template.mdl

For most of our systems, we will utilize the basic skeleton shown below:



Note that this looks like **Model210_Openloop.mdl**, but it has different subsystems in the **ECP Model 210** block. If we click on this block we can see what's inside. It initially looks like the inside of **Model210_Openloop**, but the contents of the **Manipulate Data** block are different.



To understand this subsystem, we'll start at the **In 1** on the left. The input (the signal the arrives at **In 1**) is the output of the ECP Driver, which is a 5x1 vector which consists of

1. the position of the first cart,
2. the position of the second cart,
3. the position of the third cart,
4. the position of the pendulum, and
5. an error bit.

The error bit is dealt with separately, so let's worry about the first four entities, all of which are positions. Now the ECP measures positions in **counts**, and we have to convert **counts** to either **cm** (for the cart positions) or **radians** (for the pendulum position). There is a different conversion for the carts and the pendulum, so we first separate the all of the cart positions from the pendulum positions. The array **get positions** is a 3x5 array of 1's and 0's to pick off the positions of the first three carts (from the input vector of 5 components). The array **get angles** is a 1x5 vector of 1's and 0's used to pick off the position of the pendulum (from the input vector of 5 components). Hence the top path is for the cart positions, the bottom path for the pendulum positions. Next both the cart positions and the pendulum positions must be scaled from **counts** to **cm** . Now in addition to the positions, we also want the first derivatives, so we do that next. What comes out of the mux (the black bar) is an 8x1 vector, which contains

1. the position of the first cart,
2. the position of the second cart,
3. the position of the third cart,
4. the velocity of the first cart,
5. the velocity of the second cart,
6. the velocity of the third cart,
7. the position of the pendulum, and
8. the velocity of the pendulum.

Now this is not the usual way we want the states numbered, so the matrix **get all states** is a matrix of 1's and 0's that sorts the states so the state vector is what we, want:

1. the position of the first cart,
2. the velocity of the first cart,
3. the position of the second cart,
4. the velocity of the second cart,
5. the position of the third cart,
6. the velocity of the third cart,
7. the position of the pendulum, and
8. the velocity of the pendulum

Finally, we don't need all of the states for every model, so the matrix **get desired states** is a matrix of 1's and 0's so we only output the states we are going to be using. This is the only matrix you will have to modify (it should be modified in the Matlab driver file**).**