

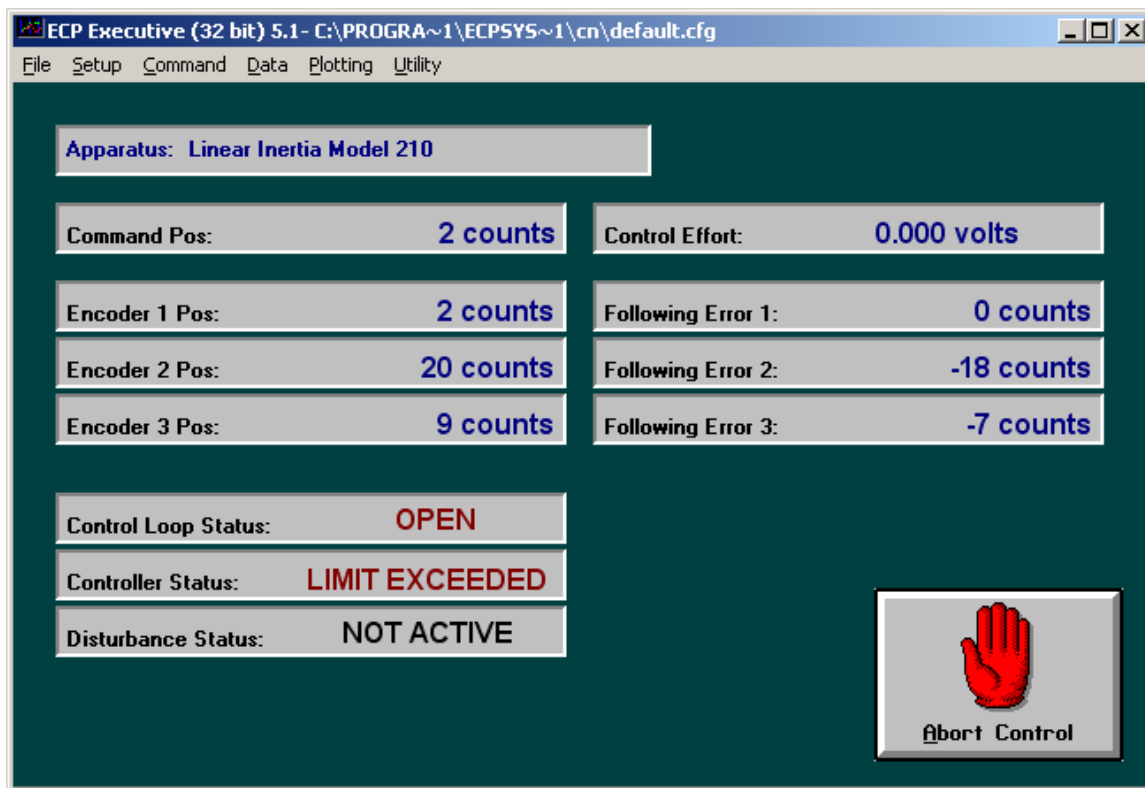
Real-Time Windows Target, Simulink, and the ECP Model 210 and Model 205

Every station in C-116 should be set up to run the ECP Model 210 (rectilinear system) and/or Model 205 (torsional system) 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.

Do not interchange the ECP Model 210 and ECP Model 205 Simulink blocks. They have scaling differences and are not interchangeable!

Step 1. 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 **Utility-> Download Controller Personality File.**

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

Select **m210_rtw_3.pmc** (to use the Model 210) or **m205_rtw_3.pmc** (to use Model 205) and click on **open**. Wait for the ECP system to load the personality file.

Step 2. Select **Utility -> Reset Controller**.

Step 3. Close the ECP Executive (**click on the X**) **Do not just minimize it!!!!**

Step 4. 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 5. 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 6. The real-time windows target utilizes Microsoft's Visual C++ compiler. We 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++ Compiler

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.

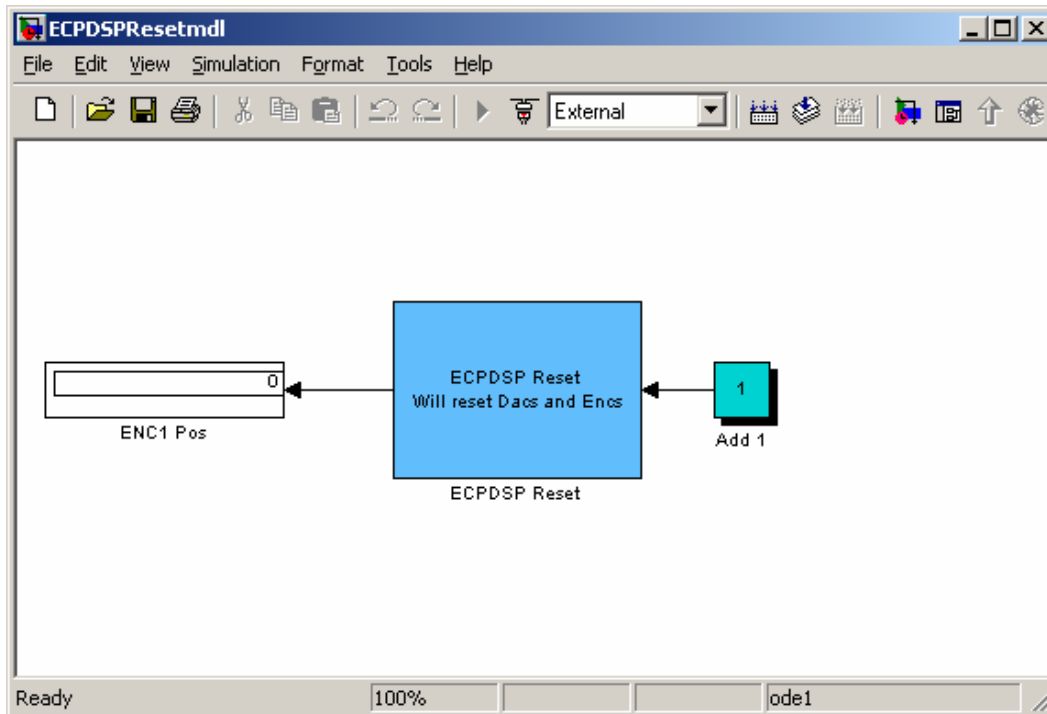
Step 7. Set the Matlab **current directory** to the folder where all of your files are located.

Step 8. Start Simulink by typing **Simulink** in the Matlab window.

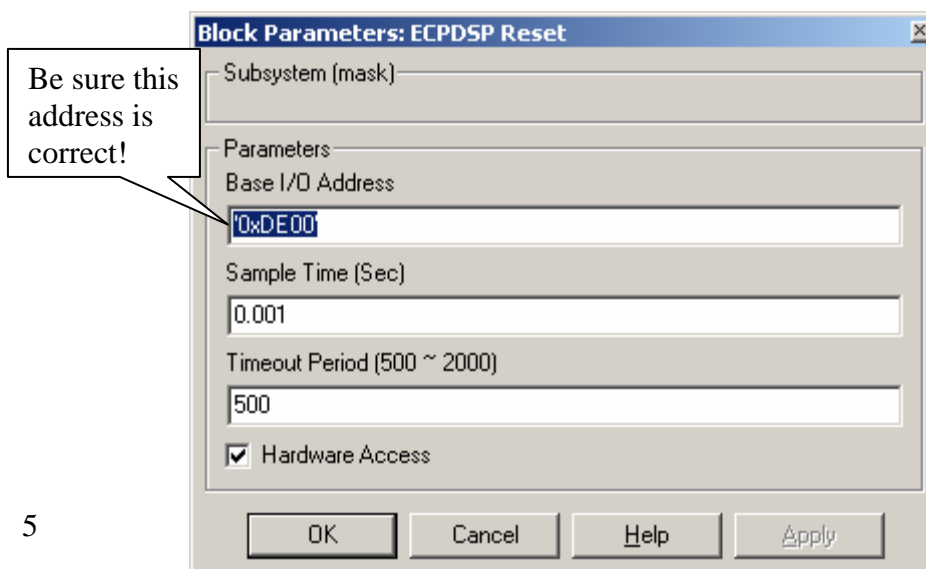
Resetting the System Using Simulink

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.

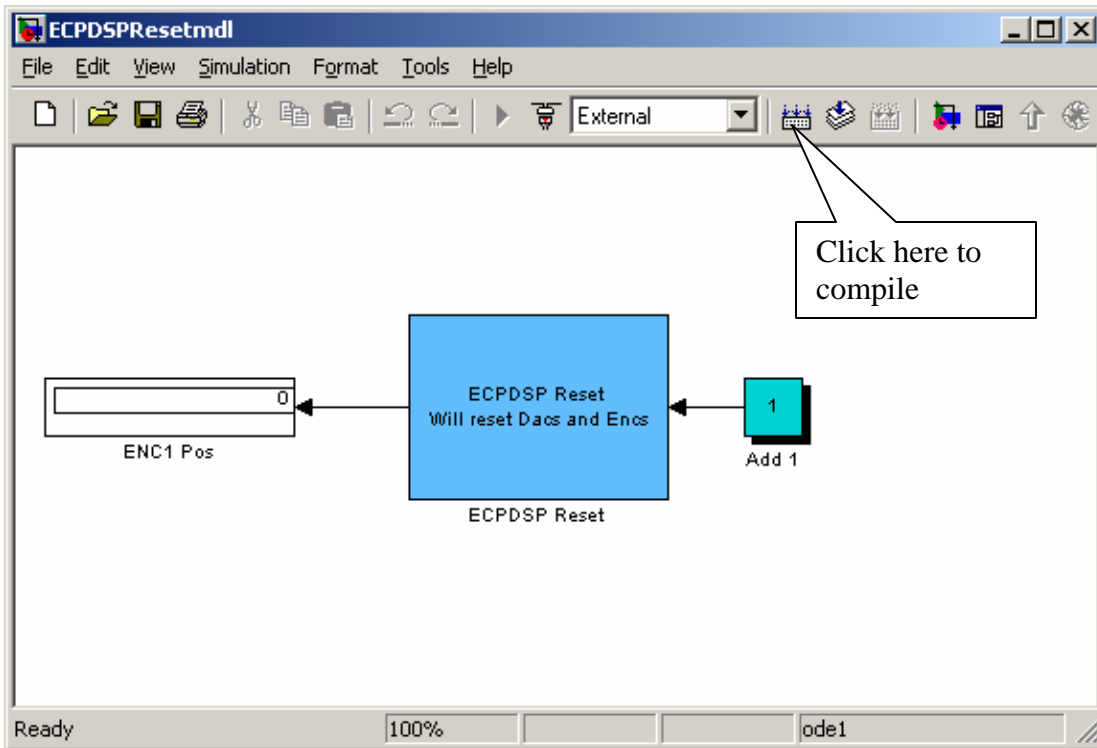
From Simulink select **File->Open->ECPDSPResetmdl.mdl** The following window should pop up.



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:



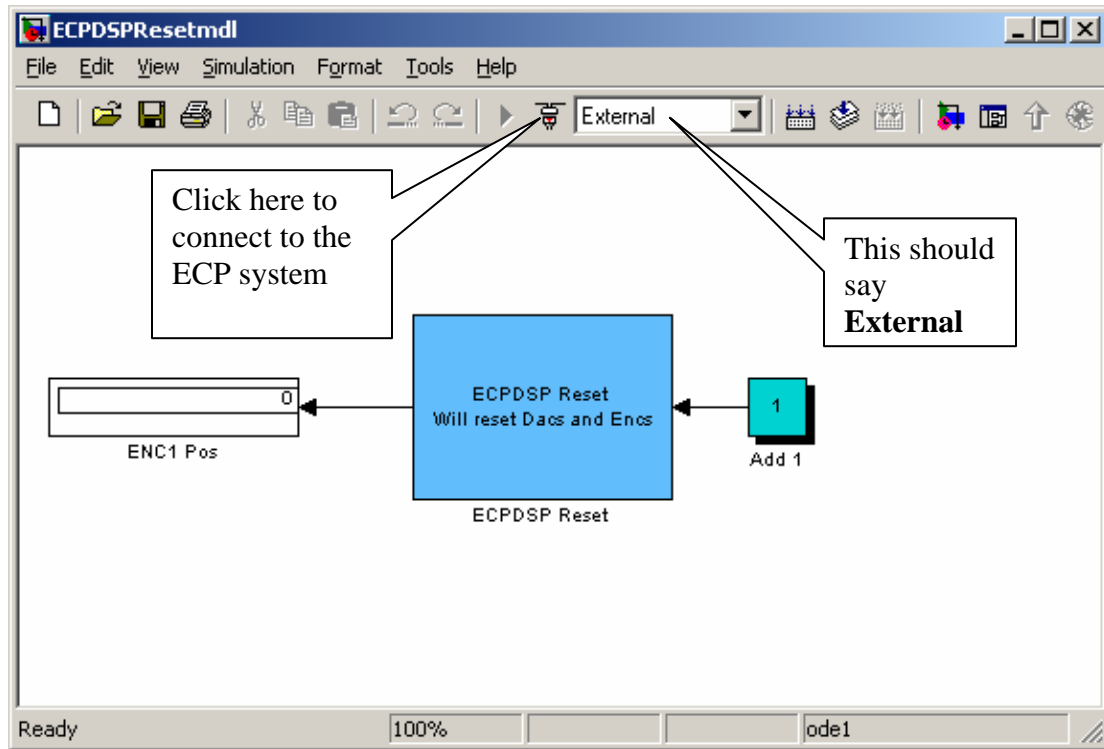
Next we need to compile **ECPDSPResetmdl.mdl**. To do this click on the *compile* icon



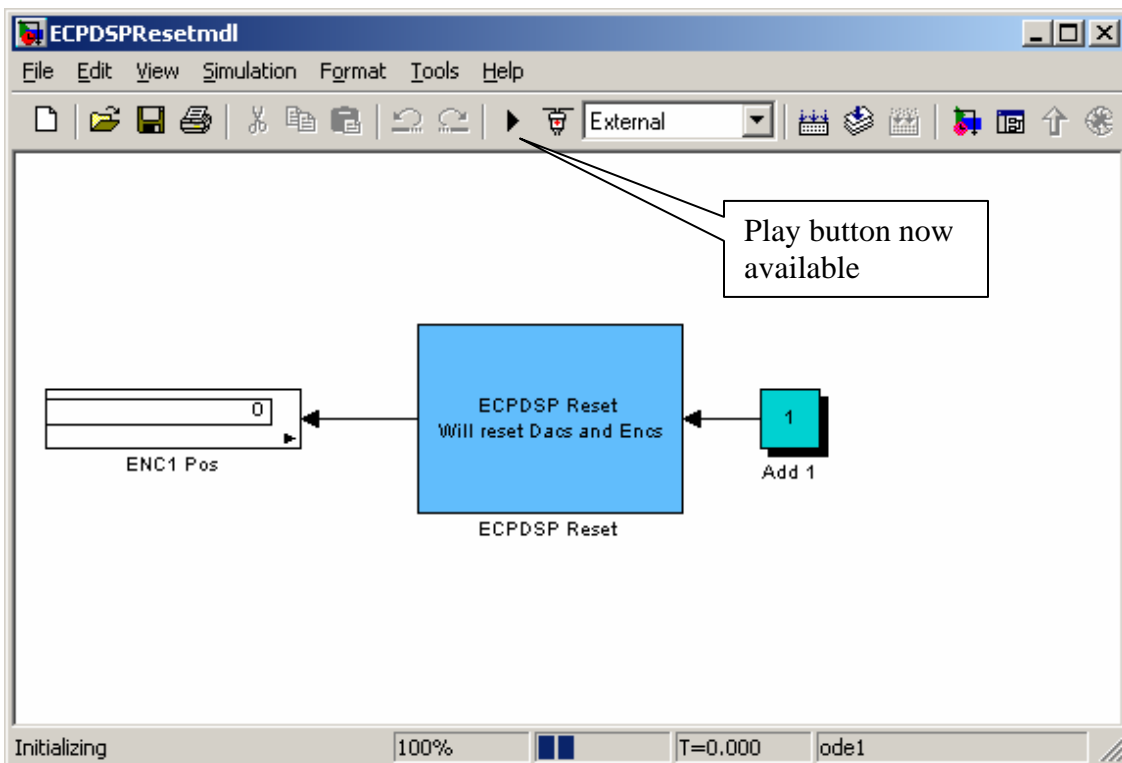
A bunch of stuff will get written to the Matlab window, ending with the line:

```
### Successful completion of Real-Time Workshop build procedure for model:  
ECPDSPResetmdl  
>>
```

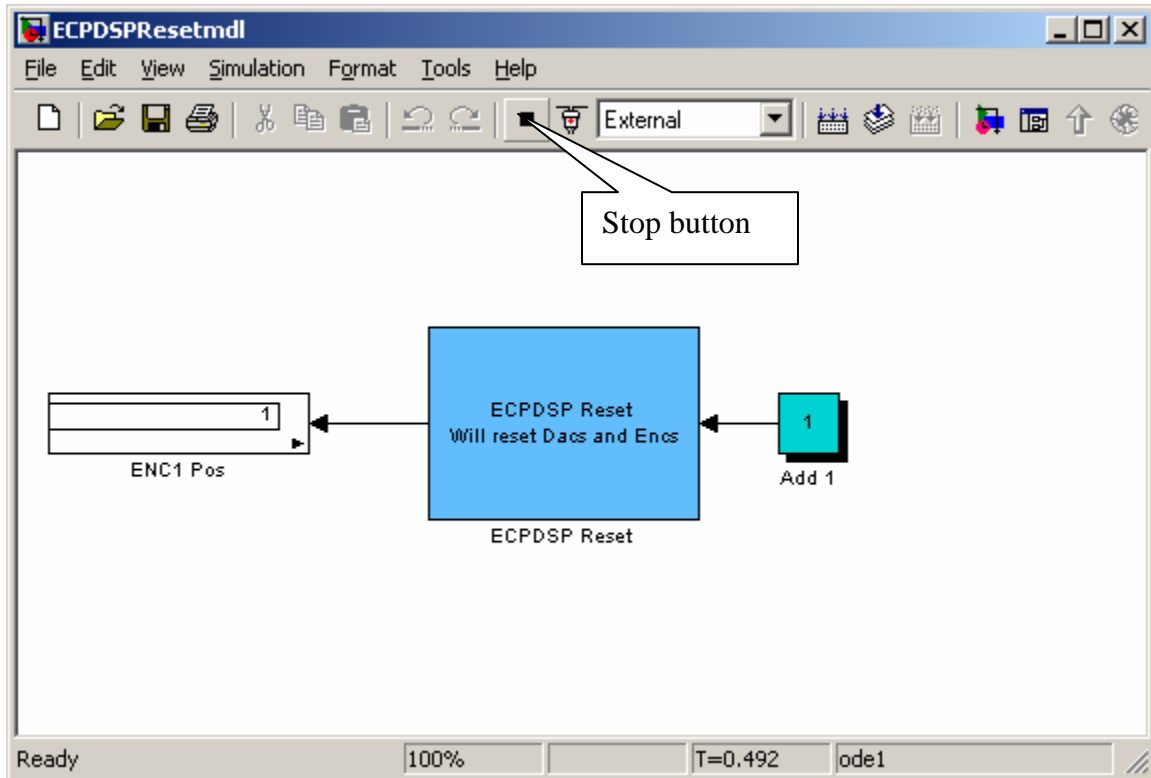
Now we need to connect this model to the real system. To do this, first make sure the word **External** is showing in the box, and then click on the icon next to the box, as shown below.



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



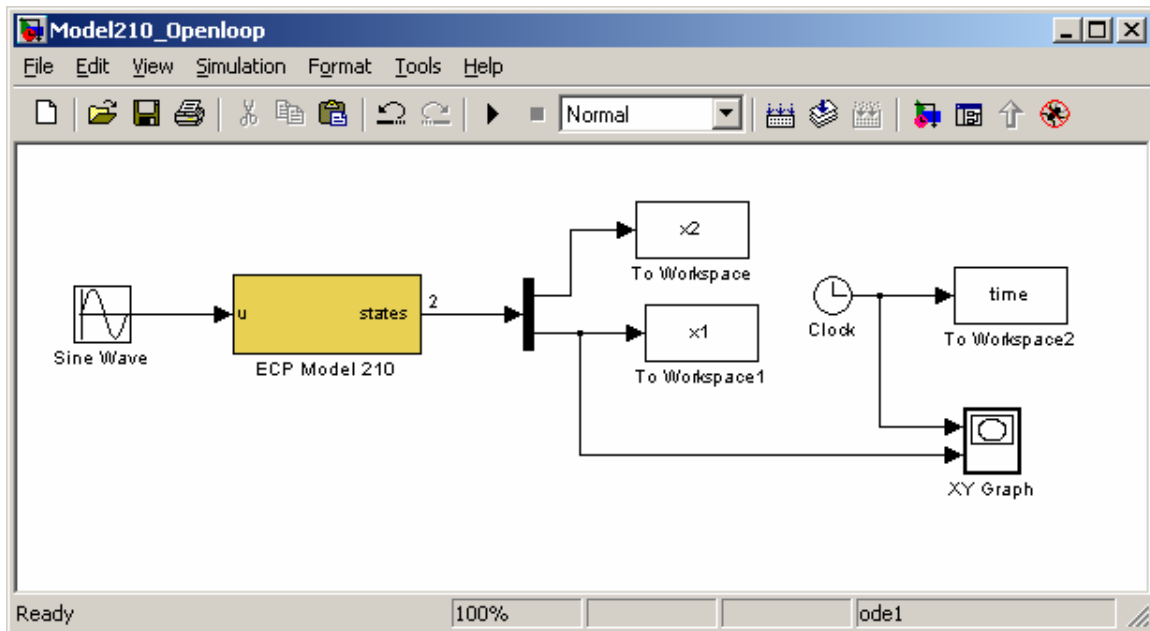
Now click on the play button. The window should look briefly like the following. The black box (where the play button was) is the stop button, you can use this to stop the simulation at any time.



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.

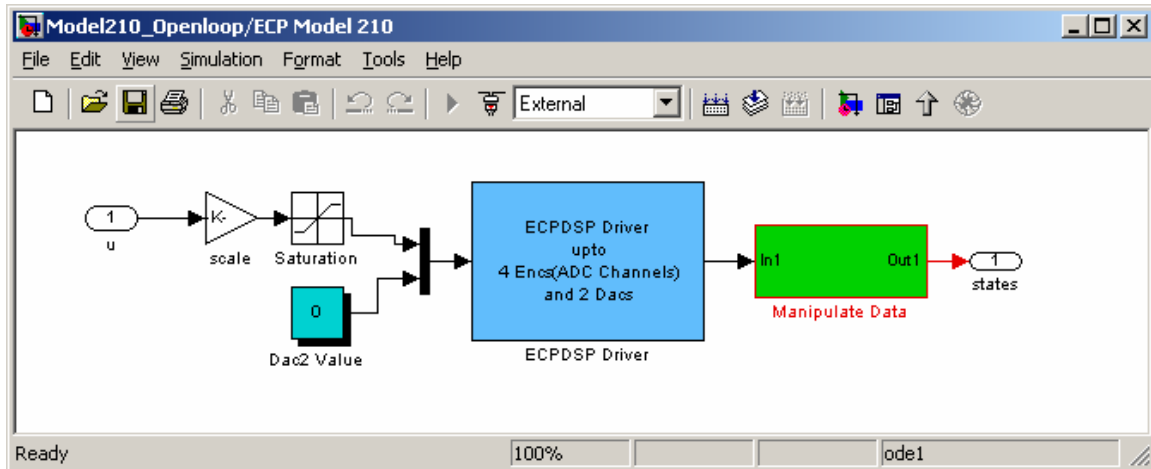
Model210_Openloop.mdl

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 in our model. To do this we will usually use the Simulink file **Model210_Openloop.mdl**. This is shown below:

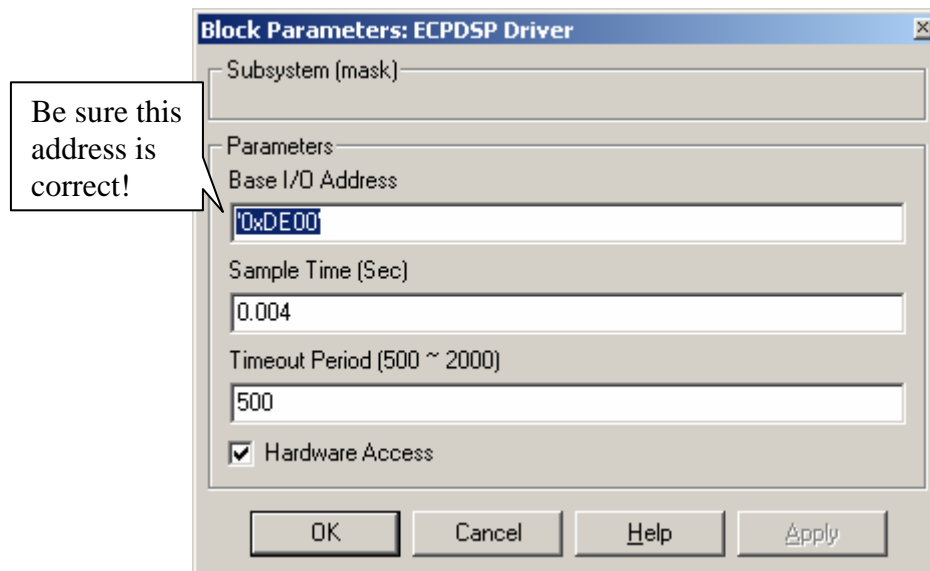


This particular implementation has the input signal in cm and the position of the two carts measured in cm. For this implementation, the position of the first cart (x_1) is plotted as a function of time as the ECP system runs. (Note that the length of time the XY Graph plots the position of the first cart is hard coded, i.e. it cannot be a variable.)

The first thing we need to do is be sure 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, click on the **ESPDSP Driver** block. You should get the following:



Once the Base I/O address, Sample Time, and Timeout Period are OK, click on OK and go back to **Model210_Openloop.mdl**.

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

Block Parameters: Sine Wave

Sine Wave

Output a sine wave where the sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period = $2\pi / (\text{Frequency} * \text{Sample time})$

Number of offset samples = $\text{Phase} * \text{Samples per period} / (2\pi)$

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters:

Sine type: Time based

Amplitude: 0.1

Bias: 0

Frequency (rad/sec): $2*(2\pi)$

Phase (rad): 0

Sample time: 0

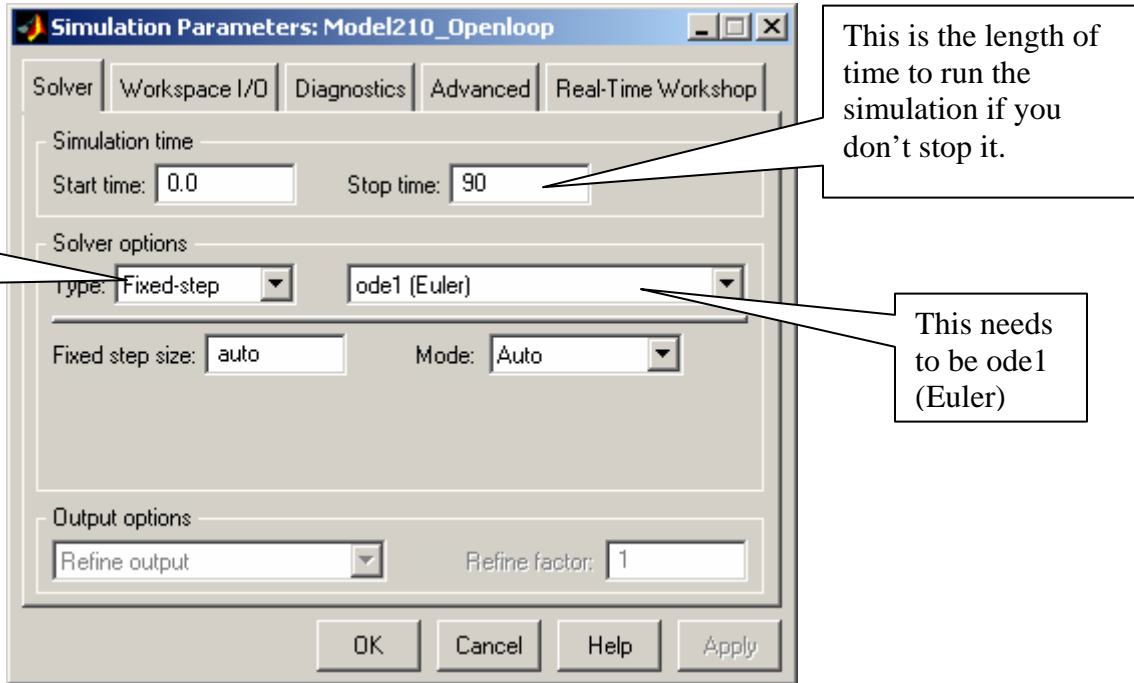
Interpret vector parameters as 1-D

OK Cancel Help Apply

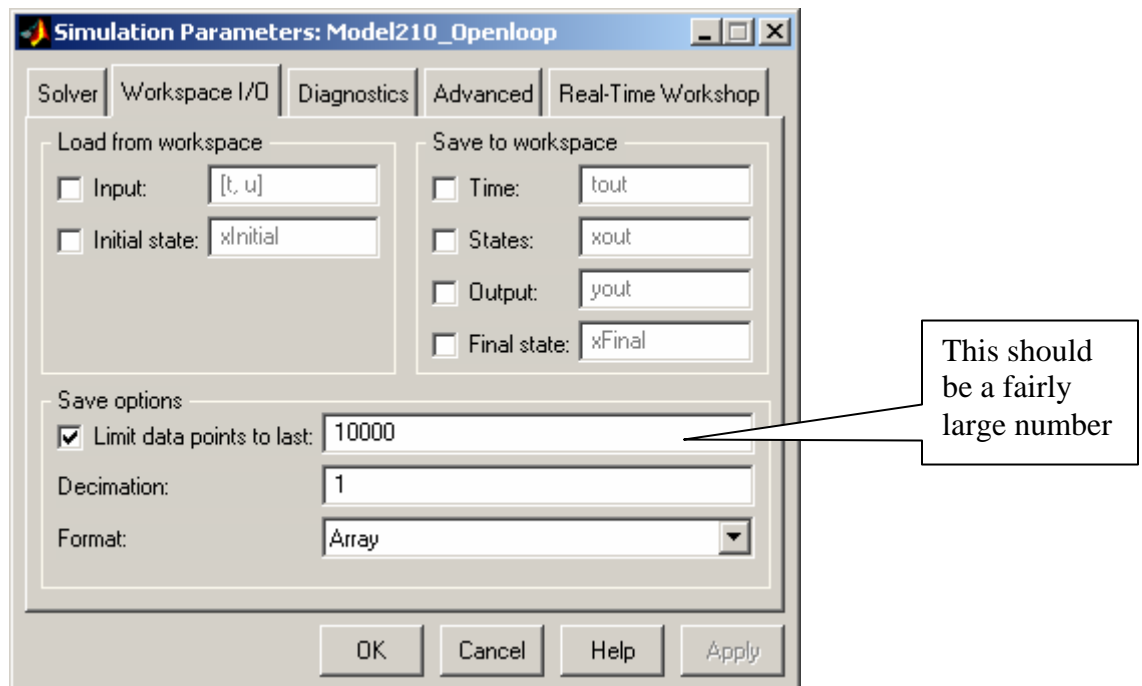
Amplitude
of 0.1 cm

Frequency in
radians/sec,
which is
frequency in
 $\text{Hz} * (2\pi)$.
This is a 2 Hz
sine wave

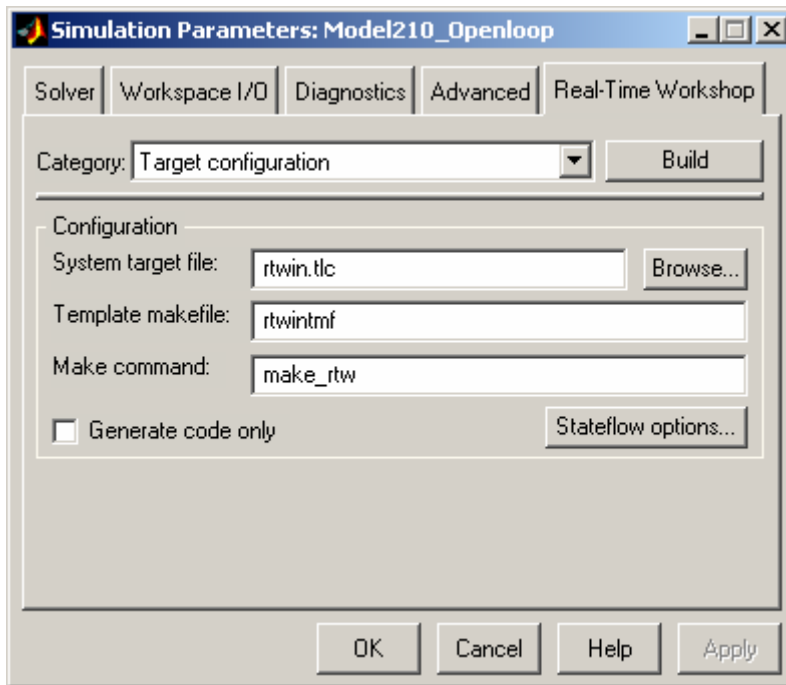
Next we need to set up some simulation parameters. Select **Simulation -> Simulation Parameters**. You will get a screen something like the following:



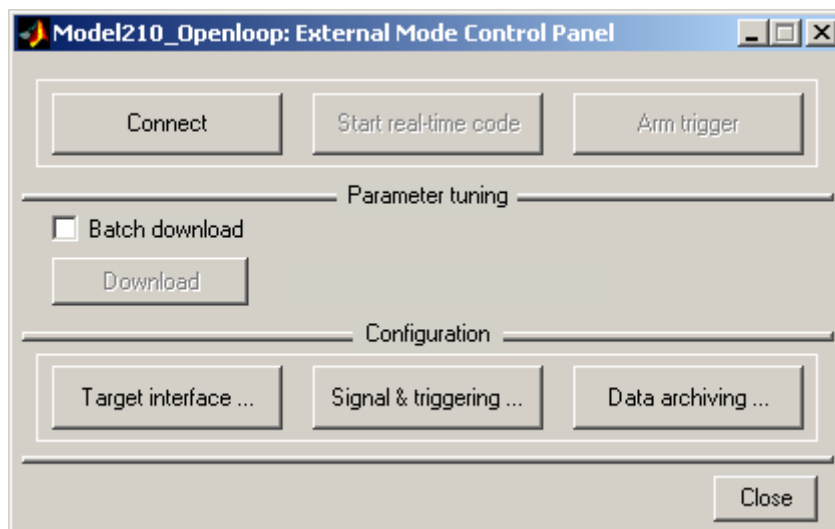
Next, you might want to select Workspace I/O and modify it to look like the following:



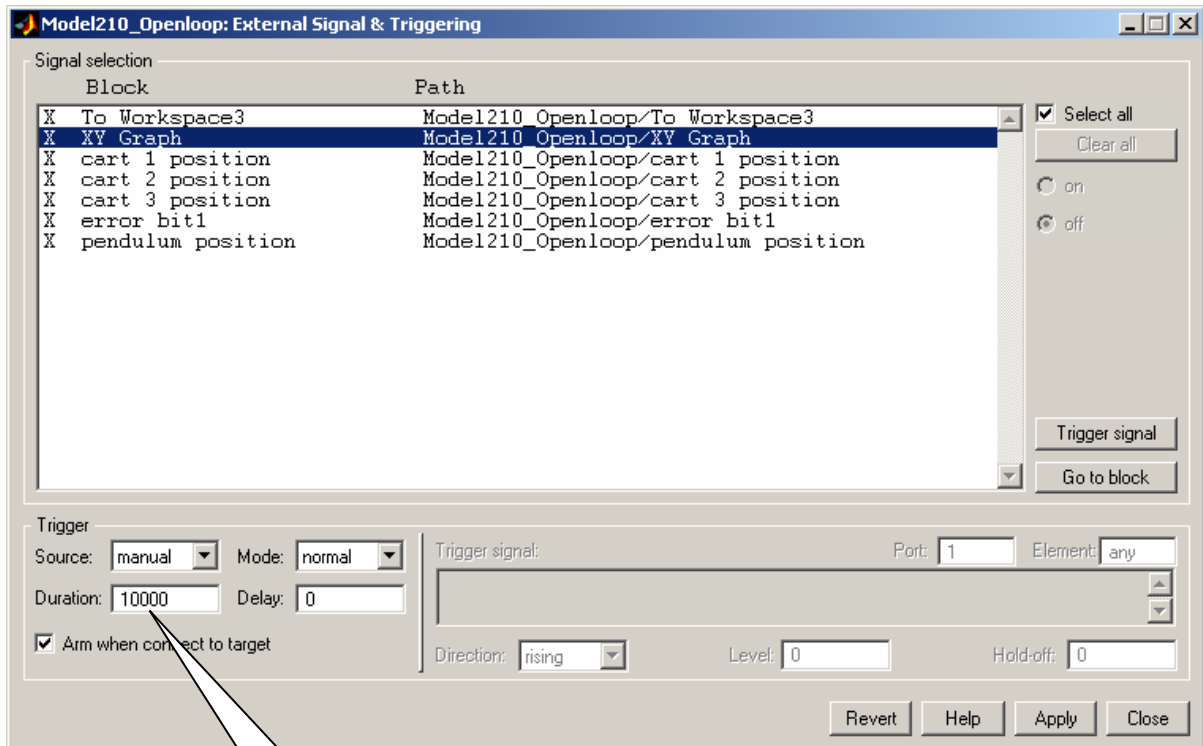
Finally, if you select Real-Time Workshop, it should look like this:



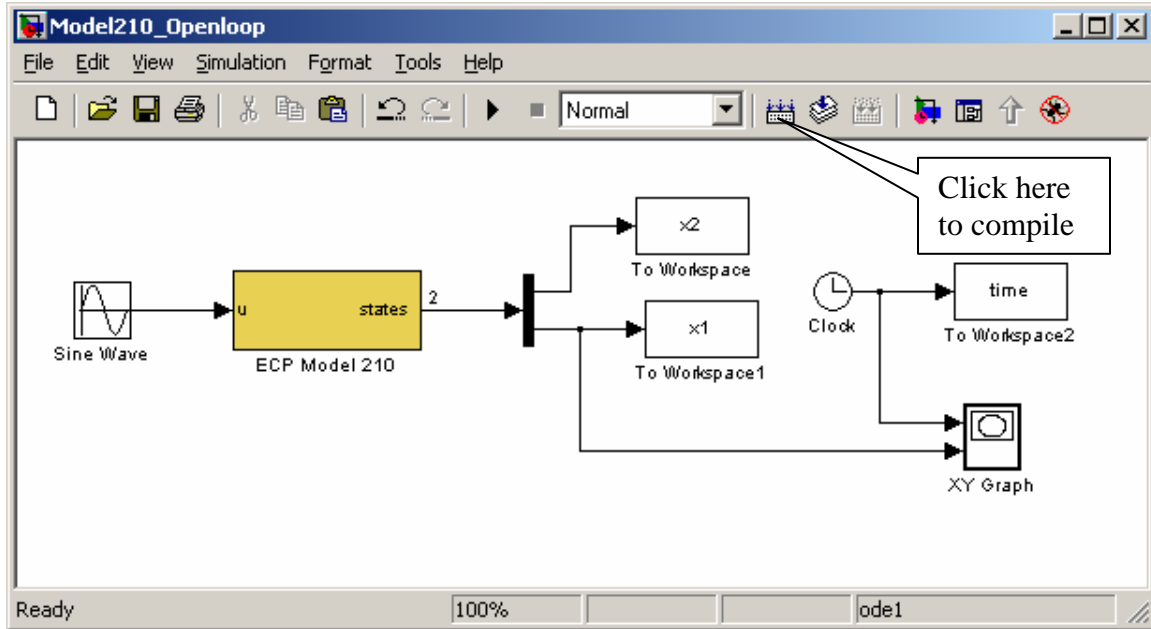
Sometimes, if the system is not saving enough of your data, you will need to select **Tools->External Mode Control Panel** and get the following display



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



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

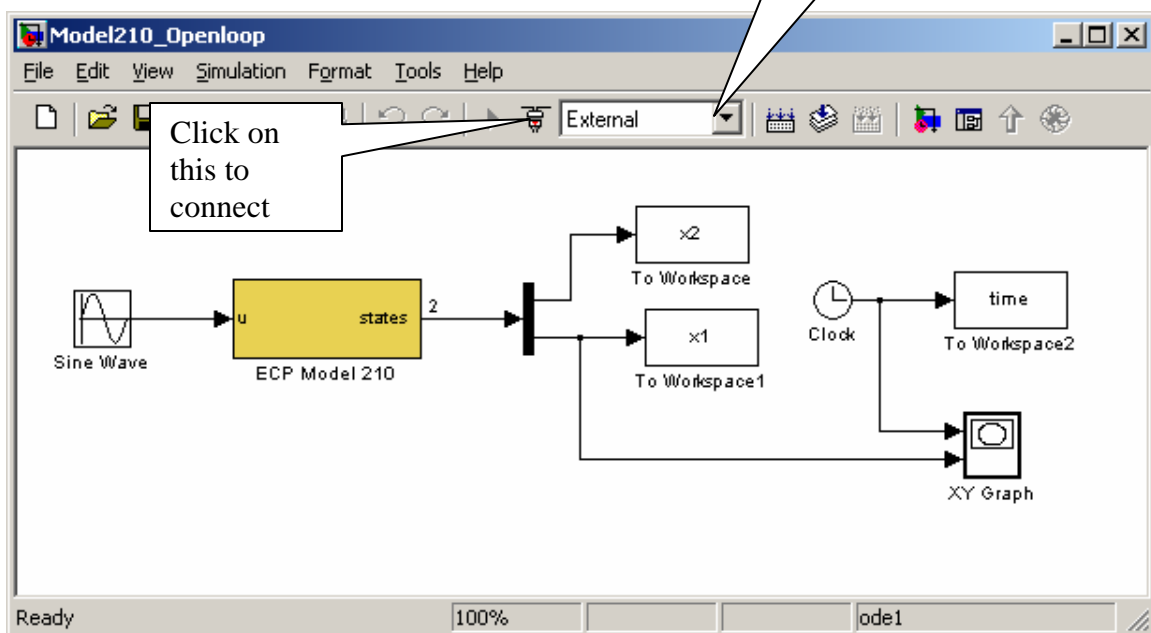


After compiling, you should get the message

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

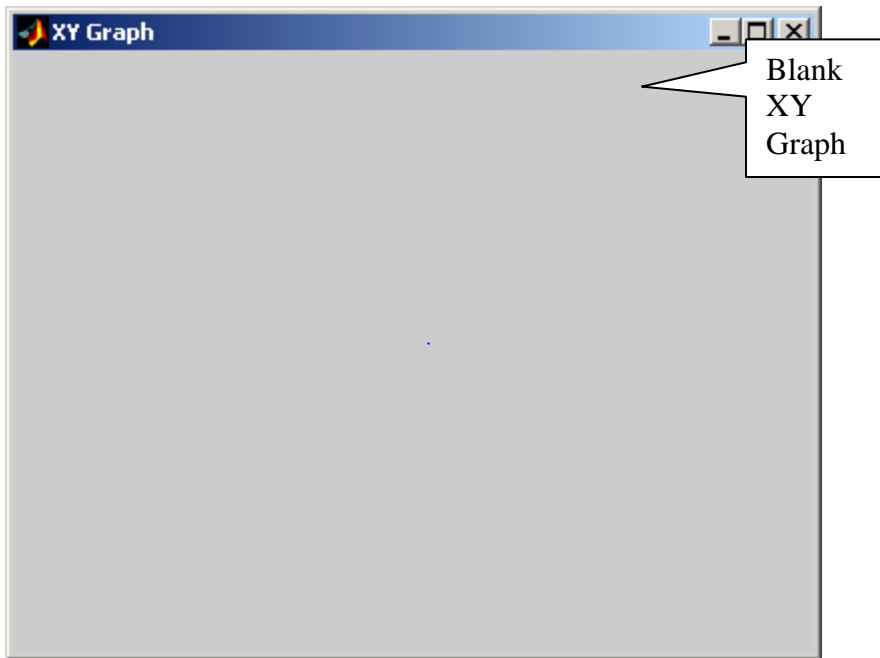
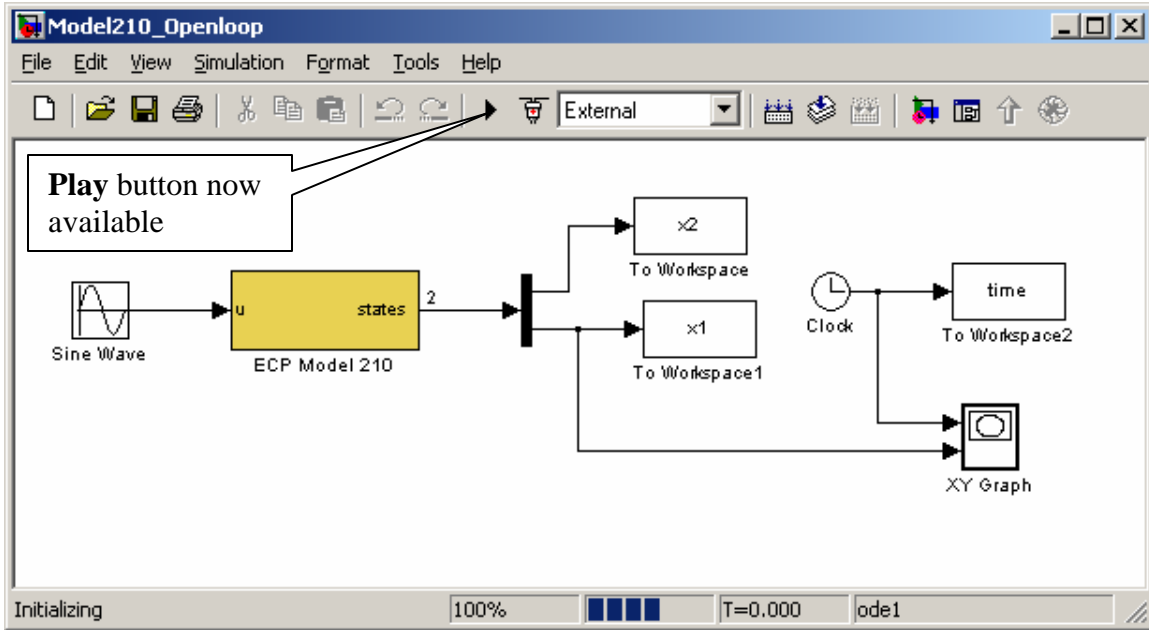
in the Matlab window. Sometimes you get a number of warnings, and if you compile it twice these go away.

Next connect to the ECP System

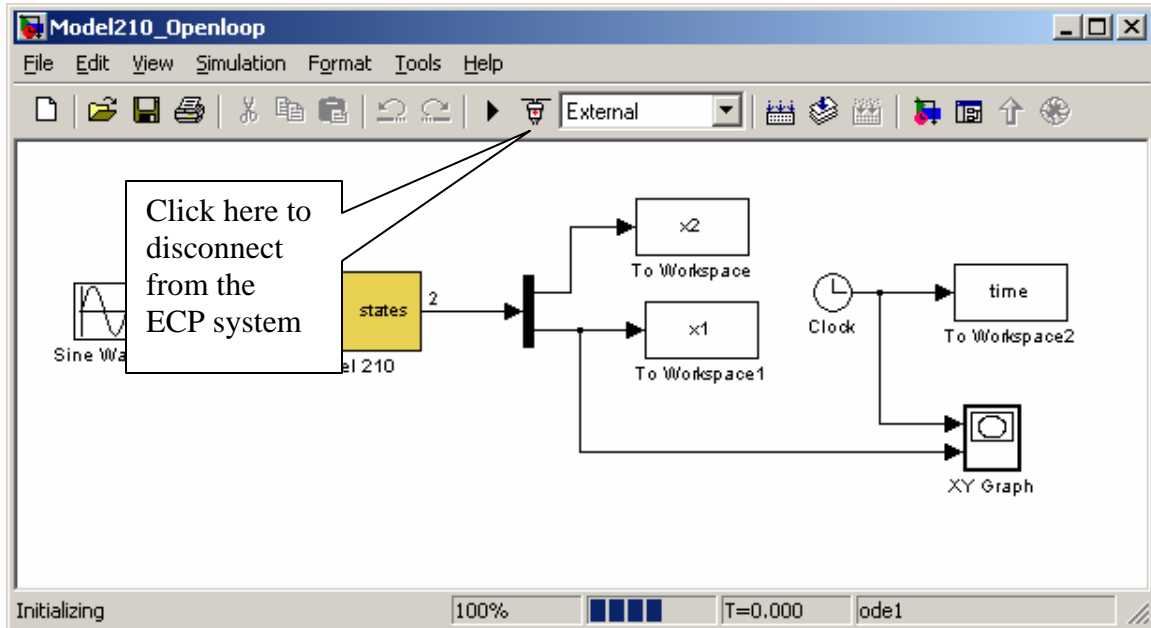


Sometimes when you try and connect you get an error, even though there is nothing wrong. Just clear the window that comes up and try again.

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

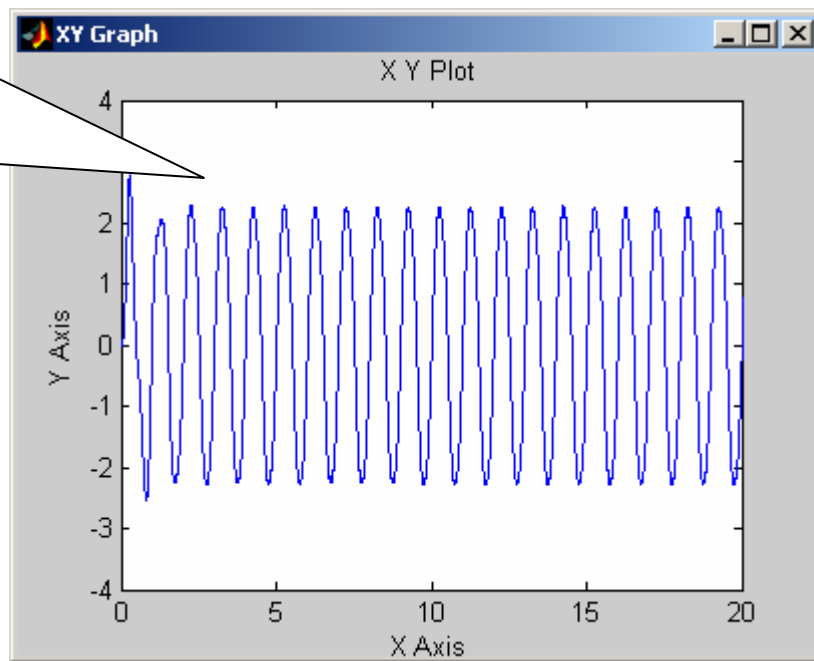


Note that if you want to disconnect from the ECP system, we click on the button next to the **play** button.

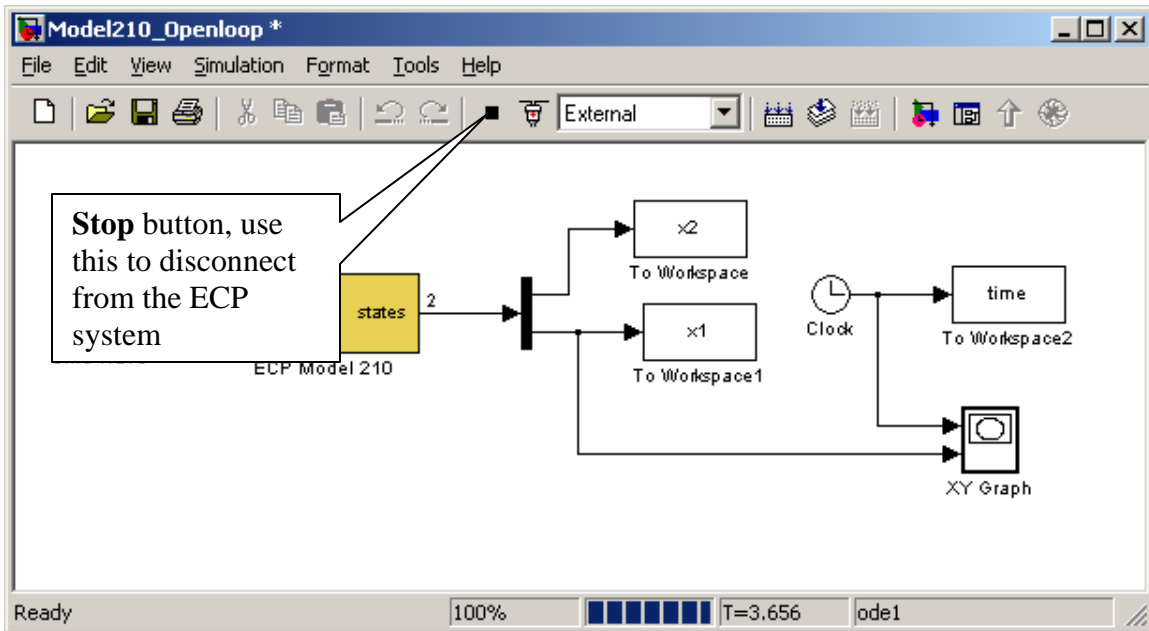


Finally, we can run the system, the XY-graph (for the model I used) will look like

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



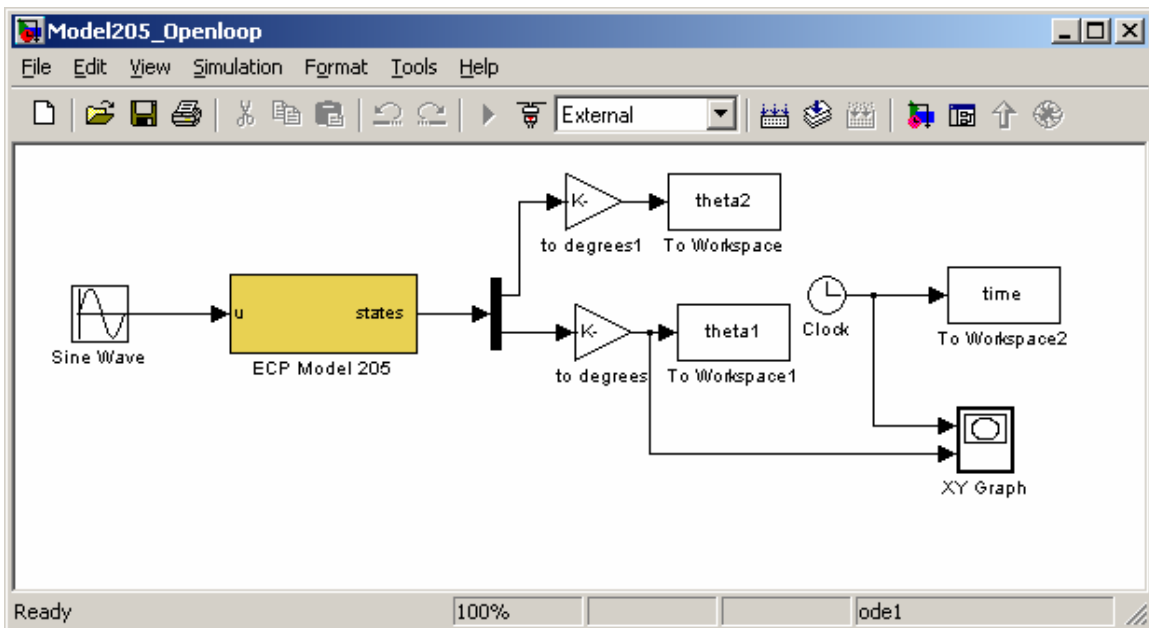
Once we have started running the ECP system, we can disconnect by clicking on the **stop** button. Any data that has been collected will be saved in the workspace.



Model205_Openloop.mdl

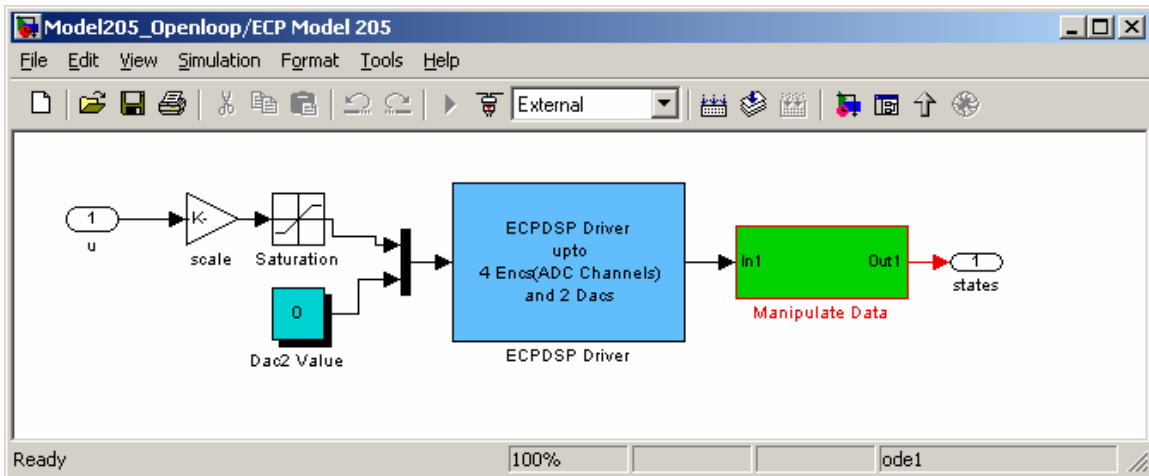
This section only contains the basic differences between the Model 205 and Model 210 files.

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 in our model. To do this we will usually use the Simulink file **Model205_Openloop.mdl**. This is shown below:



This particular implementation has the input signal in **radians** and the position of the two disks measured in **degrees**. For this implementation, the position of the first disk (theta1) is plotted as a function of time as the ECP system runs. (Note that the length of time the XY Graph plots the position of the first cart is hard coded, i.e. it cannot be a variable.) *Note that in general, the Model 205 works in radians.* In this case we have converted to degrees for convenience.

The first thing we need to do is be sure 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 205** block. You should get the following:



This basically shows us that the **ECP Model 205** block is made up of subsystems, most of which you don't have to worry about. To check the address of the DSP card, click on the **ESPDSP Driver** block. You should get the following:

Be sure this address is correct!

Block Parameters: ECPDSP Driver

Subsystem (mask)

Parameters

Base I/O Address

Sample Time (Sec)

Timeout Period (500 ~ 2000)

Hardware Access

Once the Base I/O address, Sample Time, and Timeout Period are OK, click on OK and go back to **Model205_Openloop.mdl**.

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

Block Parameters: Sine Wave

Sine Wave

Output a sine wave where the sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period = $2\pi / (\text{Frequency} * \text{Sample time})$

Number of offset samples = $\text{Phase} * \text{Samples per period} / (2\pi)$

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters:

Sine type: Time based

Amplitude: $3 * (\pi / 180)$

Bias: 0

Frequency (rad/sec): $2 * (2 * \pi)$

Phase (rad): 0

Sample time: 0

Interpret vector parameters as 1-D

OK Cancel Help Apply

Amplitude in radians, which is degrees* $\pi/180$. This is a 3 degree amplitude

Frequency in radians/sec, which is frequency in Hz * $(2 * \pi)$. This is a 2 Hz sine wave