User Manual of I-8438/8838 Matlab Embedded Controller

v1.1

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

Table of Contents

1. Introduction

I-8438/8838 is the ICP DAS MATLAB Embedded Controller solution built in Ethernet and series interface with I/O expansion slot for Matlab development environment. For this application there are over 20 I/O bridges and system-level Simulink Blocksets have been developed. By using Simulink development environment and these Matlab Driver's blocksets, control algorithm can be easily constructed and verified without writing any code. Once the algorithm has been verified, by pressing a build button, users can convert a model to executable code, and download it to controller for test or practical application via RS232, Ethernet, RS485 (available soon) and even Modem (available soon). Furthermore, engineers can put more focus on advanced control algorithm design and development.

1.1. Hardware Specifications for I-8438/8838

General environment

Operating temperature: -25°C to +75°C
Storage temperature: -30°C to +85°C
Humidity: 5 ~ 95%
Built-in power protection & network protection circuit

System

CPU: RDC, 80MHz, or compatible
SRAM: 512K bytes
FLASH ROM: 512K bytes
COM ports for the I-8438 & 8838
    COM1=RS-232, COM3=RS232/RS485, COM4=RS-232
    Ethernet: 10 BaseT
RTC, NVRAM & EEPROM
Program download from COM1 & Ethernet
Built-in 64-bit hardware unique serial number
Built-in Watchdog Timer
I/O Expansion Slot
    4-slot for I-8438
    8-slot for I-8838

1.2. Features

The I-8438/8838 series modules software driver for MATLAB development perfectly and easily combines with MATLAB/Simulink/Stateflow. With the support of library of many extended powerful blocks for the I-8000 series module I/O hardware driver, the sophisticated tasks of creating, analyzing, and simulating block diagram models can all be solved conveniently with MATLAB/Simulink/Stateflow. The I-8000 Series Modules Driver, in conjunction with MATLAB/Simulink/Stateflow, gives you the features you need for advanced controller development.

Have a look at some of the features of I-8000 Series Modules Driver:

1. **RS232 & Ethernet Communication Interfaces:** The I-8000 Matlab solution gives you the function needed for your control program to download and upload experimental data through the media of RS232 & Ethernet communication interfaces.

2. **Easy-to-use platform:** The I-8000 Matlab solution provides GUI interface for easy application, which enables you to communicate conveniently with the I-8xx8 target hardware.

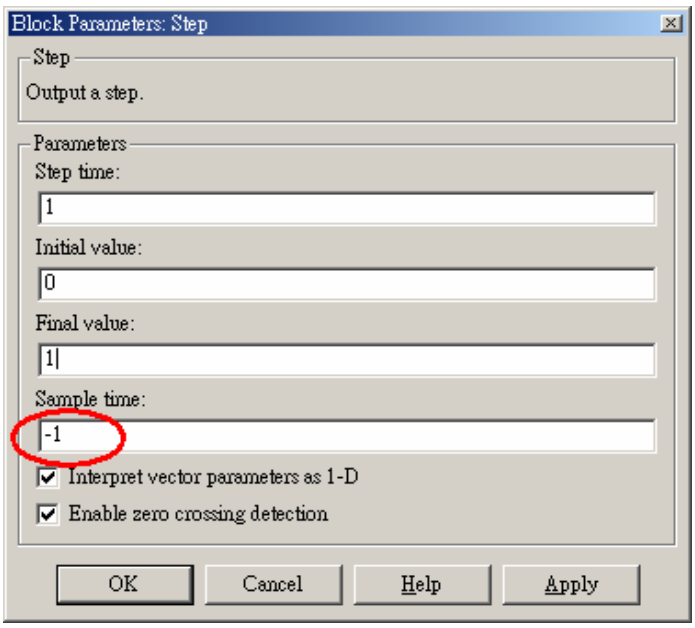
3. **Reduce the design cycle of your product:** This solution, in conjunction with MATLAB/Simulink/Stateflow, provides you the model-based control design approach. The model-based control design approach is a timesaving and cost-effective approach, allowing control engineers to work with a single model of a function or a complete system in an integrated software environment.

4. **The extensibility of I/O driver blocks:** This Matlab solution is tailored for the I-8438/8838 control system, which provides 4 or 8 expansion slots. Therefore, you can expand your I/O capability if necessary. Currently, this software driver provides over 20 I/O blocks to cooperate with Matlab development environment, which including DI, DO, DIO, AI, AO, Relay, and Encoder blocks. More I/O blocks will be developed in the future.

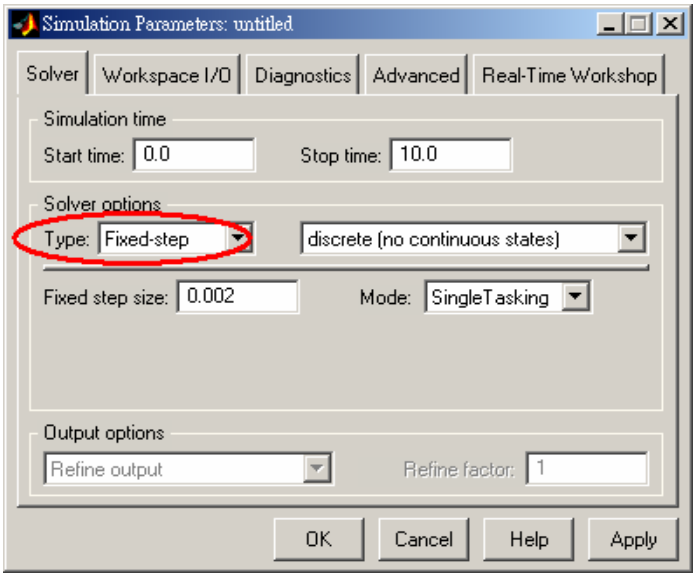
1.3. Limitations

The ICPDAS I-8000 series module software driver for MATLAB only supports *single tasking* and *Fixed-step* modes, due to the limitations of the RTW Embedded Coder.

1. **Single tasking:** In Simulink, *single tasking* means that only one sample rate can be used in the whole control system. That is, every block must have the same sampling rate. It is suggested that users set the *Sample time* to be –1 when the option is available in the block.



2. **Fixed-step:** Because the RTW 4.x or 5.0 have not supported variable step time, the *Solver options* on the *Simulation Parameters* dialog box can only be set to *Fixed-step*.



Furthermore, the RTW Embedded Coder does not support the following built-in Simulink blocks yet:

1. **Simulink\Continuous**
   - No blocks in this library are supported

2. **Simulink\Discrete**
   - First-Order Hold

3. **Simulink\Function and Tables**
   - MATLAB Fcn

4. **Simulink\Math**
   - Algebraic Constraint
   - Matrix Gain

5. **Simulink\Nonlinear**
   - Rate Limiter

6. **Simulink\Signals & System**
   - Bus Selector
   - IC

7. **Simulink\Sinks**
   - XY Graph
   - Display
   - To File

8. **Simulink\Sources**
   - Clock
   - Chirp Signal
   - Pulse Generator
   - Ramp
   - Repeating Sequence
   - Signal Generator

1.4. Module list supported for Matlab Driver

The following table is a list of currently supported I-8000 I/O modules by Matlab Driver. They include DI, DO, DIO, AI, AO, relay, and encoder modules. If you want to get more information about the specification and function of these modules, please visit the website (http://www.icpdas.com/) to obtain the current driver situation and download the related documents and latest driver.

Type	Description	Module Model
DI	Digital input module	I-8040, I-8051, I-8052,I-8053, I-8058
DO	Digital output module	I-8041, I-8056, I-8057
DIO	Digital input & output module	I-8042, I-8054, I-8055, I-8063
AI	Analog input module	I-8017H
AO	Analog output module	I-8024
Relay	Relay output module	I-8060, I-8064
Encoder	Encoder counter board	I-8090

2. Software Installation

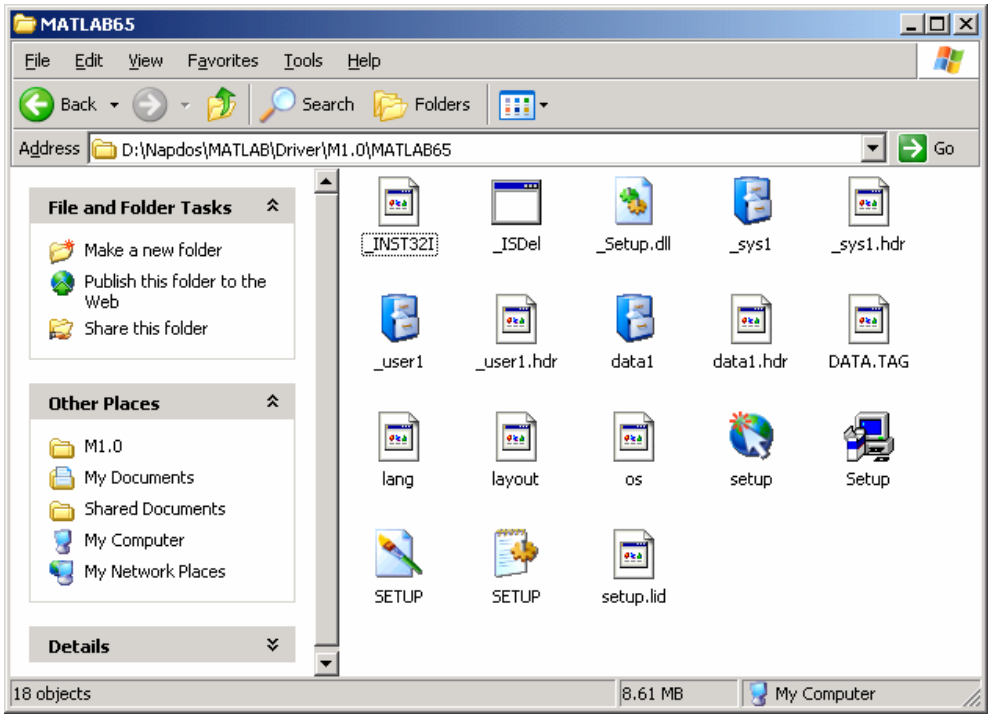
Before installing the MATLAB solution software, users need to double check here is a Matlab software development system installed in the operation system. The requirement toolbox of the MATLAB software for ICPDAS I-8000 MATLAB solution is listed as follows:

- MATLAB 6.1 or 6.5 below installed.
- Simulink 4.1 or 5.0 below installed.
- Real-Time Workshop 4.1 or 5.0 installed.
- Real-Time Workshop Embedded Coder 2.0 or 3.0 installed.
- Stateflow and Stateflow Coder 4.1 or 5.0 (not necessary).

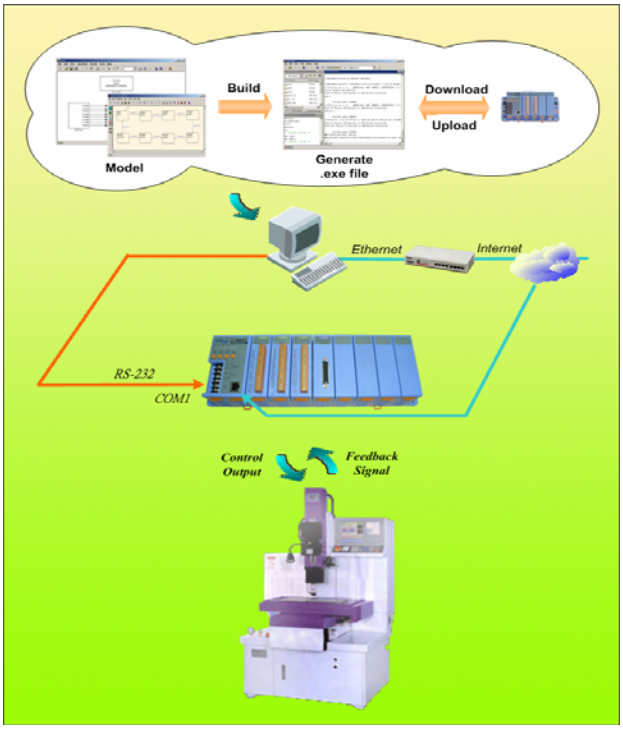
Note that current solution is only tested in the Window system, which is Win98/2000/XP. In the following sections, we will demonstrate the installation procedure step by step to make everything registered correctly.

2.1. Driver installation

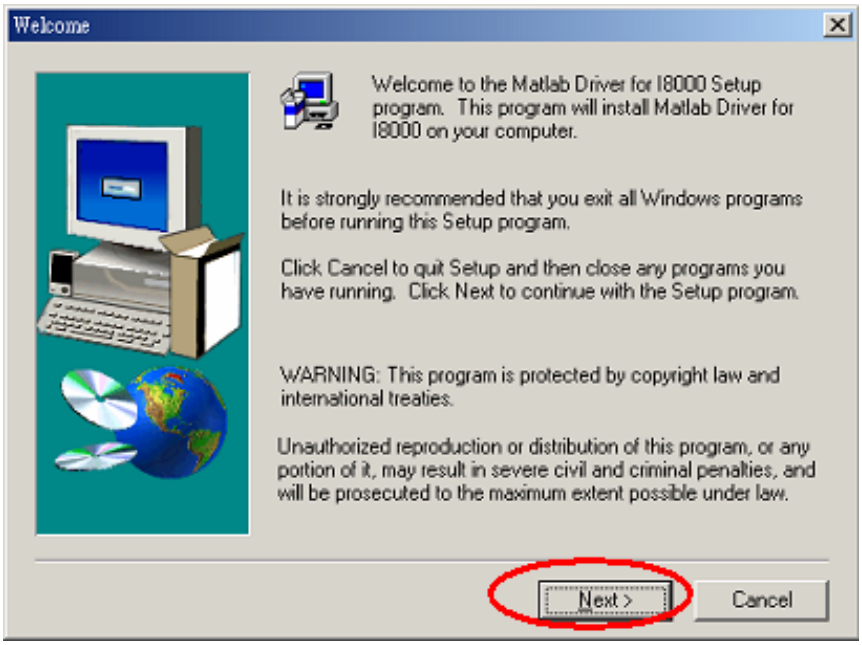
Step 1: Insert the product CD and enter the directory, Napdos\MATLAB\Driver\M1.0\MATLAB65 (for MATLAB 6.5) or Napdos\MATLAB\Driver\M1.0\MATLAB61 (for MATLAB 6.1). The files listed should look like the figure below. Double click *Setup.exe* to start the installation procedure.



Step 2: An ICP DAS MATLAB solution logo appears as shown in the figure below, and then the *Welcome* dialog box presents to prompt users to follow the installation steps. Press *Next* button to continue.



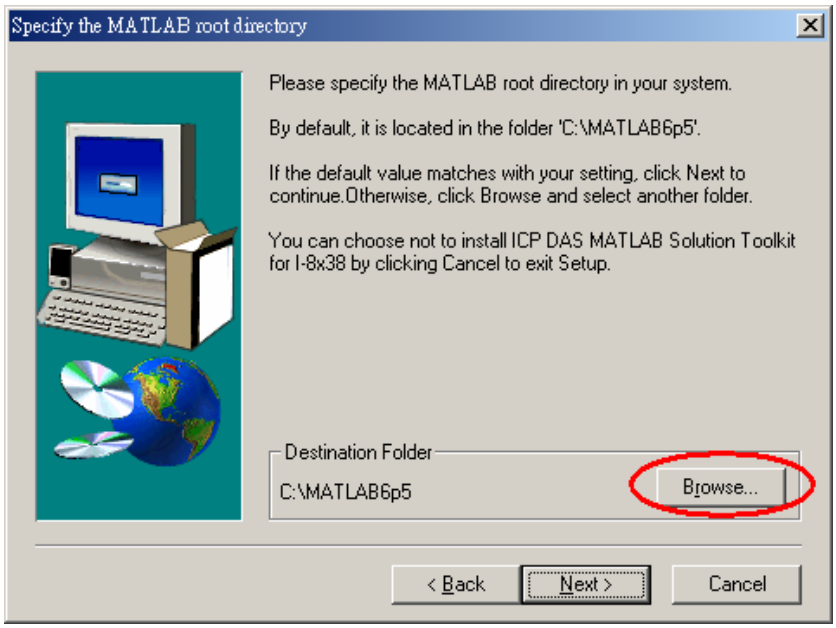
ICP DAS MATLAB solution logo



The “Welcome” dialog box

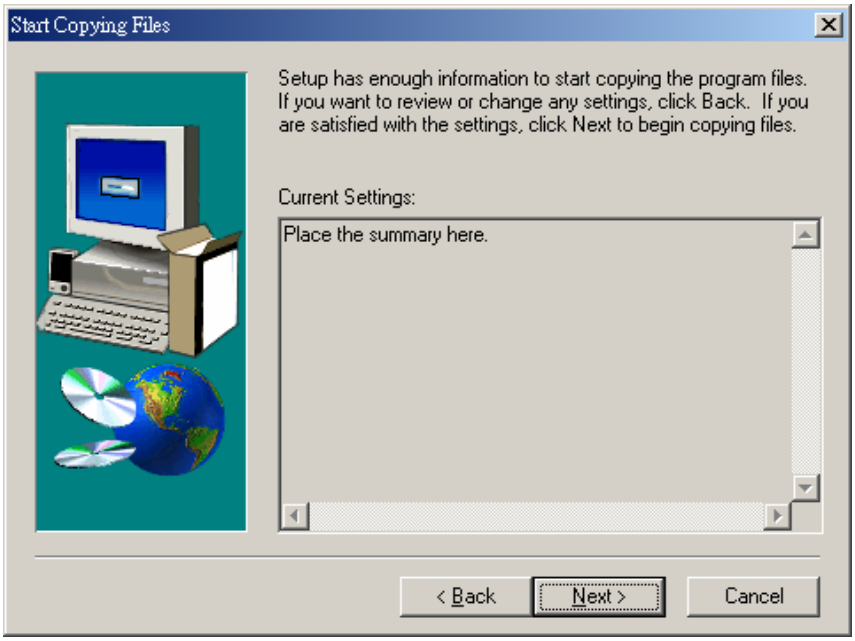
Step 3: The Installation directory for MATLAB dialog box shows the current directory information that MATLAB software has been installed, as

shown in the figure below. The default MATALB folder is "C:\Matlab6p5". **If MATALB 6.1 or 6.5 was not installed on the default path, users must click the Browse button to choose the correct path. Otherwise, the MATLAB Solution Toolkit will not work properly.** Press Next to the next step.

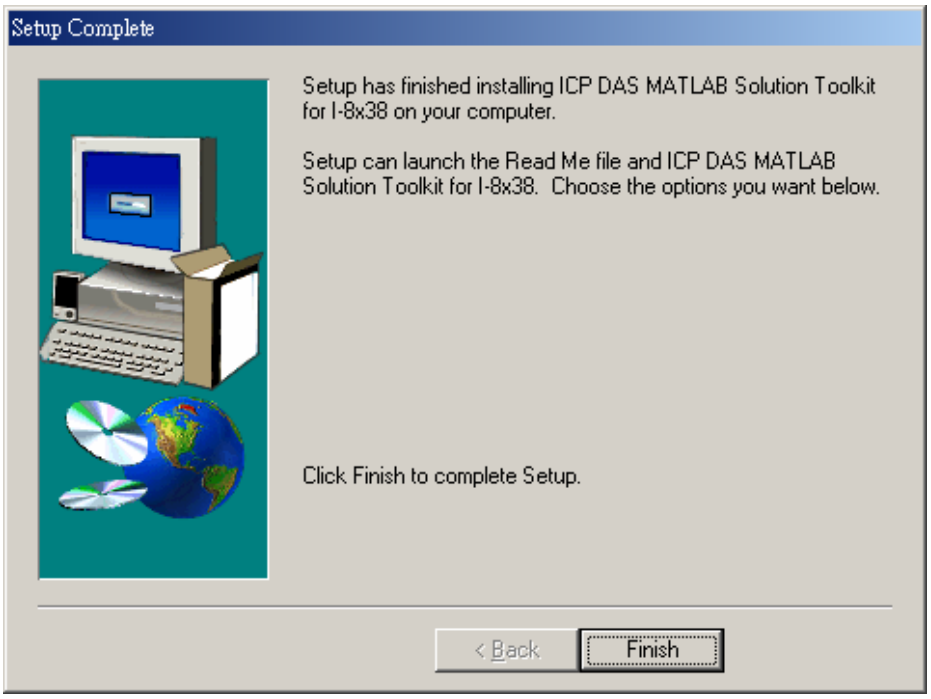


Specify the MATLAB root directory

Step 4: Then a dialog box is present to prompt you installing the software. Press Next to start copying files.

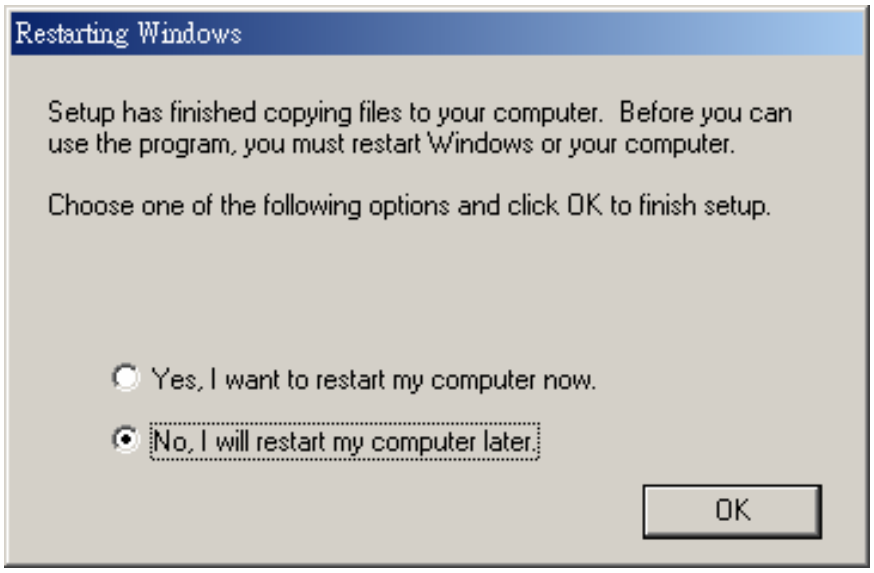


Step 5: After copying files has been completed successfully, the Setup Complete dialog box appears. Click Finish to exit the installation program.



Installation is completed successfully

Step 6: After the installation has been completed, re-boot your system to let the settings to take effect.



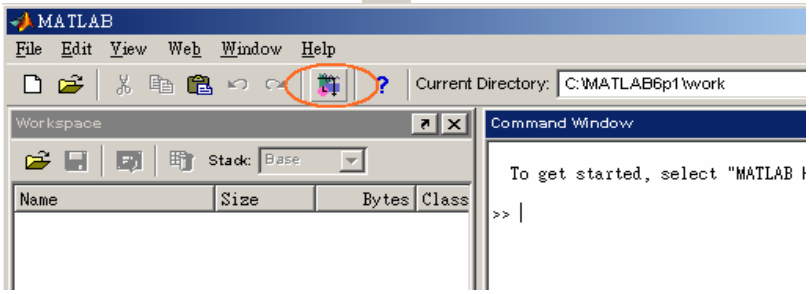
Reboot your system

3. How to work with MATLAB/Simulink

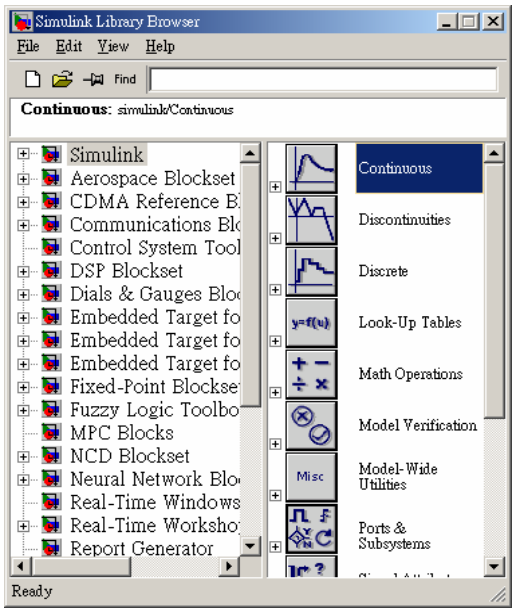
This chapter gives an example to demonstrate how to construct your control model with I-8xx8 driver blocks by using MATLAB/Simulink. Furthermore, you can also learn how to build the model into an executable file by employing RTW and RTW Embedded Coder. In the last two sections of this chapter, we will also show you how to cooperate the Stateflow Coder and Fixed-Point with the I-8xx8 driver blockset. After you completely follow this chapter instruction, it is expected that you can construct your own model, and build it into an executable file and download it to the I-8xx8 target system for application. And finally you can easily start your experiment and upload the data for further analysis.

3.1. Create a control model using Simulink

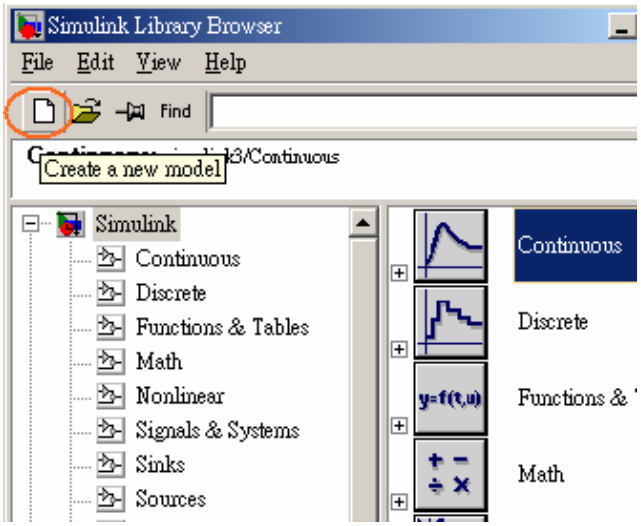
To start Simulink, you must first start MATLAB. And then you can start Simulink by clicking the Simulink icon on the MATLAB toolbar.

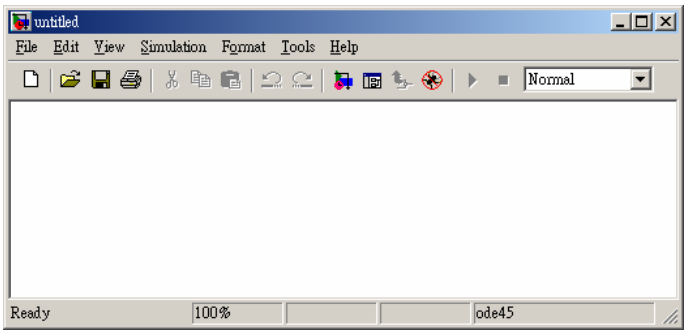


On Microsoft Windows platform, starting Simulink displays the Simulink Library Browser window as shown below.

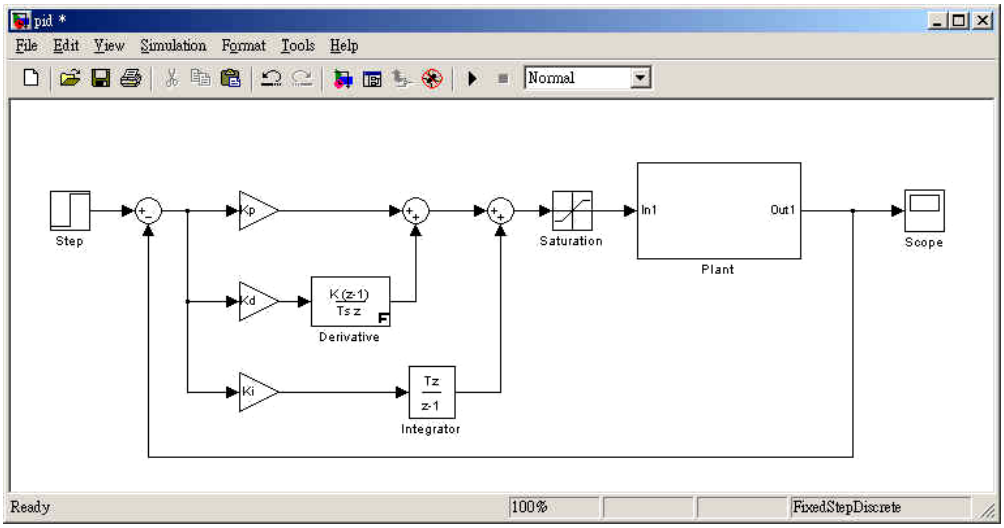


Then click  on the Library Browser's toolbar (Windows only). Simulink opens a new model window, and then you can start to construct your own control model on the blank area of the model window.



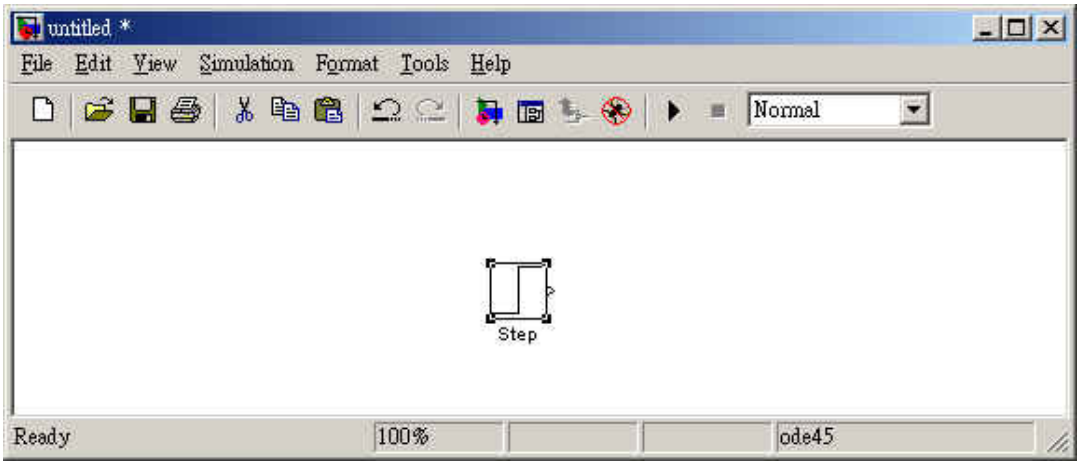


Here, we offer a simple PID controller model to you, and show you how to create a model using many of the model-building commands and actions. The block diagram of the model looks like the figure below.



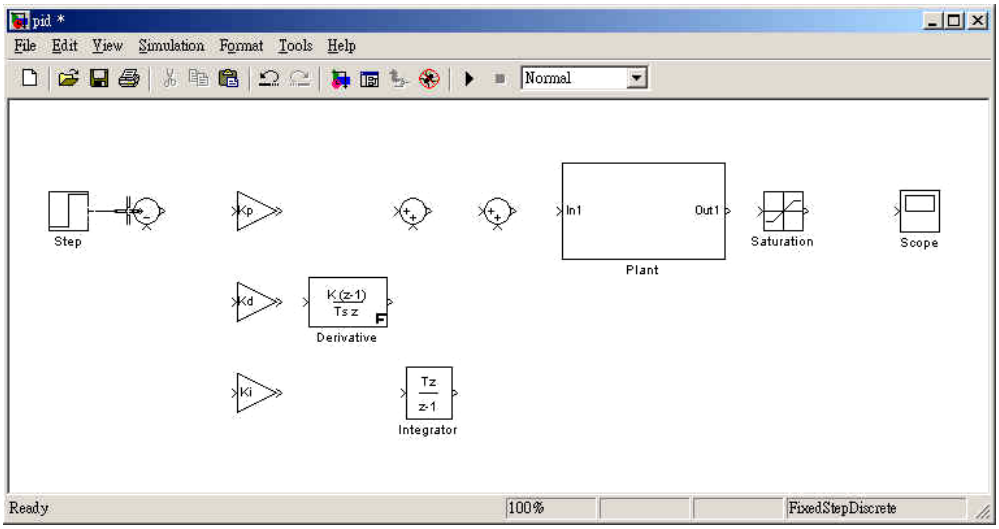
To create this model, you need to copy blocks to the model from the Simulink block libraries. For example, to copy a *Step* block to the model window, follow these steps:

1. First expands the Library Browser tree to display the blocks in the *Sources* library. Do this by clicking the *Sources* node to display the *Sources* library blocks.
2. And then click the *Step* node to select the *Step* block.
3. Now drag the *Step* block from the browser and drop it in the model window. Simulink creates a copy of the *Step* block at the point where you dropped the node icon. Now the model window should look like the figure below.



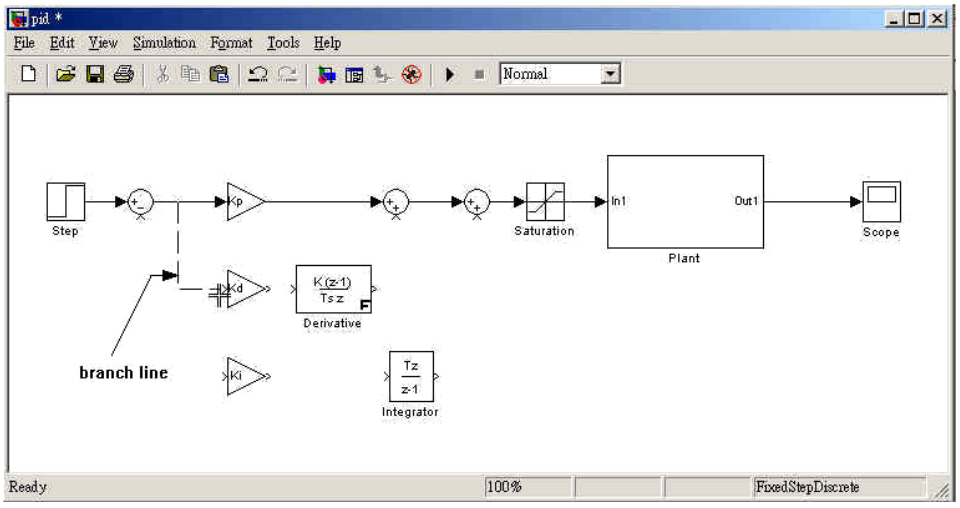
Copy the rest of the blocks in a similar manner from other libraries to the model window. You can move a block from one place in the model window to another by dragging the block. You can also move a block a short distance by selecting the block, then pressing the arrow keys.

After all the blocks are copied to the model window, it's time to connect the blocks. To connect one block with another, just select one block, and position the mouse pointer over the output port on the block. Notice that the cursor shape changes to crosshairs. Hold down the mouse button and move the cursor to the input port of another block. Now release the mouse button when the cursor shape changes to double-lined crosshairs as shown in the figure below, and then the blocks are connected.



However, we also need to connect one line to the input port of another block. This line is called *branch line*. Drawing a branch line is slightly different from drawing the line you just drew. To weld a connection to an existing line, follow these steps:

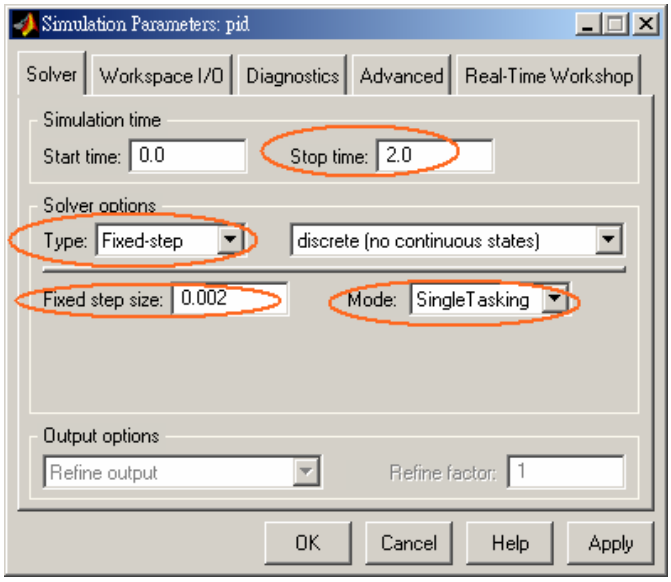
1. First, position the pointer on the existing line.
2. Press and hold down the Ctrl key (or click the right mouse button). Press the mouse button, and then drag the pointer to the input port of another block.
3. Release the mouse button when the cursor shape changes to double-lined crosshairs as shown in the figure below.



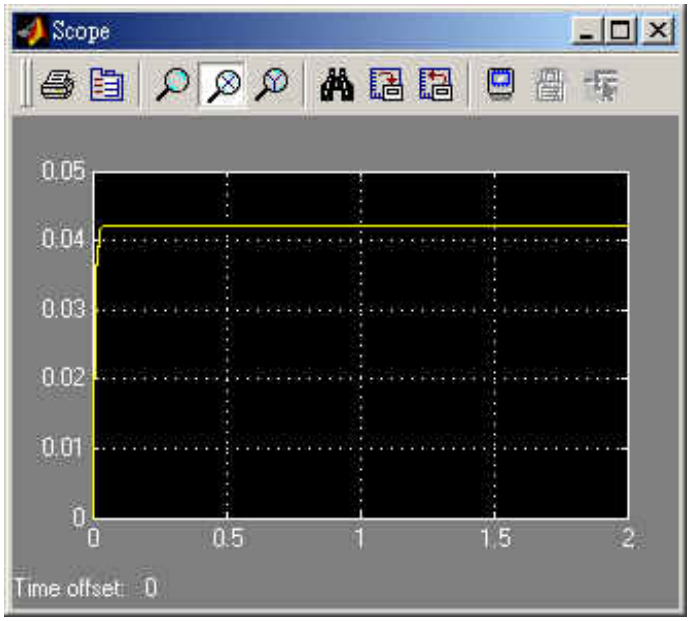
Finish making block connections. When these steps are done, your model should look like the model at the beginning of this section. Now open the Scope block to view the simulation output. Set the simulation parameters by choosing *Simulation parameters* from the *Simulation* menu. Set the options as follows:

- Stop time: *2.0*
- Solver options: *Fixed-step*
- Fixed step size: *0.002*
- Mode: *Single Tasking*

Here, we set Solver options to **Fixed-step** and Mode to **Single Tasking**. The reason is that the RTW Embedded Coder does not support variable step time yet. To compare the result of simulation with experiment fairly, the setting is suggested.



Close the *Simulation parameters* dialog box by clicking the OK button. Simulink applies the parameters and closes the dialog box. Choose *Start* from the simulation menu or click the *Start* button on the model window's toolbar and watch the traces of the Scope block's input.

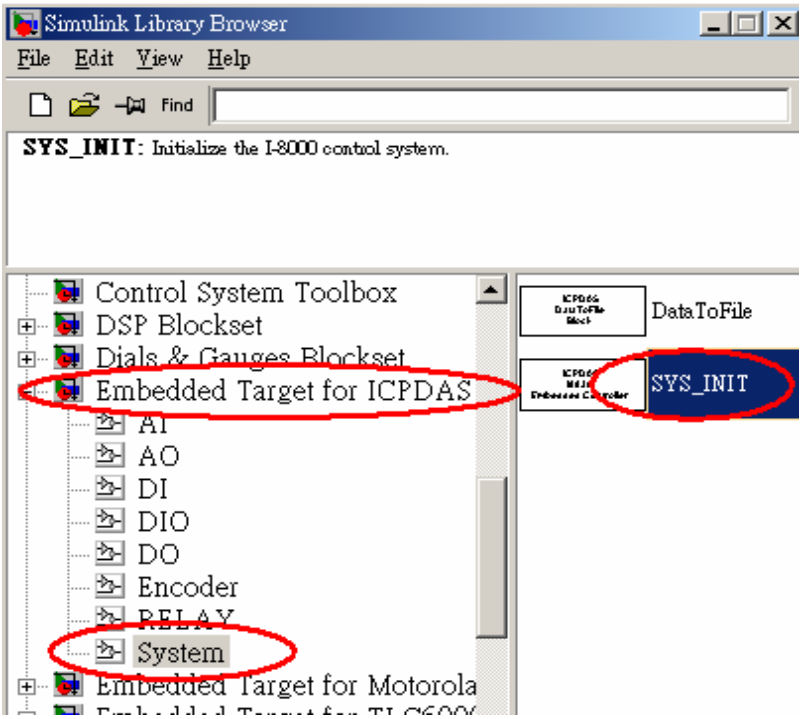


The simulation stops when it reaches the stop time specified in the Simulation Parameters dialog box or you choose *Stop* from the *Simulation* menu or click the *Stop* button on the model window's toolbar.

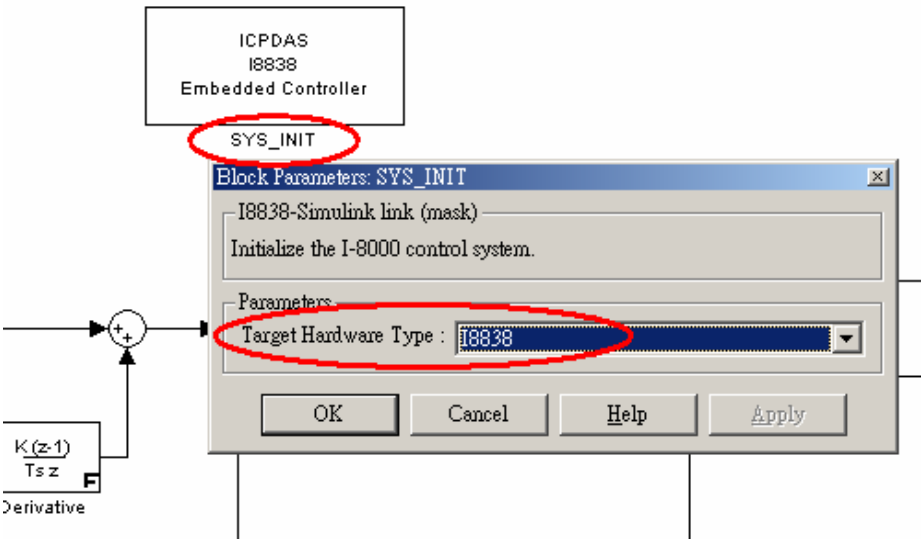
3.2. Simulink model with ICPDAS driver

If the simulation output was satisfied, you can replace the built-in Simulink blocks with the I-8000 driver blocks. To add an I-8000 driver block to the model, follow these steps:

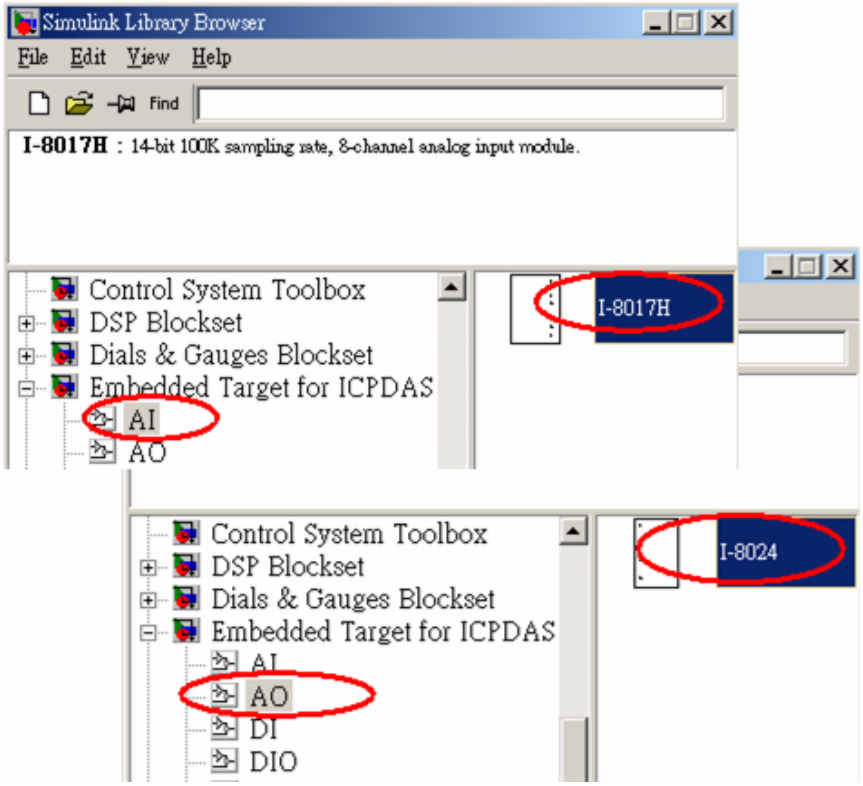
Step 1: Insert a SYS_INIT block from the System block library.



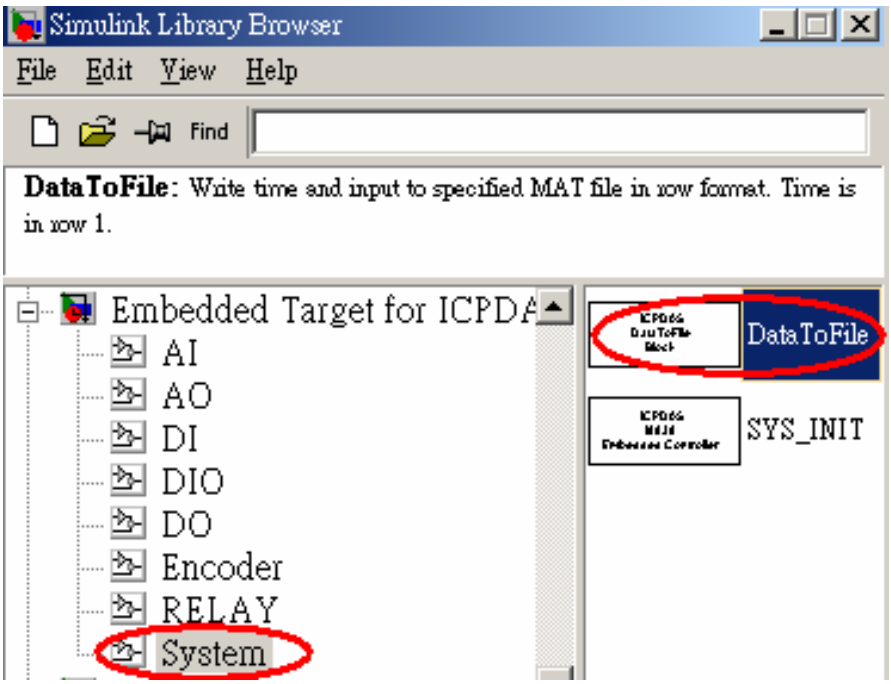
Step 2: Double-click on the SYS_INIT block to open the SYS_INIT dialog box. Then select the correct type from the popup menu on the dialog box. Here, we select the type as I8838.



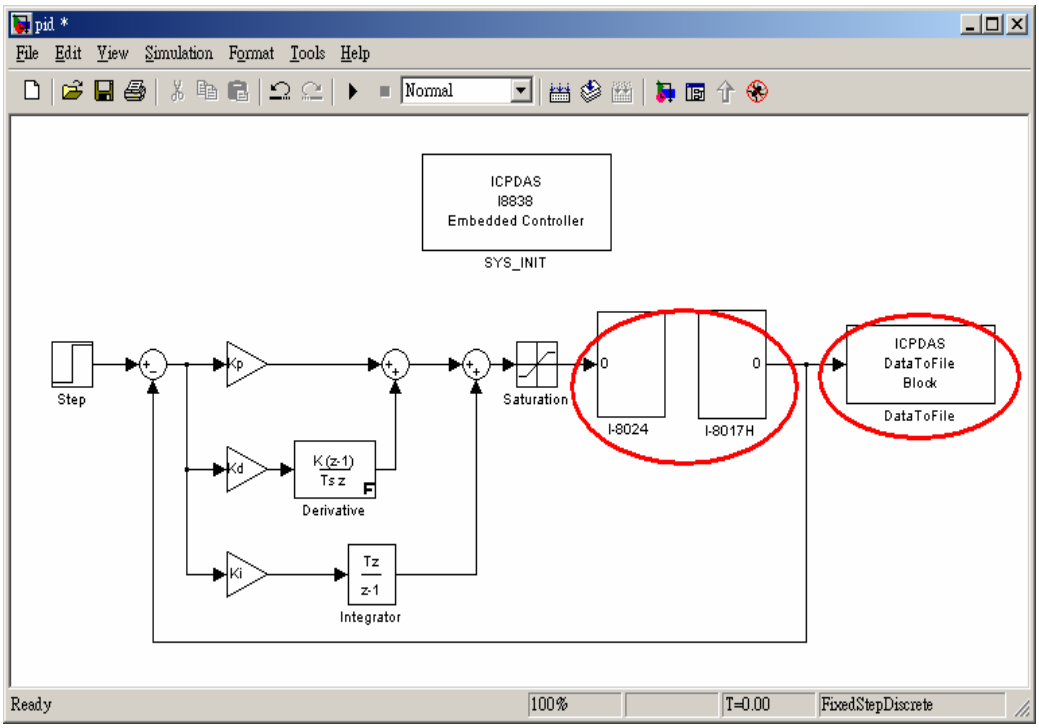
Step 3: Remove the Plant subsystem block and copy an I-8024 and an I-8017H block from the AI and AO library respectively.



Step 4: Replace the Scope block with a DataToFile driver block.



Step 5: Connect the blocks as shown in the following figure.

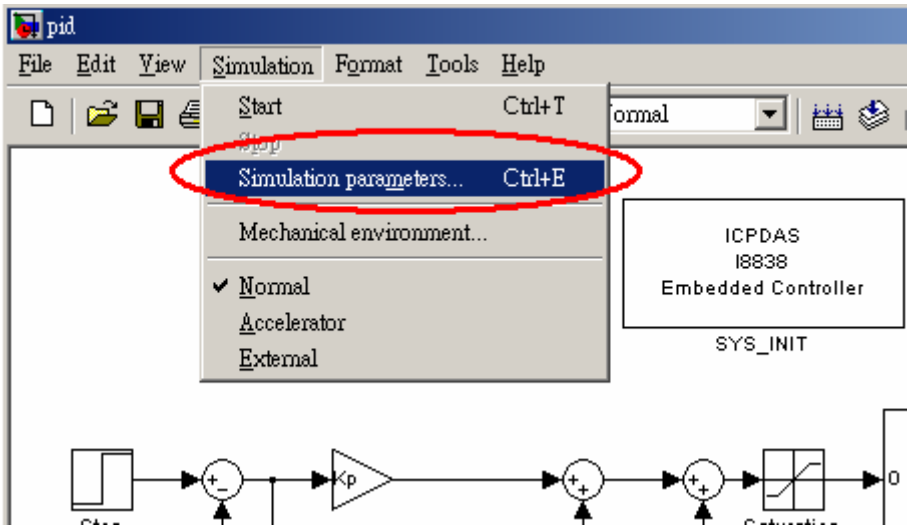


After the above steps are done, you can start to build your control model with the ICPDAS I-8000 driver blocks.

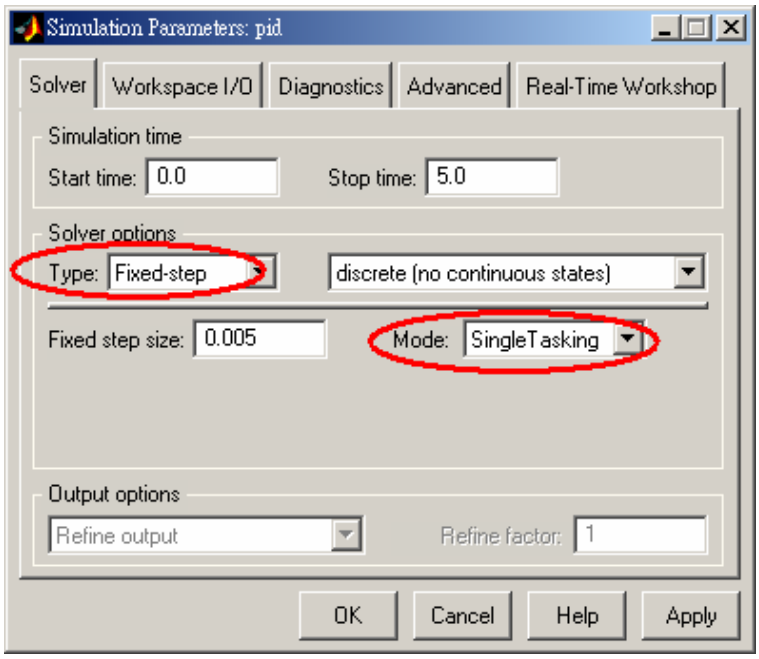
3.3. Build the program by RTW

In this section, we will show you how to convert the control model created in the previous section into an .exe file by RTW. To do this, just follow the following steps:

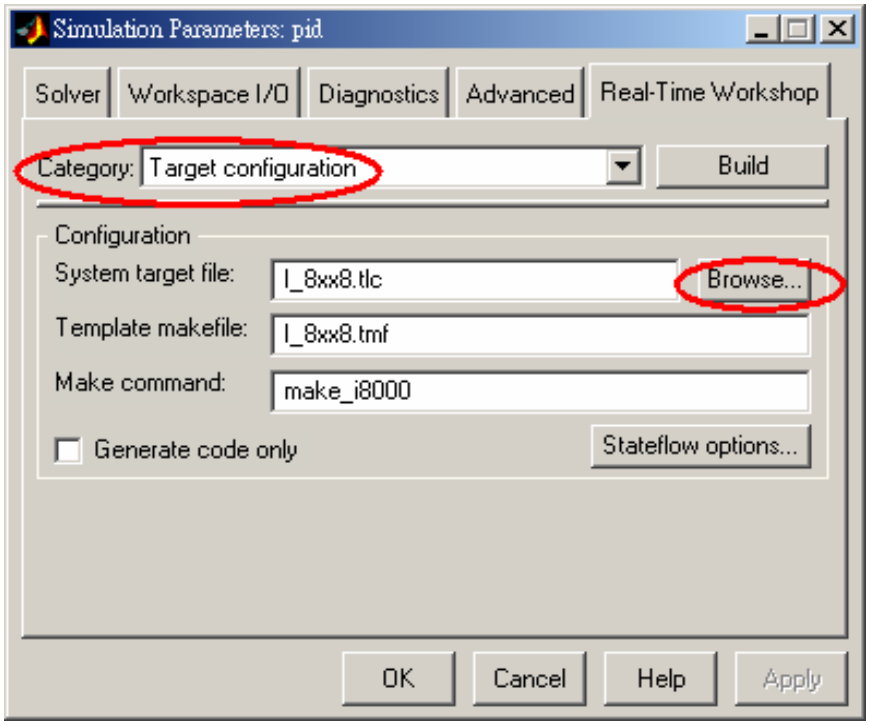
Step 1: Open the Simulation Parameters dialog box by choosing *Simulation parameters* from the *Simulation* menu.



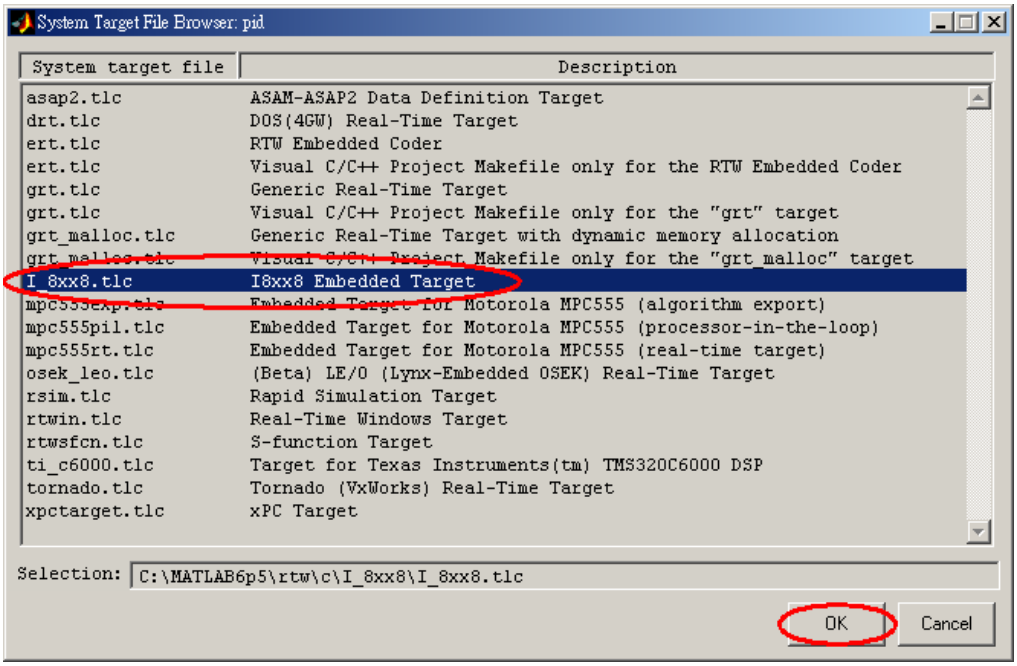
Step 2: On the dialog box that displays, select Type as *Fixed-step*, Mode as *Single Tasking* in the "Solver options" field.



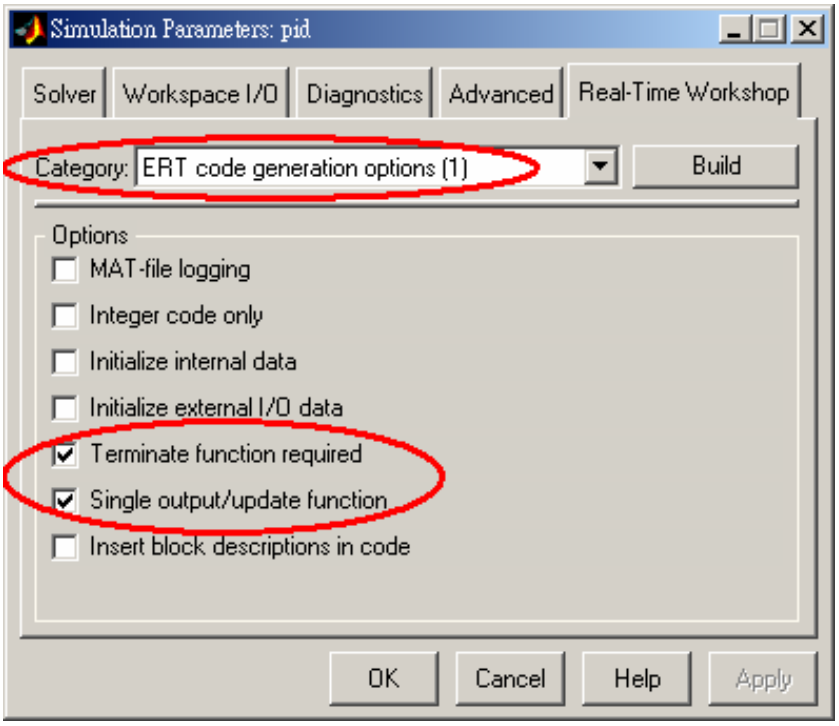
Step 3: Then click the “Real-Time Workshop” tab and the pane changes. On the pane that shows up, select “Target configuration” from the “Category” field. Then click the Browse button to open the “System Target File Browser” window.



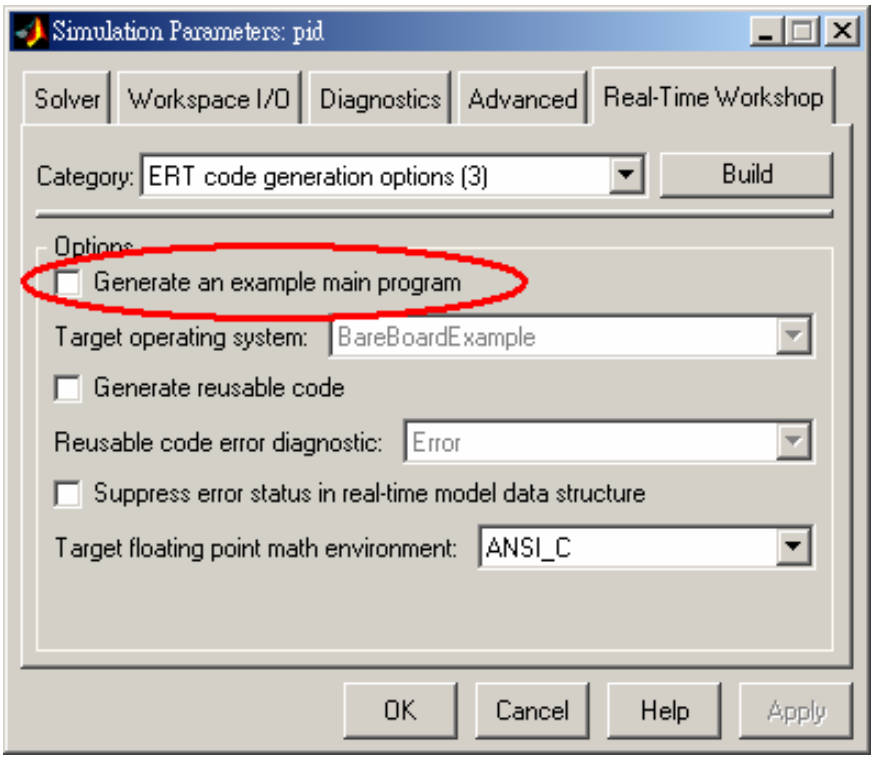
Step 4: On the System Target File Browser dialog, select the correct system target file from the list and then click the OK button to close the dialog box. Here, we choose I_8xx8.tlc.



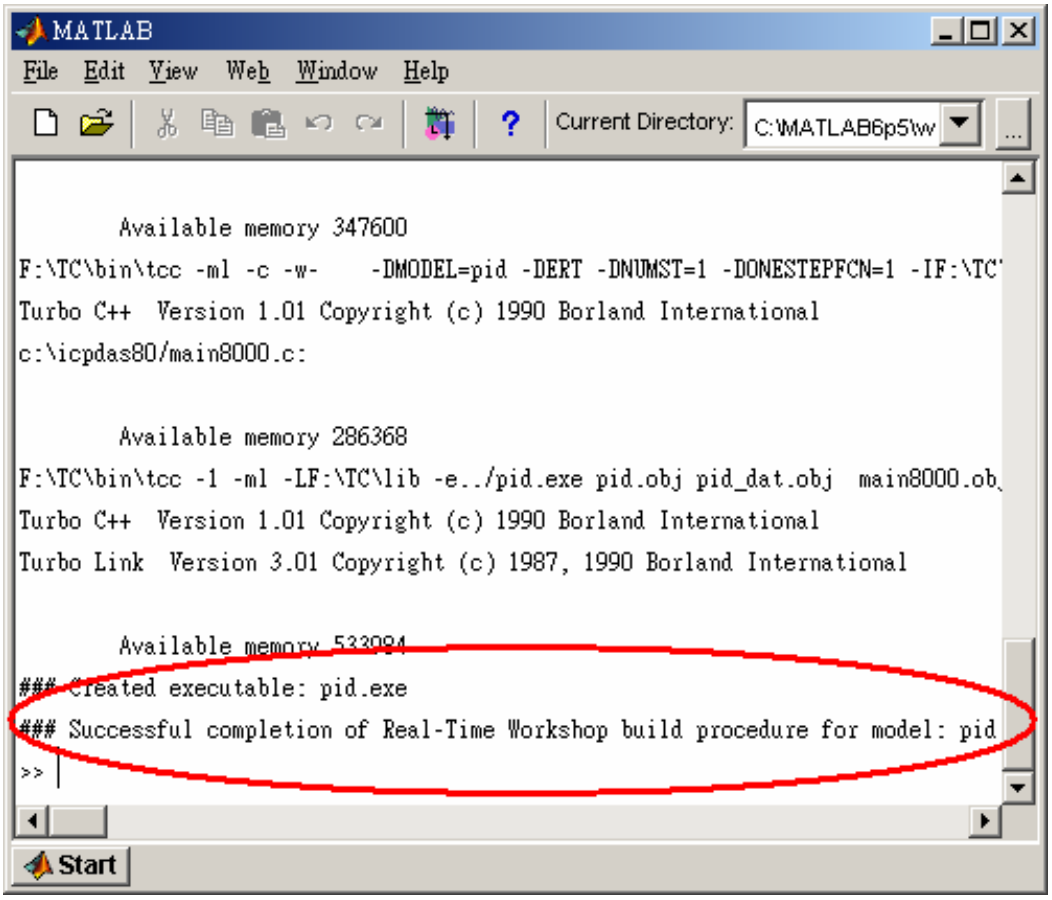
Step 5: And select the “ERT code generation options” (for MATLAB 6.1) or “ERT code generation options (1)” (for MATLAB 6.5) in the Category field. Then check the **Terminate function required** and **Single output/update function** options on the pane.



Step 6: For MATLAB 6.5, you have to select “ERT code generation options (3)” from the Category field. Then cancel the option **Generate an example main program**.



Step 7: When the above steps are done, click the Build button to start the build process. After the process ends successfully, the message in the MATLAB command window looks like as the following figure.



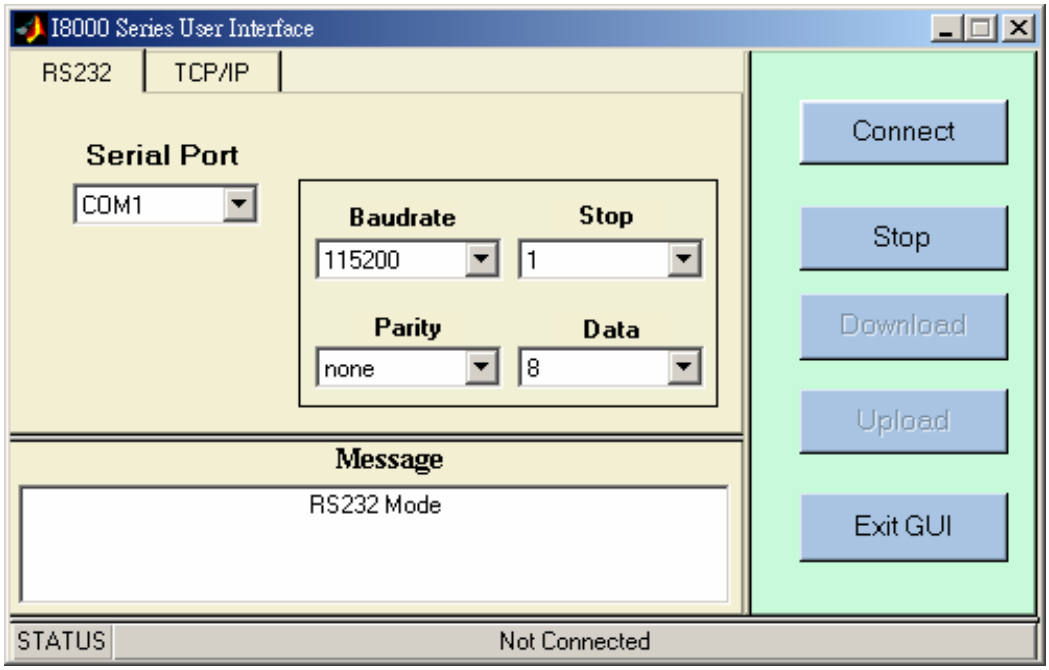
Note:

- **The name of the model cannot be over 4 characters. (This is due to the limitation of Turbo C/C++ Compiler.)**

3.4. Program downloading & data uploading

After the build process is completed, you can download the executable file generated to the I8xx8 embedded controller in the following way:

Enter **gui8000** at the MATLAB prompt, and the GUI dialog box appears. It provides two communication modes, RS232 and TCP/IP, for you to download application program.



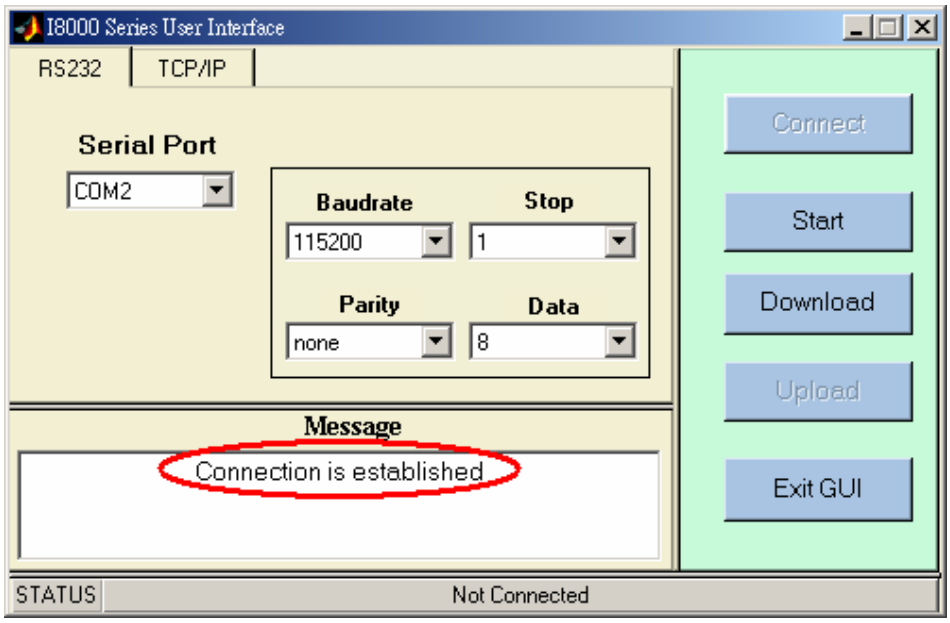
In RS232 mode, please follow steps to download the program to the I-8xx8 target system for application:

Step 1: Turn on the I8xx8 control system and **set I8xx8 control system in OS mode－it means that you must accomplish the step1 to step3 of Appendix A** before you download your program.

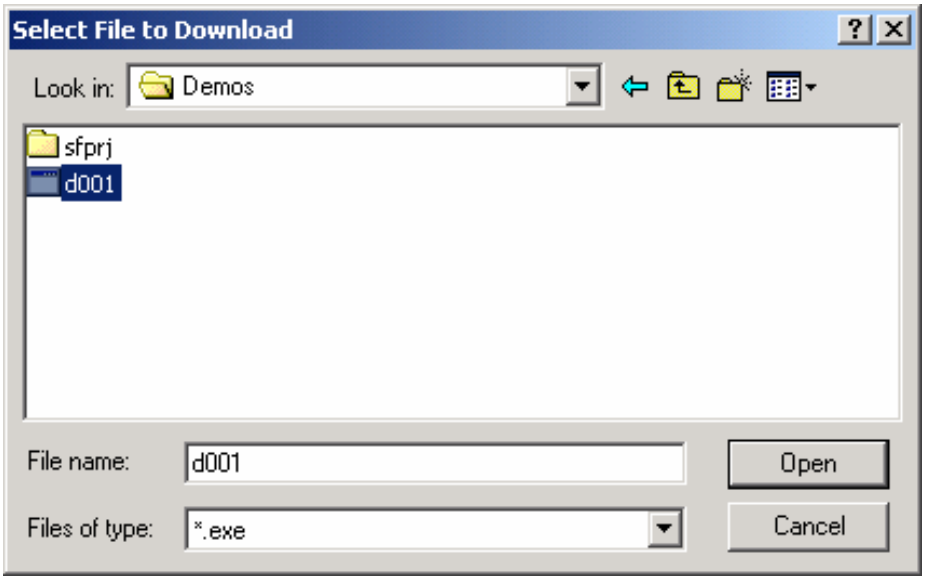
Step 2: Enter **gui8000** command at the MATLAB prompt and the GUI dialog box appears.

Step 3: Select the serial port you use in the PC to connect to the I8xx8 control system. Then set *Baud rate*, *Parity*, *Data bits*, *Stop bit* as '115200, none, 8, 1'. The default baud rate of the I8xx8 control system is 115200.

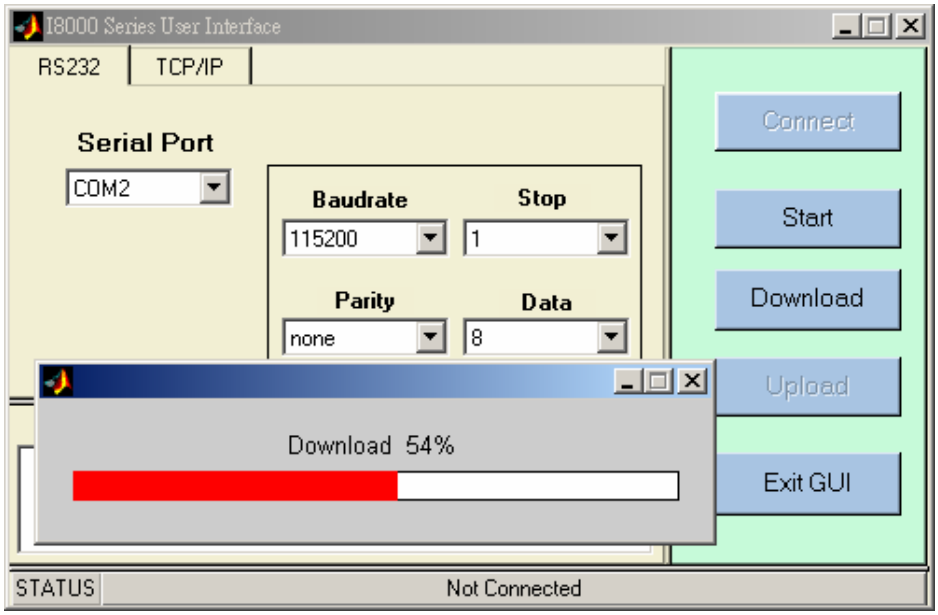
Step 4: **Close 7188xw.exe first** and then click the *Connect* button. If the connection is successful, the message, “Connection is established’ on the dialog box”, will show up.



Step 5: Now it’s time to download your program. Click the *Download* button and the *Select File to Download* dialog box appears. Select the file you want to download to the I8xx8 control system and press the OK button to close the dialog box and then the download process starts.



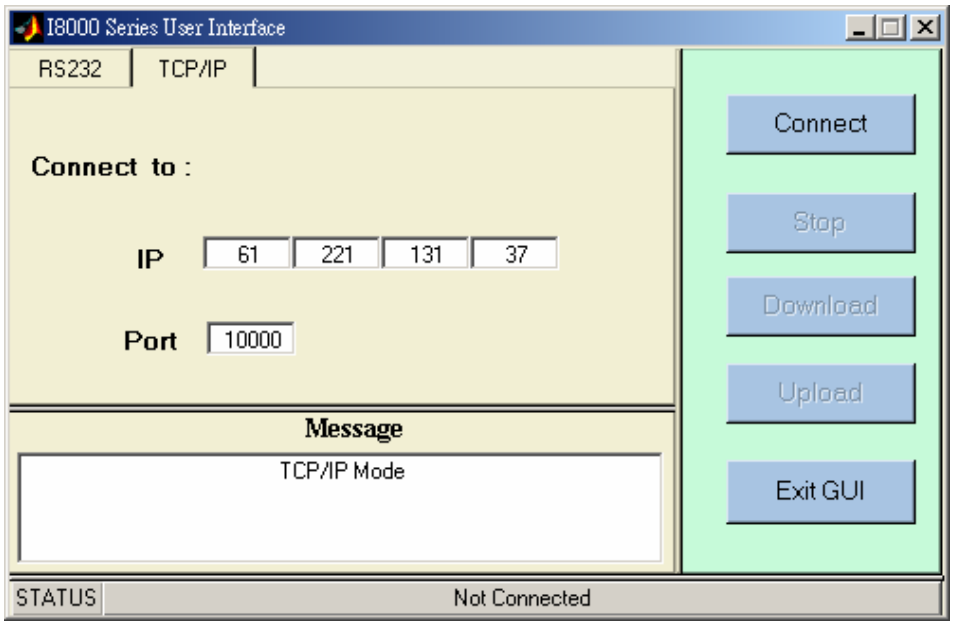
Step 6: On the dialog box that appears, you can see the progress of program downloading. Then you can click the *Start* button to run your control program after the progress of program downloading was finished.



Step 7: After you execute the program successfully, then you can click the *Upload* button to collect the data from the I8xx8 control system, and data will be saved in a file whose name is the filename you assigned in the DataToFile dialog box. This data file will be placed in the current working directory.

Step 8: Click the *Exit* button or ☒ on the right-upper corner to close the GUI dialog box.

To download the program in TCP/IP mode, click on the *TCP/IP* tab on the GUI dialog box. The panel on the dialog box changes as below figure.



In **TCP/IP mode,** you must specify IP and Port of the I-8x38 control system on the pane of the dialog box and **set I8xx8 control system in Firmware mode－it means that you can use the step8 of Appendix B to restart the firmware** before you download your program.

Then in a similar manner, connect to the I8xx8 control system, download the program and start it, and upload the data from the I8x38 control system. **Note: The default port of the I-8x38 control system is 10000. If you assign an incorrect value, then you will not be able to connect to it.**

3.5. Working with Stateflow

What is Stateflow?

Stateflow is a graphical design and development tool for control and supervisory logic. Using Stateflow you can:

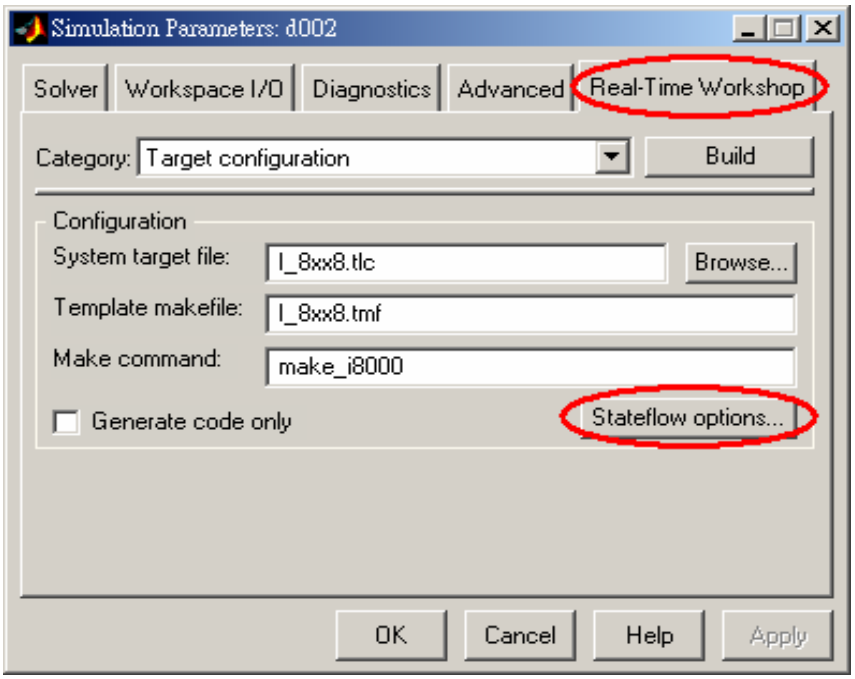
- Visually model and simulate complex reactive systems based on finite state machine theory.
- Design and develop deterministic, supervisory control systems.
- Easily modify your design, evaluate the results, and verify the system's behavior at any stage of your design.

To get more information about the Stateflow, please visit the website http://www.mathworks.com/products/stateflow.

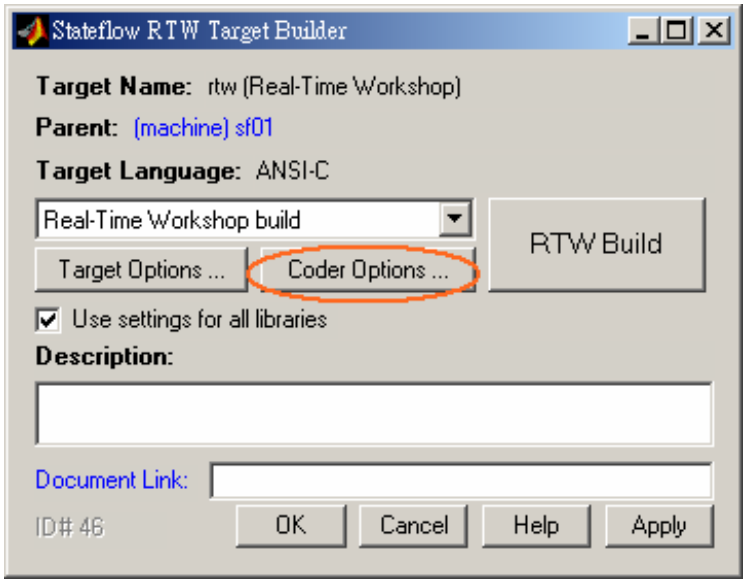
How to work with Stateflow

To convert your Simulink model containing the Stateflow charts to a target program, you have to configure the related options. In the first place, it is necessary to configure the RTW options as stated in section 3.4. Second, you need to adjust the Stateflow Coder options to the I-8xx8 control system. To configure the Stateflow Coder options, just do the steps as follows:

Step 1: Click *Simulation parameters* from the *Simulation* menu. On the *Simulation Parameters* dialog box that appears, click the *Real-Time Workshop* tab and then the pane changes.



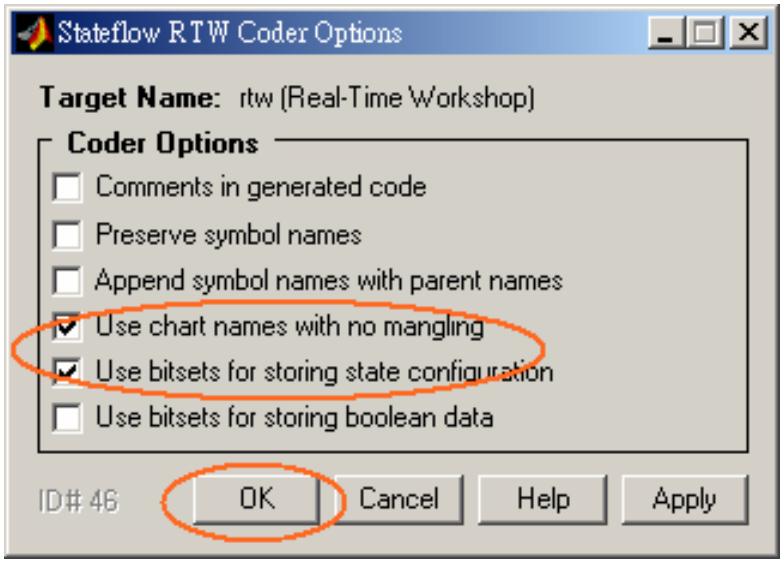
Step 2: Then click *Stateflow options* and the *Stateflow RTW Target Builder* dialog box appears as shown in the figure below.



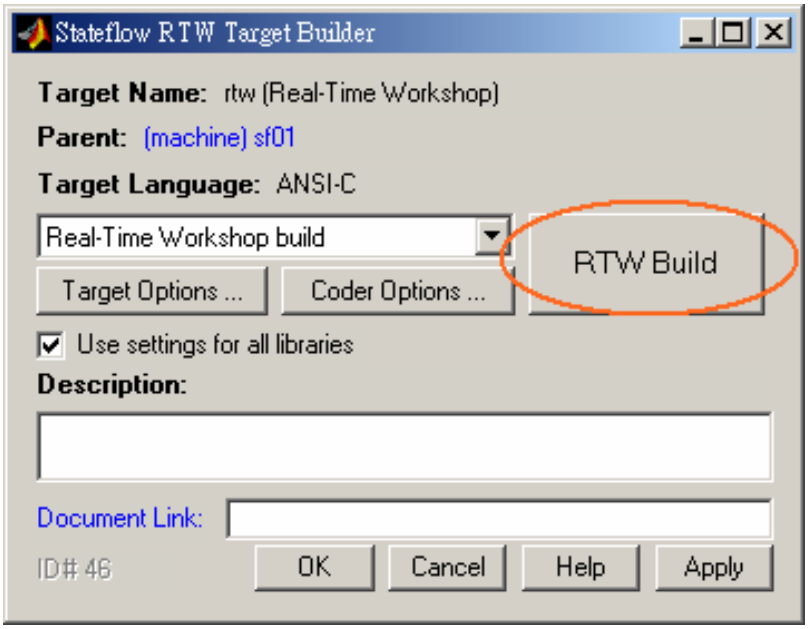
Step 3: Click *Coder Options…* to open the Stateflow RTW Coder Options dialog box. On the dialog box, check the two options:

Use chart names with no mangling
Use bitsets for storing state configuration

Then click OK to apply the setting you made and close the dialog box.



Step 4: Then click the *RTW Build* button on the *Stateflow RTW Target Builder* to start the build process. When the build process is completed successfully, an executable file is created in the current directory.


Stateflow RTW Target Builder
Target Name: rtw (Real-Time Workshop)
Parent: (machine) sf01
Target Language: ANSI-C
Real-Time Workshop build
RTW Build
Target Options ...
Coder Options ...
Use settings for all libraries
Description:
Document Link:
ID# 46
OK
Cancel
Help
Apply

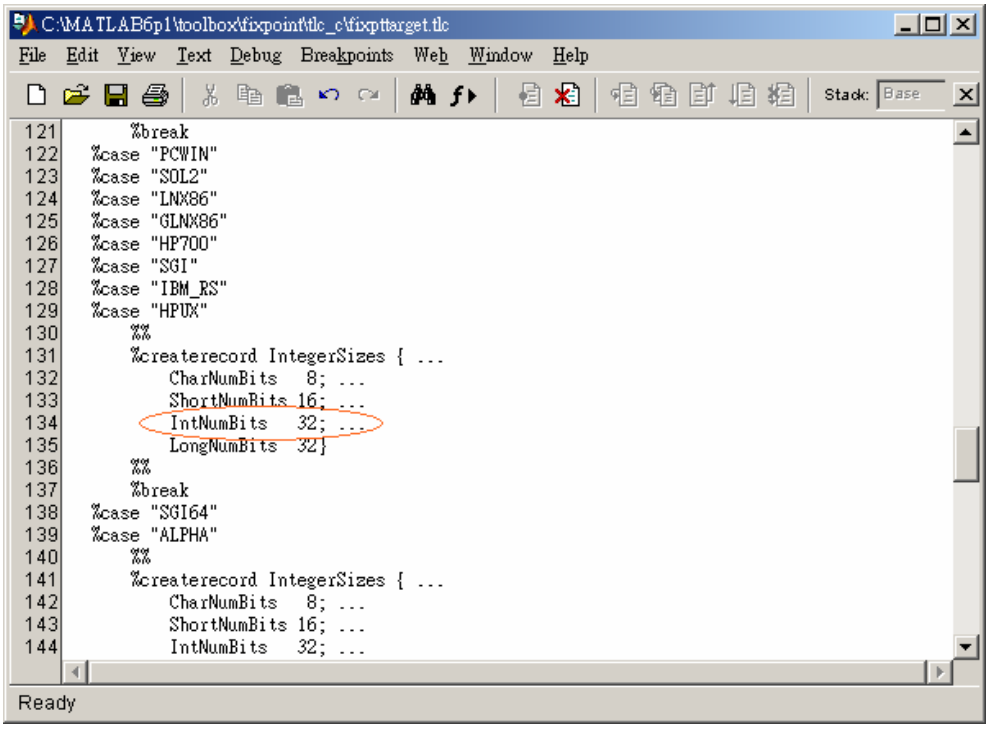
3.6. Working with Fixed- Point Blockset

What is Fixed-Point Blockset?

The Fixed-Point Blockset enables you to design control systems and digital filters that will be implemented using fixed-point arithmetic. With it, you can convert from floating-point to fixed-point modeling without changing any blocks. When combined with Real-Time Workshop, the blockset enables you to generate an efficient, integer-only C code representation of your model. When used with Real-Time Workshop Embedded Coder, the blockset lets you generate real-time C code for use on a fixed-point target. To get more information about the Fixed-Point blockset, please visit the website, http://www.mathworks.com/products/fixpoint.

How to work with Fixed-Point Blockset (Only for MATLAB 6.1)

As stated above, the Fixed-Point Blockset lets you generate real-time C code for use. However, for the I8xx8 control system, you need to modify the target language compiler file, fixedpttarget.tlc, which is located on the path matlabroot\toolbox\fixpoint\tlc_c\. Open the file and rewrite *IntNumBits 32* in the 134th row as *IntNumBits 16* as the figure below shown.



The reason is that the turbo C/C++ compiler or compatible defines the data type, int, as 2 bytes, equivalent to 16 bits. Nevertheless, it is defined as 4 bytes, equivalent to 32 bits, in the MATLAB environment. To let the build process complete successfully, you have to do as stated above.

4. Demos

In the following sections, we will show you the usage of the I-8000 driver blocks by a series of demos, including DI, DO, AI, AO, Relay, and Encoder.

4.1. DI & DO Modules

Introduction

Digital I/O interfaces are frequently used for the control system. In this section, we will guide users how to use the DI and DO driver block by an I-8053 module and an I-8057 module.

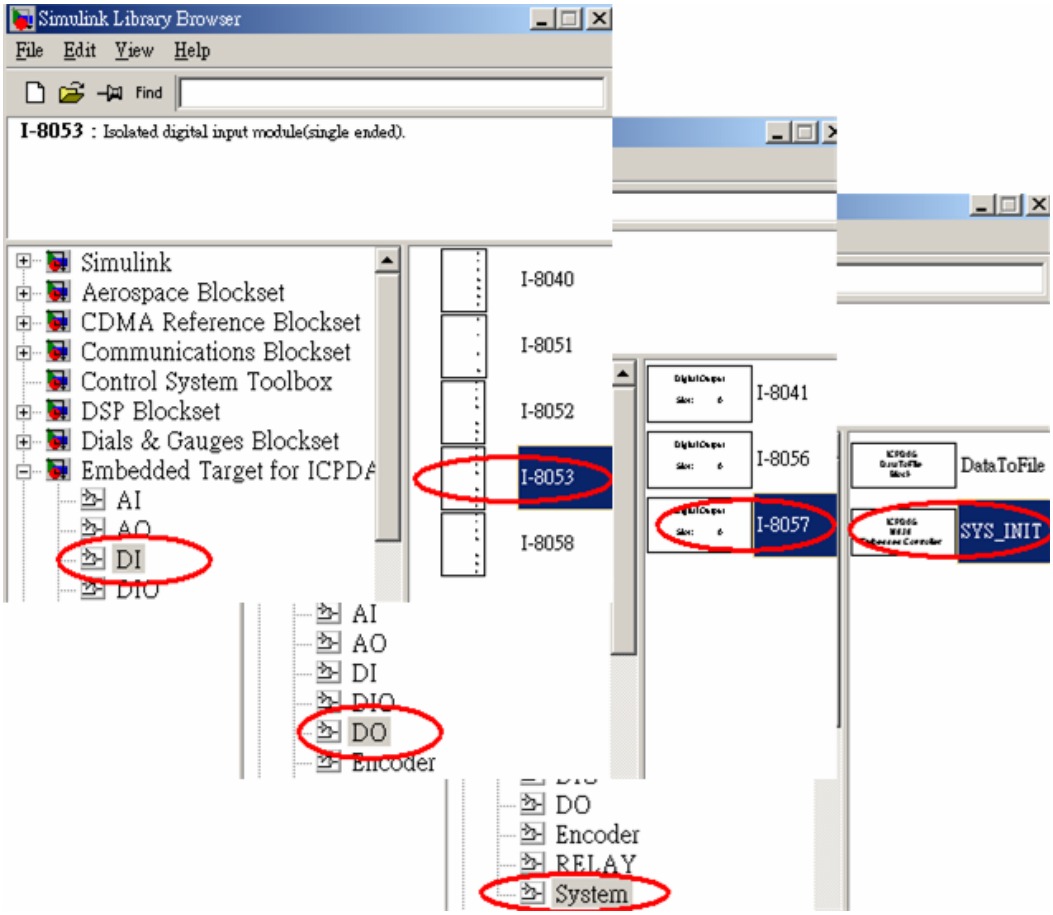
The I-8057 driver block expects a value smaller than 65535 from its input port. This is because I-8057 module is a 16-channel digital input module. If you assign a value 5 to the I-8057 module, then channel 0 and channel 2 of the I-8057 module will be set ON.

The I-8053 driver block will output a value 1 from the individual port if the corresponding channel of the module acquires a digital input. Otherwise, it will output a value 0.

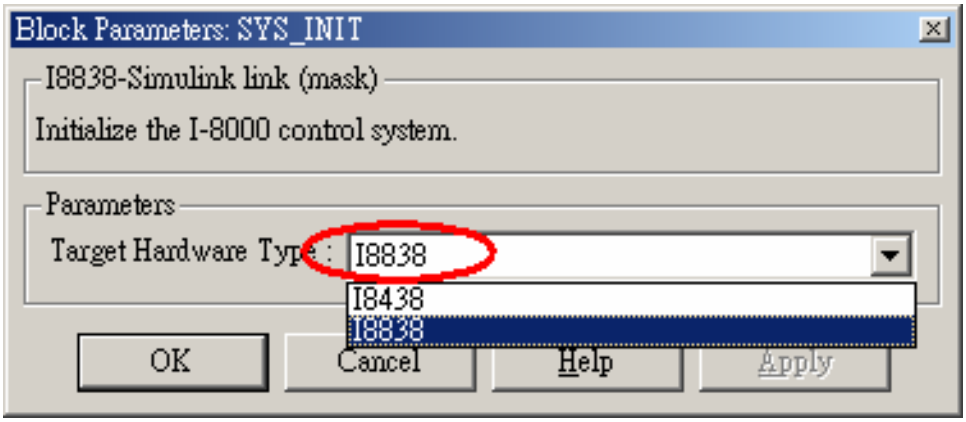
Demonstration

This demonstration uses 2 DO channels of I-8057 module and 2 DI channels of I-8053 module to test the digital input and digital output function. The digital output channels are connected to digital input channels. The following steps describe how to create and implement the experiment of digital input/output modules.

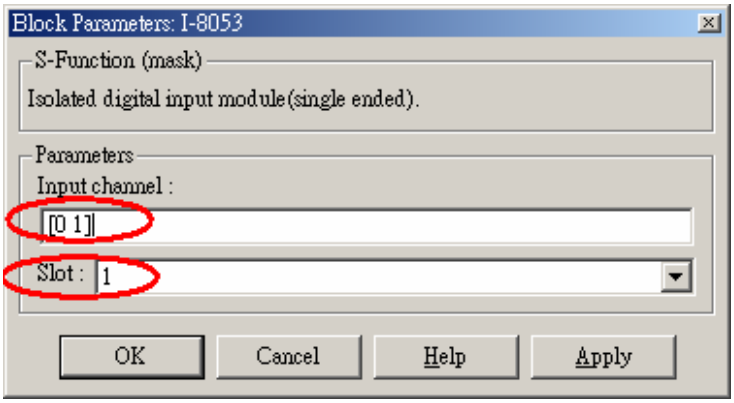
Step 1: Create a new model window and copy SYS_INIT, I-8053, and I-8057 blocks from the System, DI, and DO library to the model window respectively.



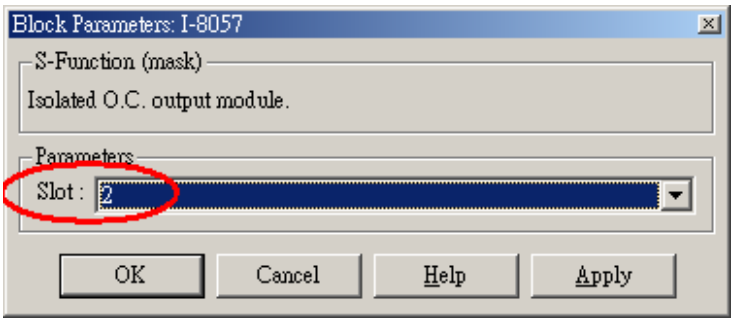
Step 2: Double-click on the SYS_INIT block to open the dialog box. On the dialog box that appears, select the correct target hardware type from the field, Target Hardware Type. In this demo, the target hardware is selected as I-8838.



Step 3: Double-click on the I-8053 block, and then the dialog box appears. To use the first two channels of I-8053 module, enter [0 1] in the field “Input channel”. And select the slot where the I-8053 module was mounted from the drop-down list.

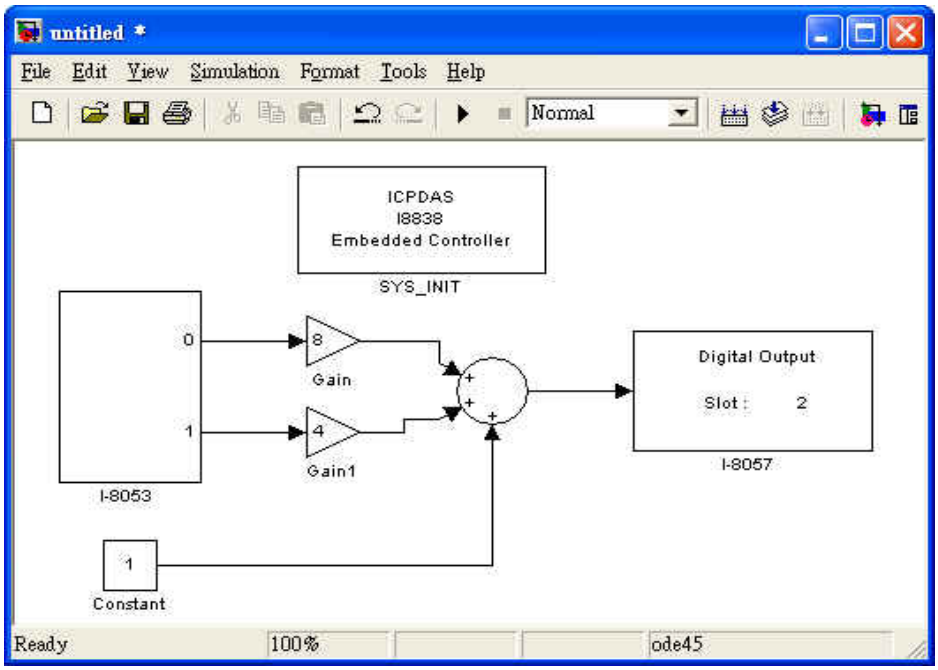


Step 4: In a similar manner, open the I-8057 dialog box and select the slot where I-8057 module is mounted.



Step 5: Copy two *Gain* and one *Sum* blocks to the model window from the *Simulink\Math* (for MATLAB 6.1) or *Simulink\Math Operations* (for MATLAB 6.5) library.

Step 6: Connect all the blocks as shown in the following figure.



Step 7: Click *Simulation parameters* from the *Simulation* menu on the model window. On the dialog box that appears, configure the RTW options (refer to section 3.3). And then click *Build* button to start the build process.

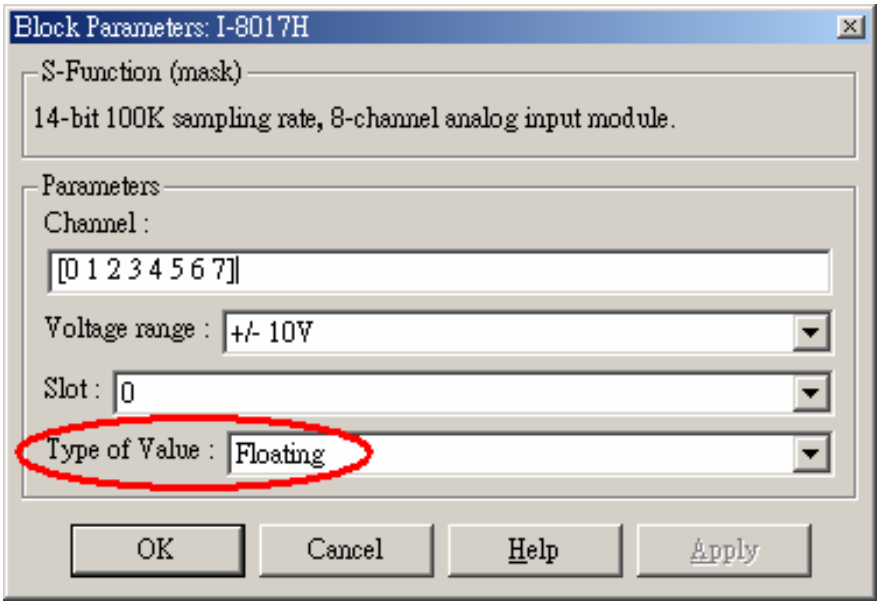
Step 8: When the build process ends successfully, download the control program generated to the I-8x38 and start it.

4.2. AI Modules

In this section, we will describe how to use the AI driver block, and a demo model will be presented.

Floating or Hex Input Type

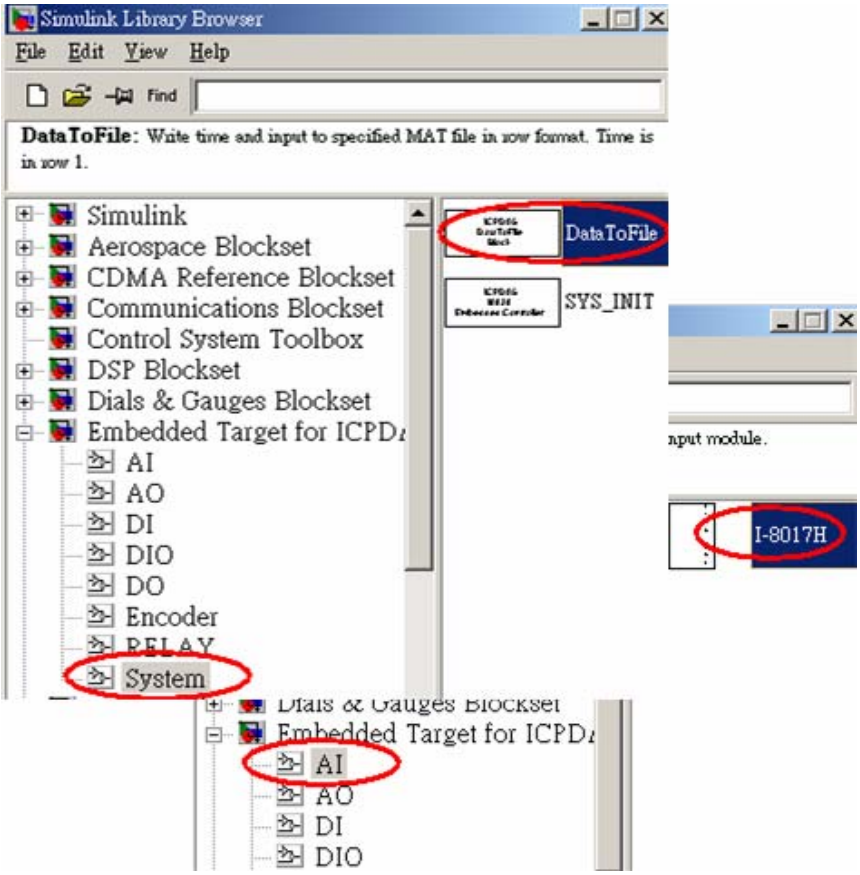
The AI driver block supports two types of input values, *floating* and *hex*. When the input type is set to *floating*, the output port will output a value, which is the same with the voltage measured from the corresponding channel. If the input type is set to *hex*, the output port will output a value between *-8192* and *8191* (This value is affected by the *Voltage range*; ex. When the *voltage range* is set to +/-10V, the output port will output a value *-4096* while the corresponding channel acquires a *-5V* analog input.).



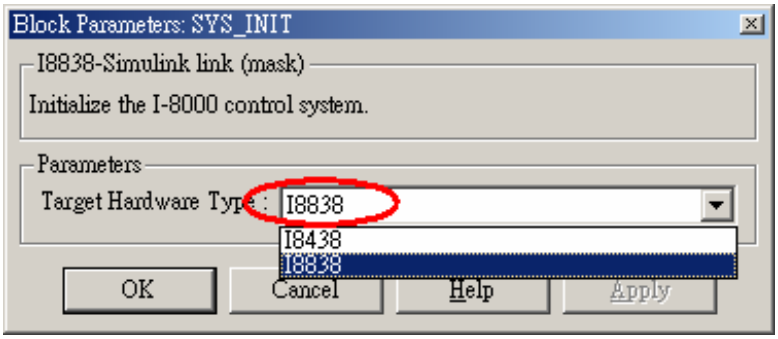
Demonstration

This demonstration uses 1 *I-8017H* and 1 *DataToFile* driver block to test the analog input function. By using the *DataToFile* driver block, the data acquired from the I-8017H module can be uploaded to PC for analysis. The following steps describe how to create and implement the experiment.

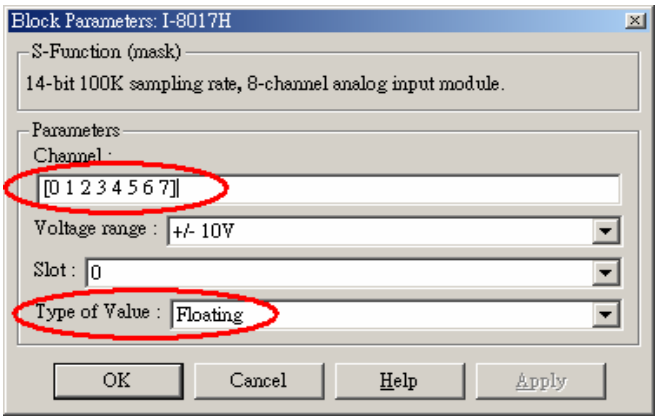
Step 1: Create a new model window and copy *SYS_INIT*, *I-8017H*, and *DataToFile* blocks from the *System* and *AI* library to the model window respectively.



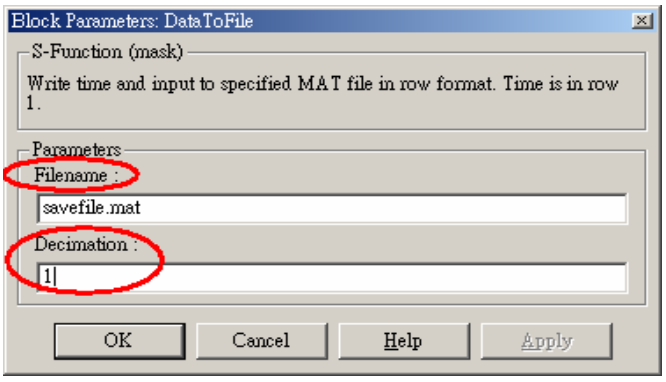
Step 2: Double-click on the SYS_INIT block to open the dialog box. On the dialog box that appears, select the correct type of target hardware from the field, “Target Hardware Type”. In this demo, the target hardware is chosen as I-8838.



Step 3: Double-click on the I-8017H block, and then the dialog box appears. In this demo, we use all the channels, so enter [0 1 2 3 4 5 6 7] in the “Channel” field. Then set voltage range as +/-10V, and specify the slot where I-8017H module is mounted. Finally, set *Type of Value* to *Floating*.

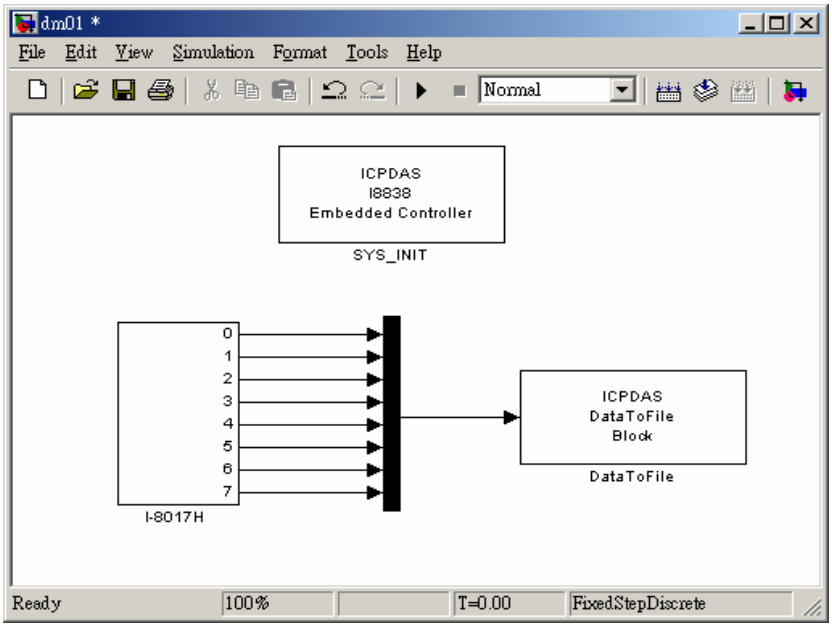


Step 4: In a similar manner, open the DataToFile dialog box and specify the filename in the "Filename" field. Enter 1 in the "Decimation" field, and then the analog input signal will be recorded in every sampling time interval.



Step 5: Copy one Mux block to the model window from the *Simulink\Signals & Systems* (for MATLAB 6.1) or *Simulink\Signal Routing* (for MATLAB 6.5) library.

Step 6: Connect all the blocks as shown in the figure below.

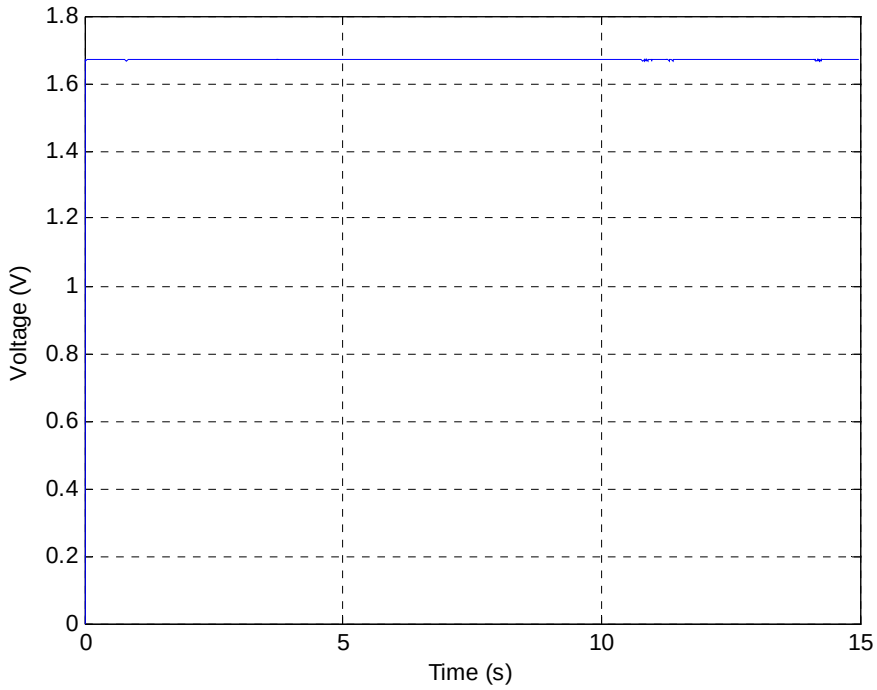


Step 7: Click *Simulation parameters* from the *Simulation* menu on the model window. When the dialog box appears, configure the RTW options (refer to section 3.3). And then click *Build* button to start the build process.

Step 8: When the build process ends successfully, download the control program generated to the I-8x38 and upload the experiment data for further analysis (refer to section 3.4).

Step 9: If the stop time that you specified is up, you can start the data uploading. After the uploading process is completed, use the built-in MATLAB scripts to plot the result. It would look like the figure below.
For Example:

plot (tcpdata(1, :), tcpdata(2, :))

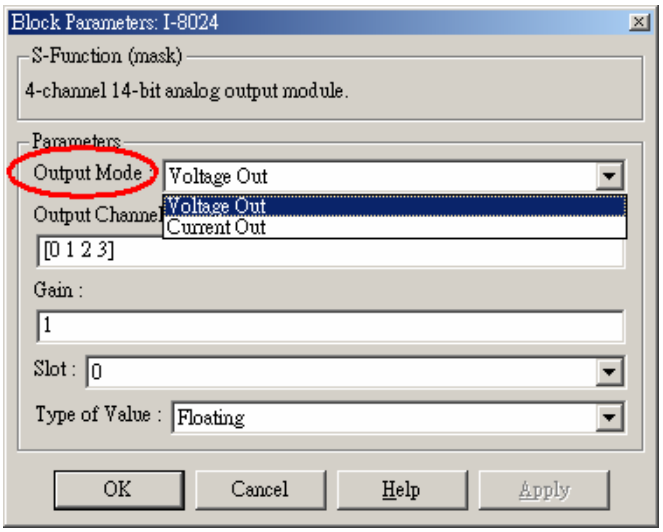


4.3. AO Modules

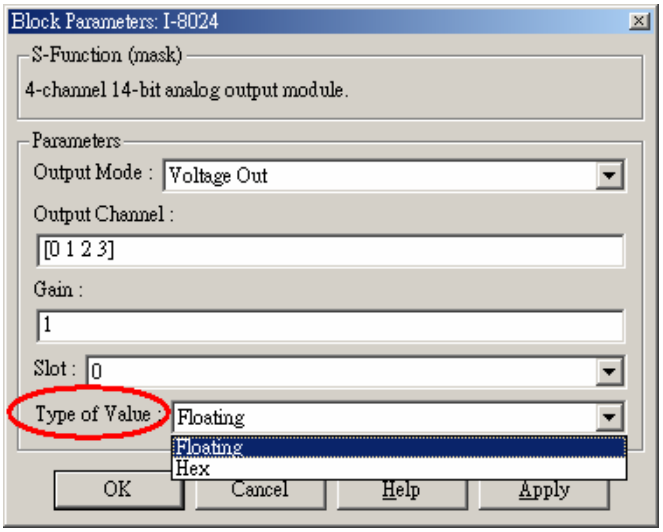
This section attempts to make you familiar with the usage of the I-8024 module, which is a 4-channel and 14-bit analog output module. In the beginning of this section, we will give you a detailed introduction of the I-8024 driver block. And then a demonstration will be presented in the rest of this section.

Introduction to I-8024 driver block

On the dialog box of the I-8024 driver block, you can select the outputs of the I-8024 module as either “Voltage Out” or “Current Out”. If you select “Voltage Out” in the “Output Mode” field, an output range between +10V and –10V will be available; otherwise, a current between 0 and 20 mA will be instead.



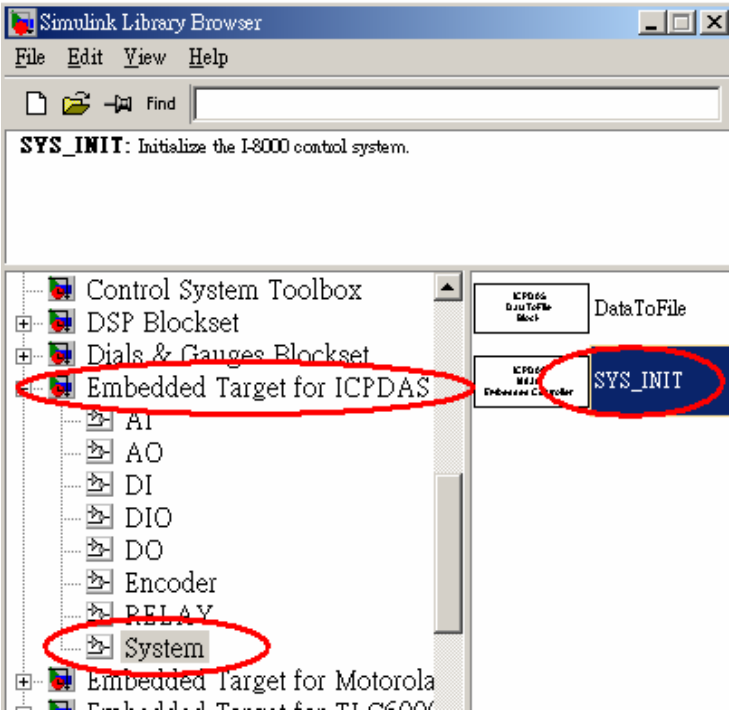
Furthermore, the input type of the I-8024 driver block can be either “Floating” or “Hex”.



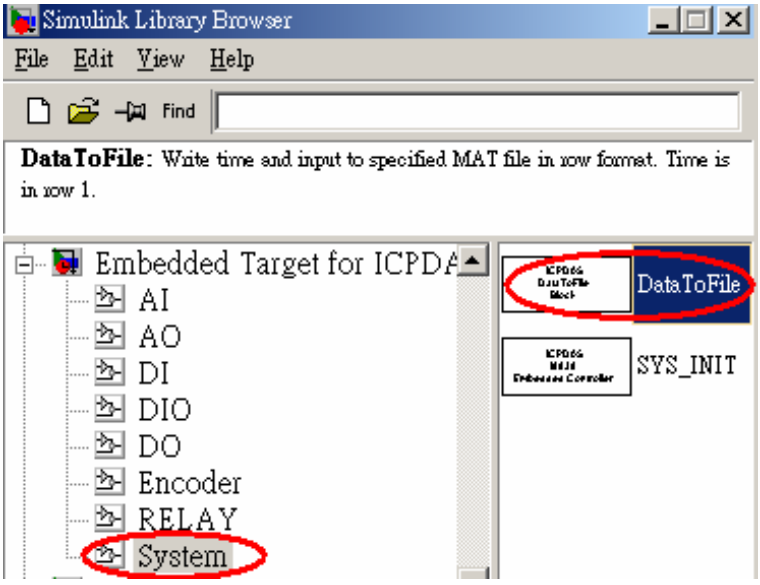
Demonstration

In this demo, we use 1 channel of the I-8024 module and 1 channel of the I-8017H module. Then channel 0 of I-8024 module is physically connected to channel 0 of I-8017H module. After the physical wire connection is done, send a sine wave to the input port of the I-8024 driver block. The following figures describe the steps by which this demo is created and implemented.

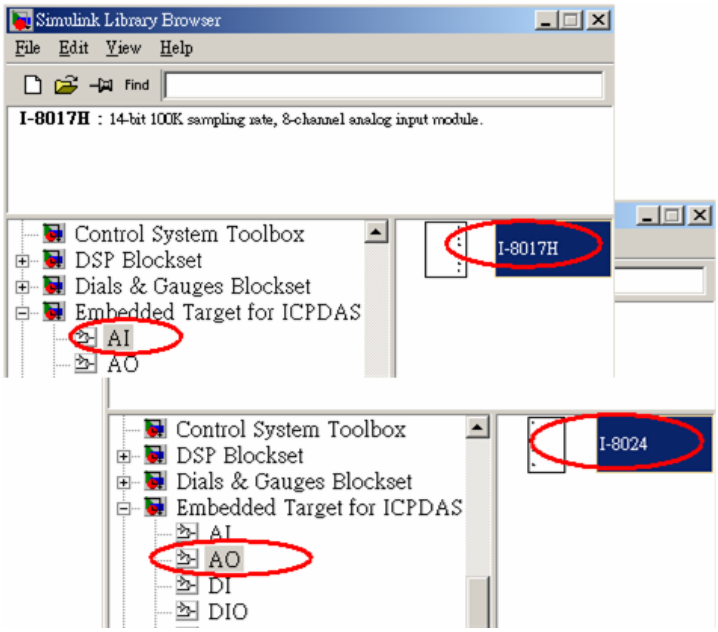
Step 1: Create a new model in Simulink and insert a SYS_INIT block from the System block library.



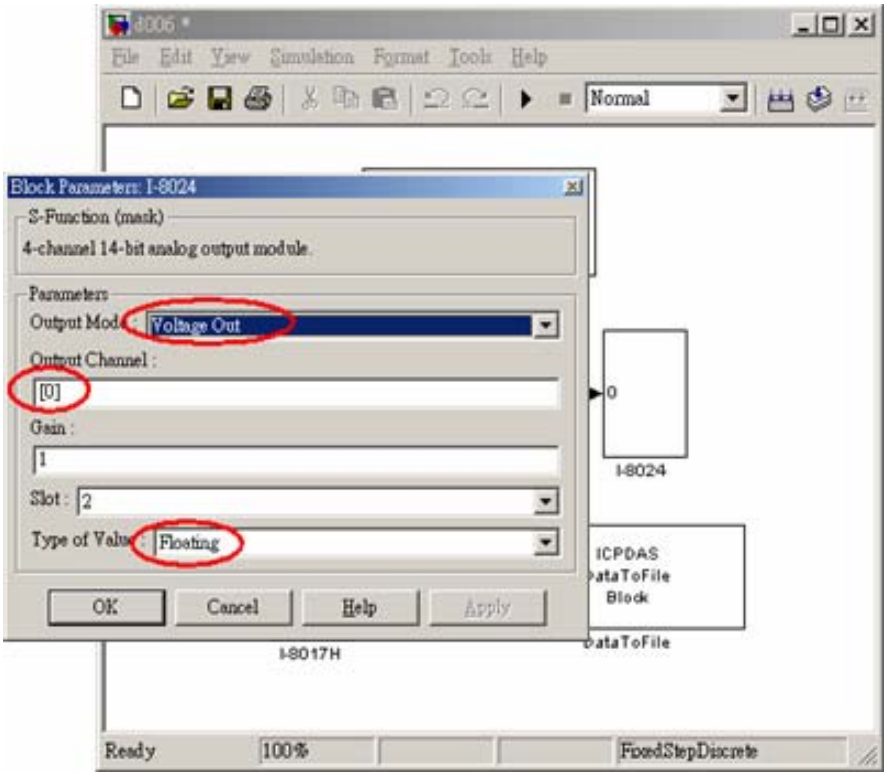
Step 2: Copy the DataToFile block from the System block library to the model.



Step 3: In a similar manner, insert an I-8024 block and an I-8017H block separately from the AO and AI block library.

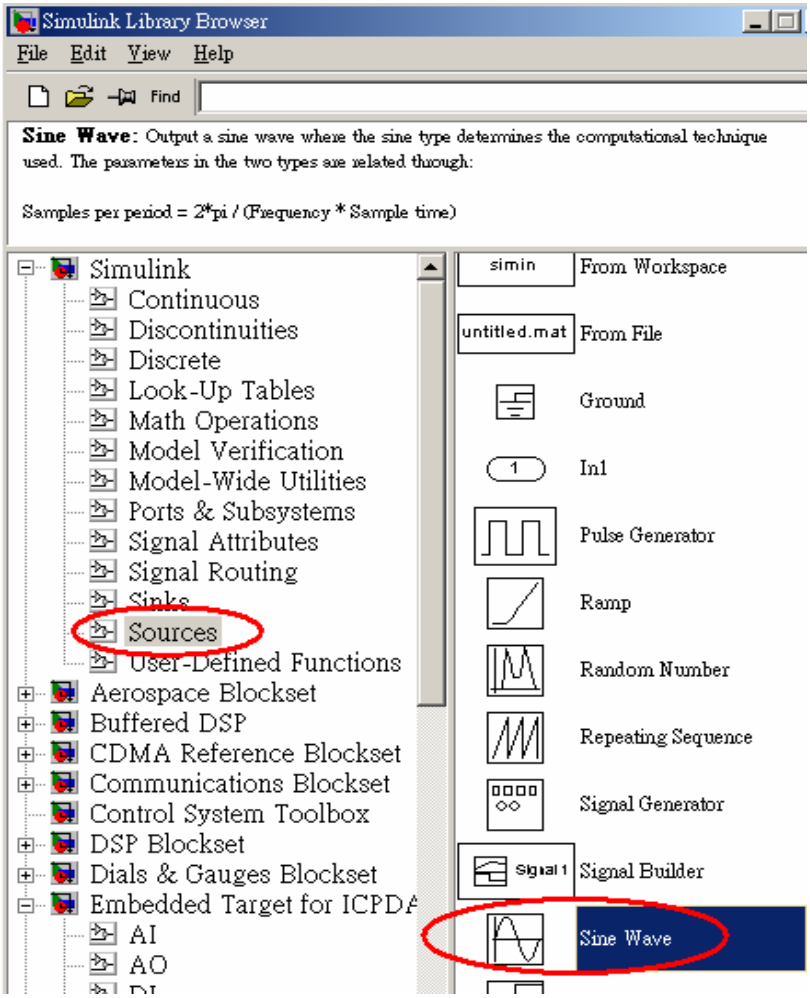


Step 4: Double click on the I-8024 block to setup the AO module. Here we use channel 0 of the I-8024 module, which is mounted on slot 2 of the I-8xx8 embedded controller. And the output mode is set to “Voltage Out”; the type of value is selected as “Floating”.

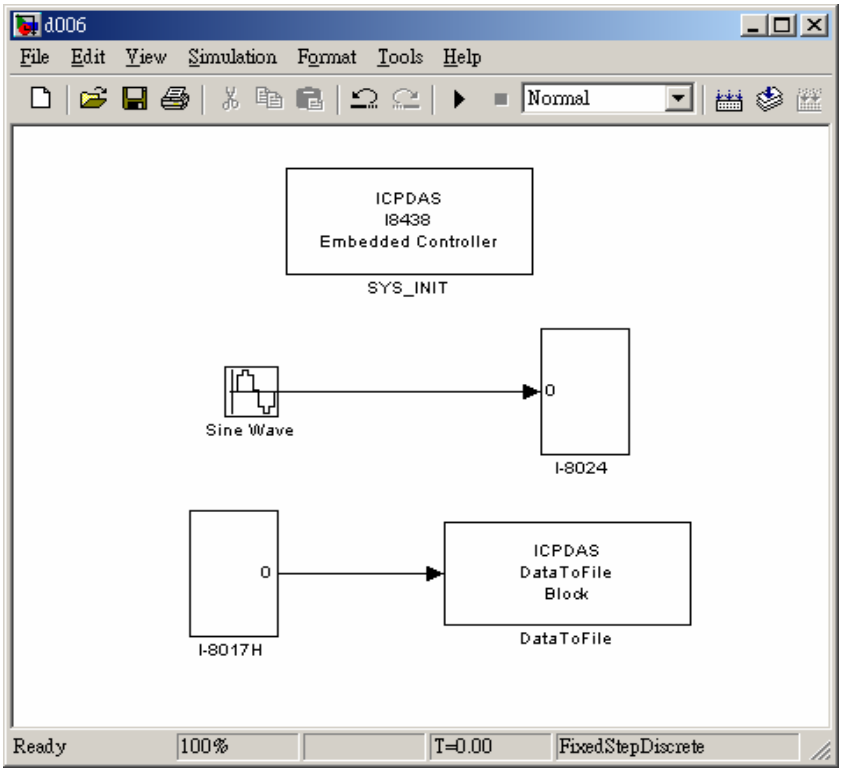


Step 5: As to the setting of SYS_INIT and I-8017H, please refer to the section 4.1 and 4.2.

Step 6: Add a Sin Wave block from the Simulink\Sources library.



Step 7: Connect all blocks as shown in the figure below.

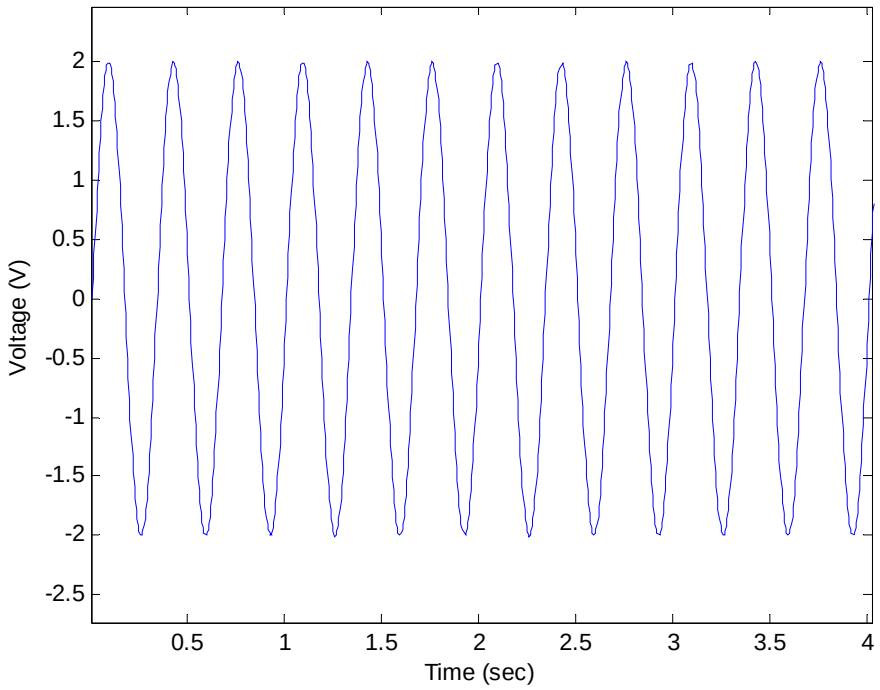


Step 8: Click Simulation parameters from the Simulation menu to open the Simulation Parameters dialog box. Then configure the RTW options (refer to section 3.3) and press the Build button to start the build process.

Step 9: When the build process ends successfully, download the .exe file generated to the I-8x38 and run (refer to section 3.4).

Step 10: If the stop time that you specified is up, you can start the data uploading. After the uploading process is completed, use the built-in MATLAB scripts to plot the result. It would look like the figure below.
For Example:

plot (tcpdata(1, :), tcpdata(2, :))

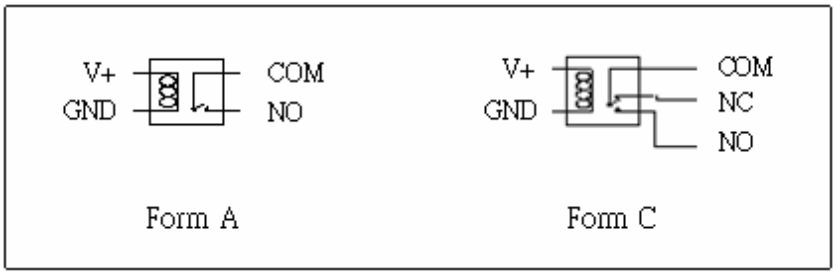


4.4. Relay Modules

In the area of automatic control, the relay component is frequently used as a voltage-activated switch, consisting of a bi-directional switched connections and a switching input. The I-8000 series module MATLAB Driver also provides the relay driver block to facilitate your design. In the following subsection we demonstrate the usage of this block.

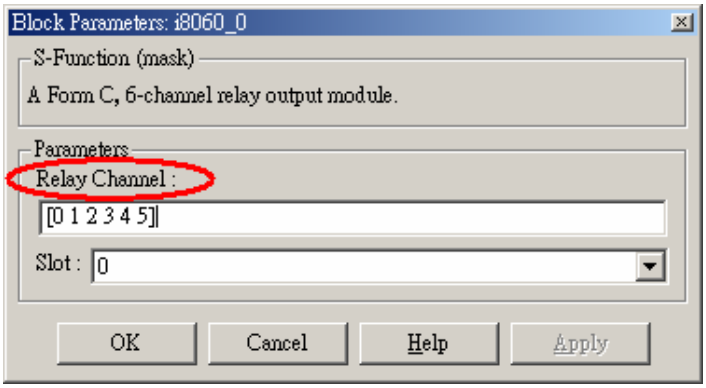
Introduction to Relay driver block

Basically, the relay modules provided by ICP DAS are categorized into two forms, form A and form C. The figure below shows the difference between them.



The relay of form A is represented by the I-8064, and the relay of form C is represented by the I-8060. No matter which form a relay is, however, the usage of the driver block is the same. The individual input port of the driver block expects a value greater than 0 to activate the corresponding physical relay output. In other words, if you assign a value 1 to the channel of the relay driver block, the connection of the relay will be switched to NO.

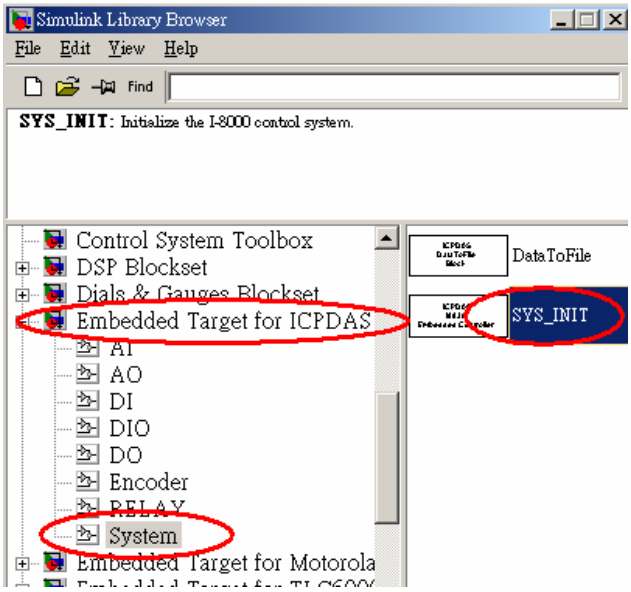
On the dialog box of the relay driver block shown in the figure below, you can select the channels, which are used by entering a row vector in the field of “Relay Channel”. And remember to choose the correct slot where the relay module is mounted.



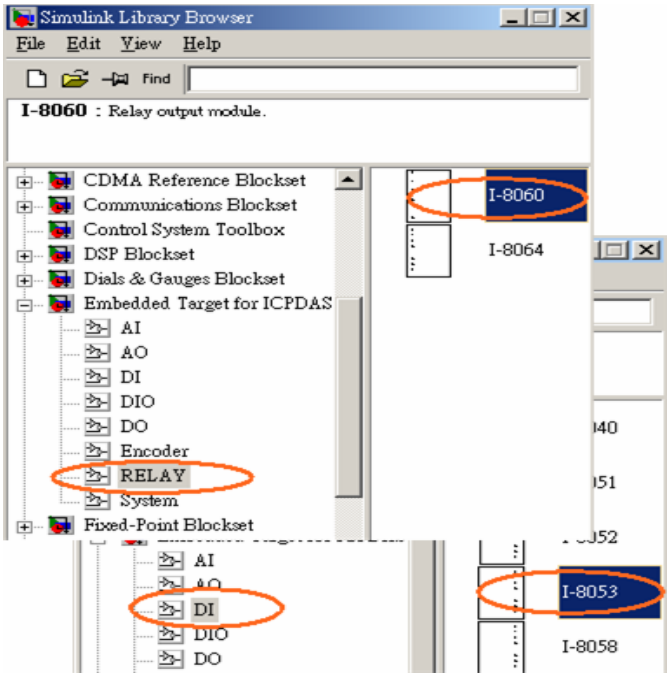
Demonstration

This demo uses an I-8060 relay module and an I-8053 DI module to setup the experiment. Channel 0 of the I-8053 module is connected to a limit switch. When channel 0 of the I-8053 module is activated, it will cause the channel 0 of the I-8060 module switch to NO. The following steps describe how this demo is created and implemented.

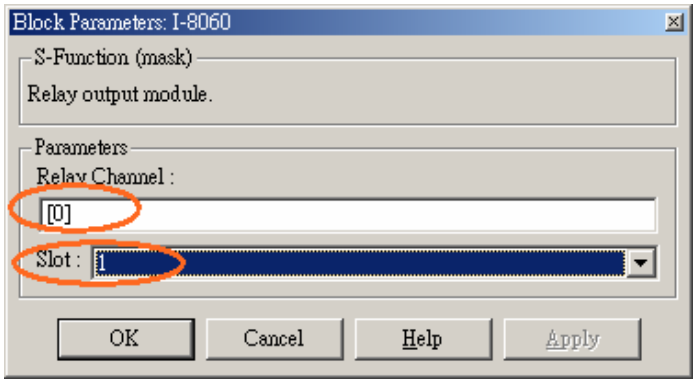
Step 1: Create a new model in Simulink and insert a SYS_INIT block from the System block library.



Step 2: Copy an I-8053 block and an I-8060 block separately from the DI and RELAY block library.

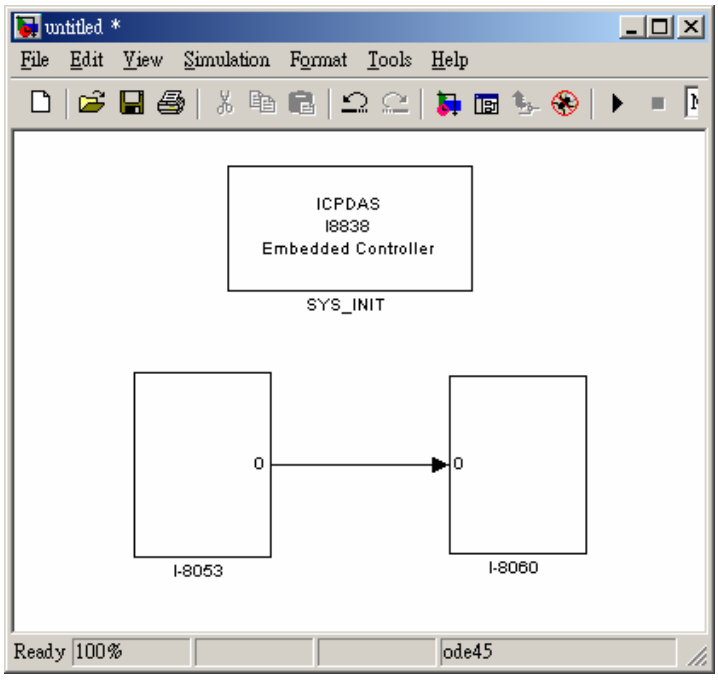


Step 3: Double click on the I-8060 block to setup the relay module. Here we use channel 0 of the I-8060 module, which is mounted on slot 1 of the I-8x38 embedded controller.



Step 4: Configure the SYS_INIT and the I-8053 block. (Please refer to the section 4.1.)

Step 5: Connect all blocks as shown in the figure below.



Step 6: Click Simulation parameters from the Simulation menu to open the Simulation Parameters dialog box. Then configure the RTW options (refer to section 3.3) and press the Build button to start the build process.

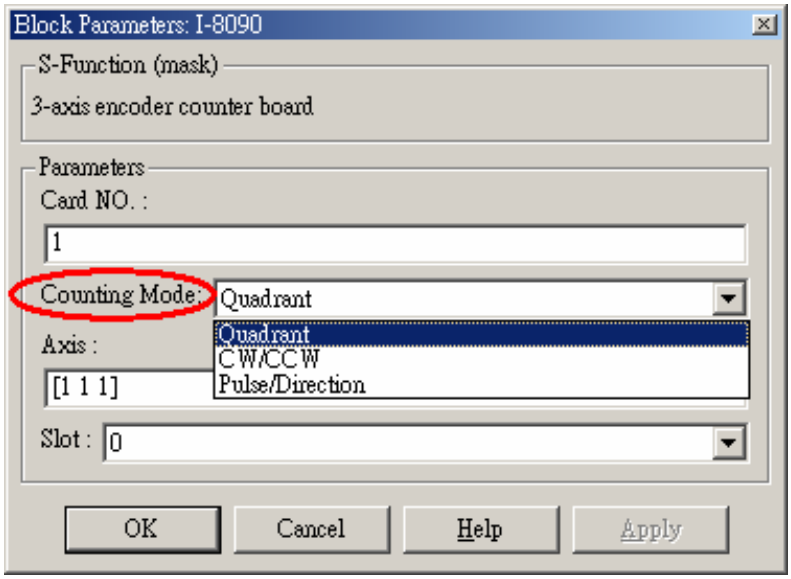
Step 7: When the build process ends successfully, download the .exe file generated to the I-8x38 and run it (refer to section 3.4).

4.5. Encoder Module

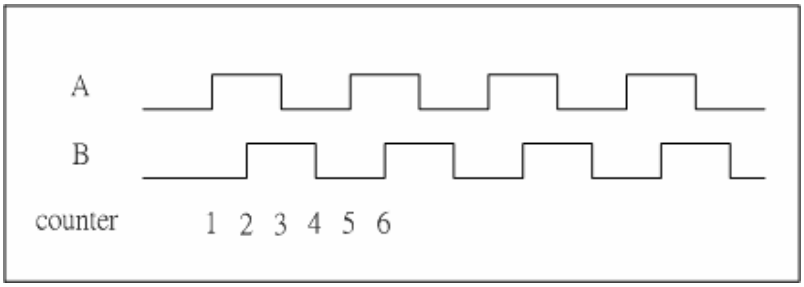
In most applications of motion control, encoder input modules are used to accept signals from encoders on the motors. Therefore, the I-8000 series module MATAB Driver also provides the driver block for I-8090, which is a 3-axis encoder counter board on I-8000 platform. By using the I-8000 series module MATLAB Driver, you can design your motion controller easily and quickly. In the following paragraphs, we will show you the usage of the I-8090 driver block.

Introduction of I-8090 driver block

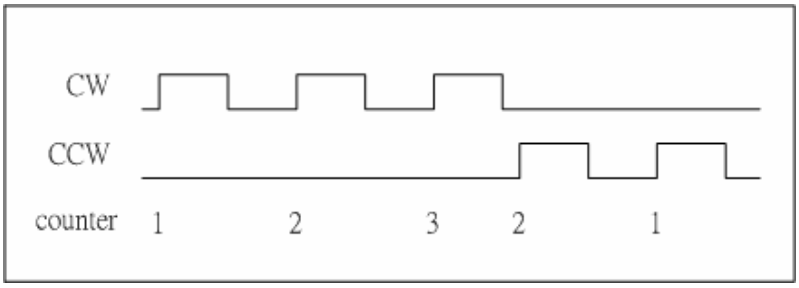
On the dialog box of I-8090 driver block, you can select the counting mode of the I-8090 among *Quadrant*, *CW/CCW*, and *pulse/direction*. The setting in this field must match with your wire connections. Otherwise, the measurement of data will not be correct. To get more information about wire connections, please refer to the I-8090/8091 user's manual.



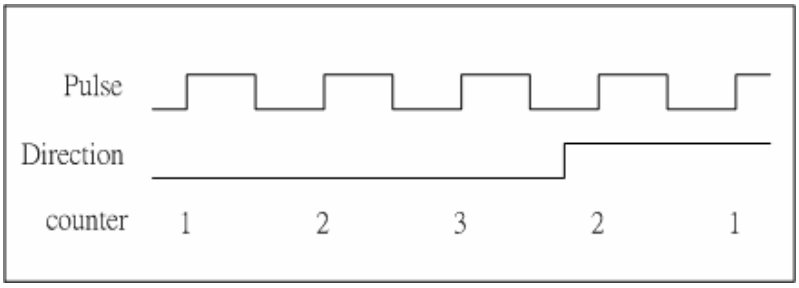
The difference among quadrant, CW/CCW, and pulse/direction counting modes is shown in the following figures.



Quadrant counting mode

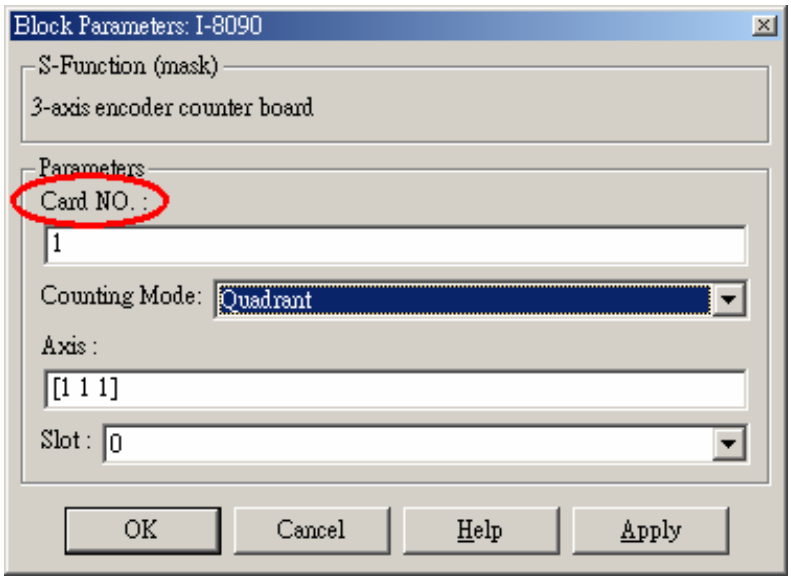


CW/CCW counting mode



Pulse/Direction counting mode

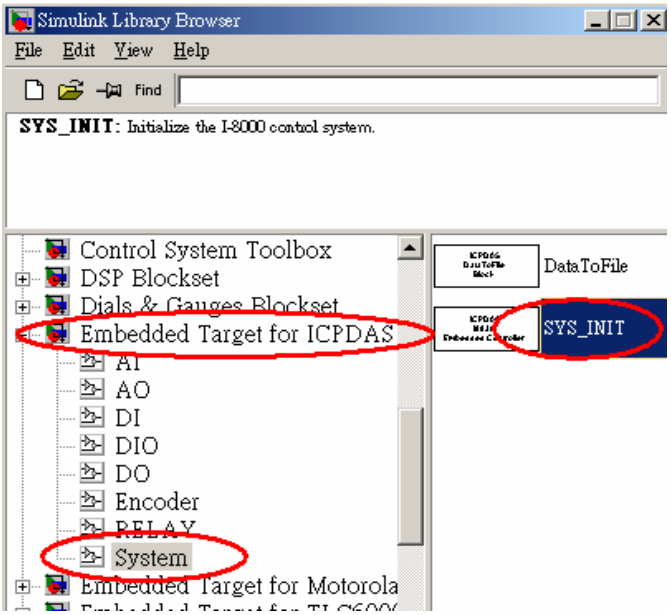
Besides, you must specify a value smaller than 19 in the “Card NO” field on the dialog of I-8090 driver block. This value is used to identify different I-8090 encoder cards, and hence the same number cannot be assigned to different I-8090 modules.



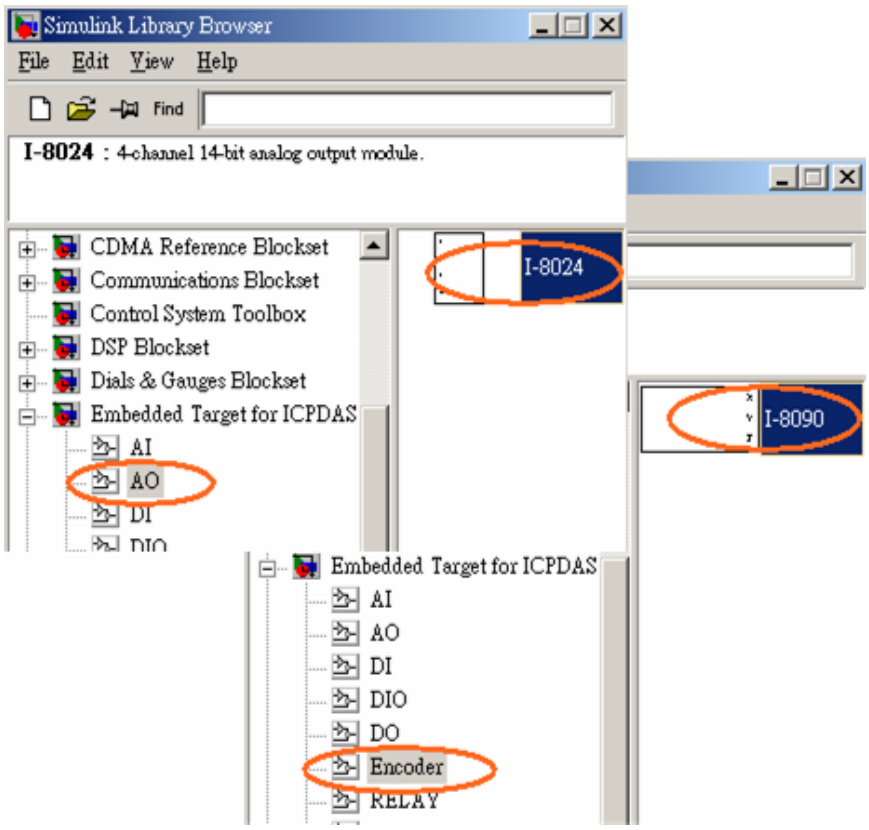
Demonstration

This demo uses an I-8024 AO module to drive the motor and an I-8090 encoder input module to accept the signals from the encoder of the motor. And we also add a DataToFile block to the model for data recording. Here, we will show you the procedure step by step.

Step 1: Create a new model in Simulink and insert a SYS_INIT block from the System block library.

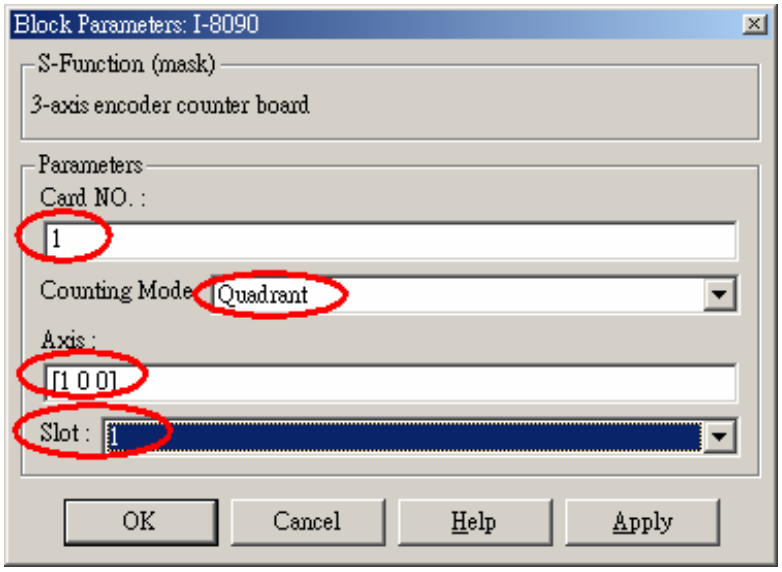


Step 2: Insert an I-8024 block and an I-8090 block separately from the AO and Encoder block library.



Step 3: Double click on the I-8090 block to setup the encoder module. Here we use X-axis of the I-8090 module, which is mounted on slot 1 of the

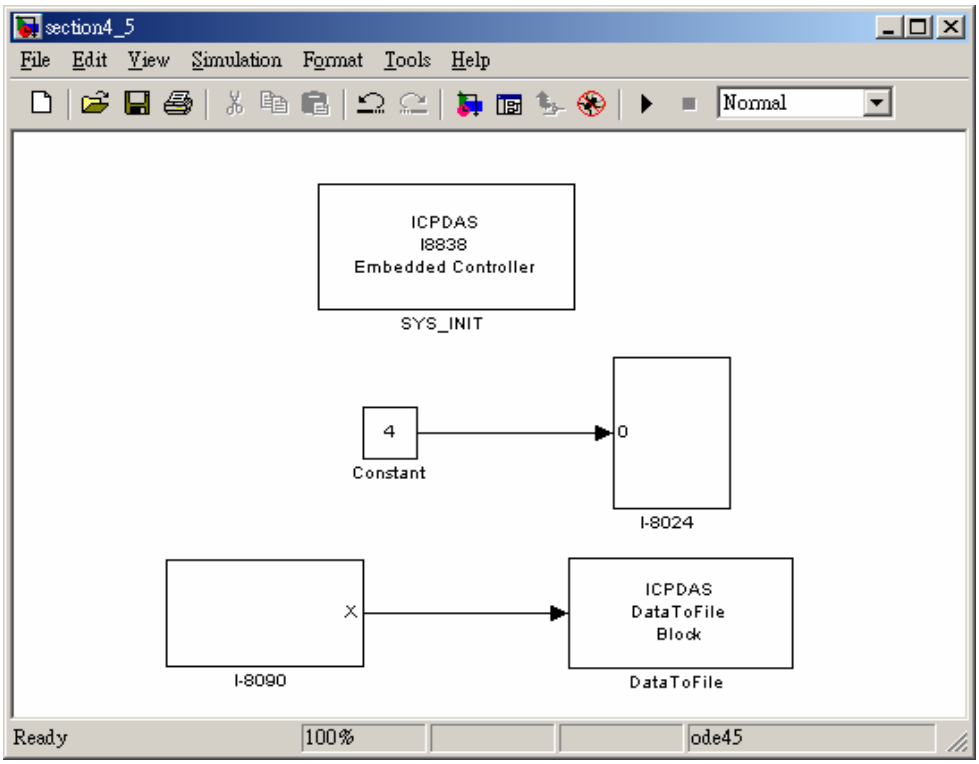
I-8xx8 embedded controller. And select the counting mode as "Quadrant" and assign "Card NO" to 1.



Step 4: Configure the SYS_INIT and the I-8024 block. (Please refer to the section 4.1 and 4.3.)

Step 5: Add a DataToFile block from the System block library, and use it to store the data of the encoder. At the same time, insert a Constant block to send the voltage command to the I-8024 block.

Step 6: Connect all blocks as shown in the following figure.



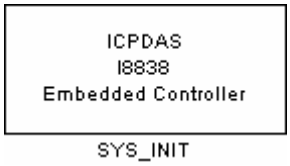
Step 7: Click Simulation parameters from the Simulation menu to open the Simulation Parameters dialog box. Then configure the RTW options (refer to section 3.3) and press the Build button to start the build process.

Step 8: When the build process ends successfully, download the .exe file generated to the I-8x38 and run it (refer to section 3.4).

5. Matlab Driver Block Reference

This section presents detailed descriptions and usage of all blocks in the MATLAB Driver block library.

SYS_INIT



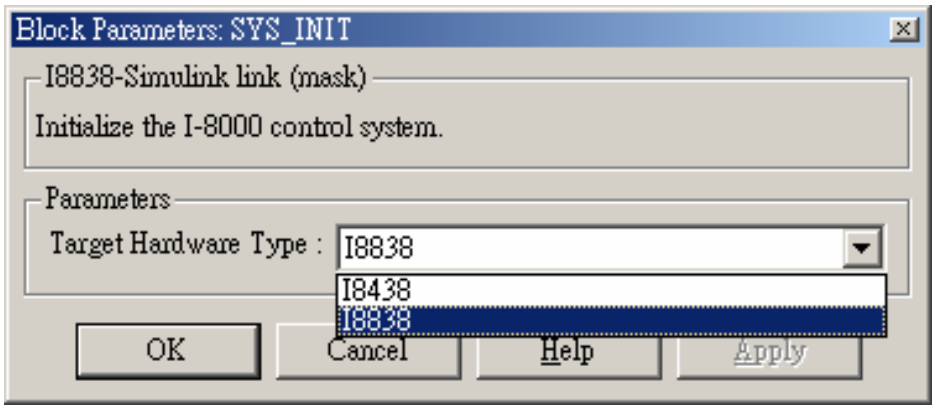
Description:

Initialize the I-8xx8 control system.

Library:

System

Dialog Box:

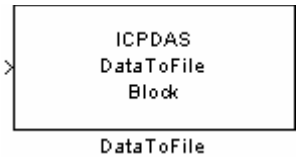


Driver Block Parameters:

Target Hardware Type - The type of the I-8xx8 control system that is used. The setting must match with the actual situation; otherwise, it might cause unexpected problems. The following table presents a list of the types available and the difference between them.

TYPE	Slots Available
I-8438	4
I-8838	8

DataToFile



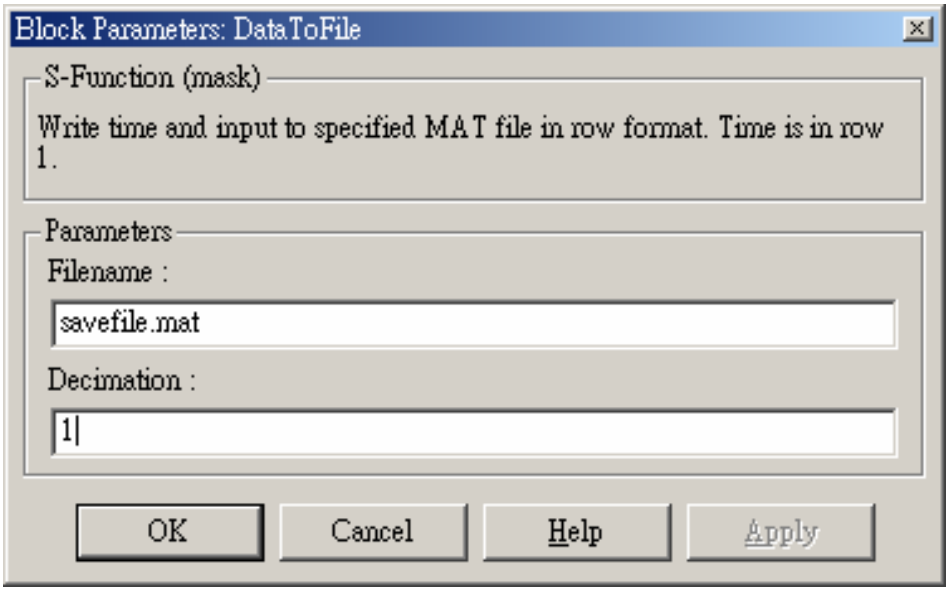
Description:

Write time and input to specified MAT file in row format. Time is in row 1.

Library:

System

Dialog Box:

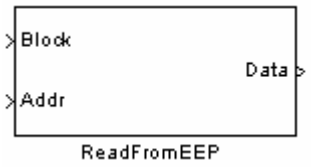


Driver Block Parameters:

Filename - The fully qualified filename of the MAT-file in order to store the data. The default filename is savefile.mat. The file will be stored in the current working directory.

Decimation - A decimation factor. The default value is 1. This parameter allows you to write data at every nth sample, where n is the decimation factor.

ReadFromEEP



Function:

Read data from EEPROM

Library:

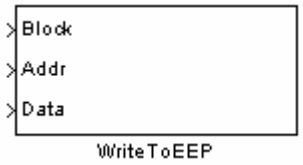
System

Description:

In this block, there provides no parameter for users to set. To use this block properly, what users have to do is to assign the intended value to input ports, Block and Addr. And the data read from eeprom will be exported to the "Data" port. More details of these ports are given as follows:

1. **Block –** This parameter specifies the number of the eeprom's block, from which you want to read. For I-8438/8838, its range is between 0 and 7.
2. **Addr –** A value between 0 and 255 is acceptable. In I-8438/8838, the eeprom is divided into 8 blocks, and each block is 256 bytes in size. Therefore, you would assign 1 to the "Block" port and 234 to the "Addr" port, if you want to obtain the data of Block 1, Offset 234.
3. **Data –** This port is used to output the data read. It will output a value between 0 and 255.

WriteToEEP



Function:

Write data to EEPROM

Library:

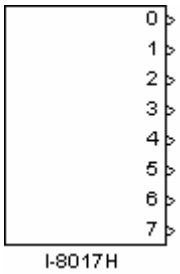
System

Description:

This block is used to write data to EEPROM, one byte per loop. To operate this block correctly, you need to keep some details in mind. They are given as follows:

1. **Block –** In the I-8438/8838 control system, the eeprom is divided into 8 blocks. The number of the block is counted from 0 to 7. Because block 7 is reserved for MINIOS7, however, users shouldn't assign the value, 7, to this port.
2. **Addr –** In each block of the eeprom, it is 256 bytes in size. So users can assign a value between 0 and 255 to this port.
3. **Data –** The data that you want to write to the eeprom. A value 0 ~ 255 is meaningful.

I-8017H



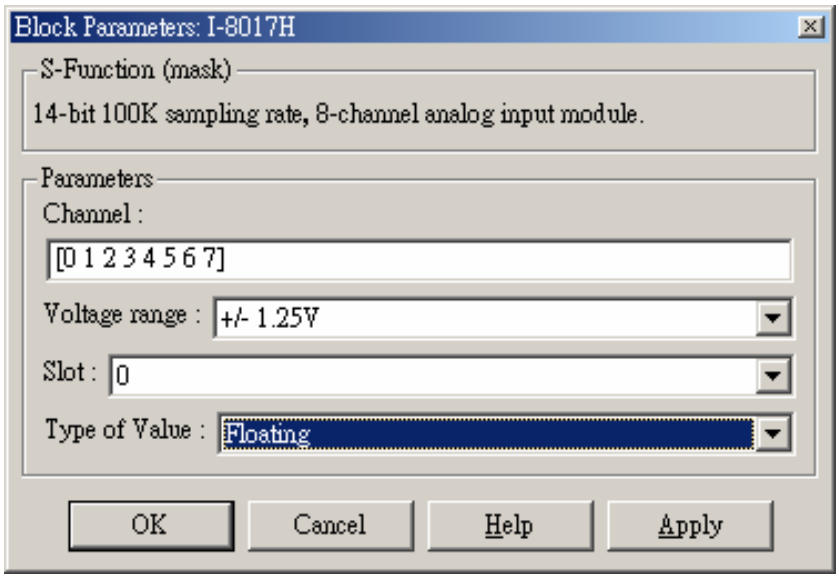
Description:

8-channel Isolated Analog Input Module.

Library:

AI

Dialog Box:



Driver Block Parameters:

Channel - Enter numbers between 0 and 7. This block allows the selection of analog input lines in any order. The number of elements defines the number of analog inputs used. For example, to use the first 8 analog inputs, enter [0 1 2 3 4 5 6 7]

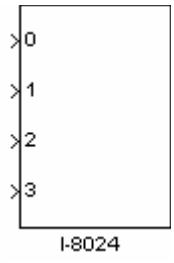
Voltage range - The I-8017H, AI module, provides 4 ranges of the input voltage, and they are +/-10V, +/-5V, +/-2.5V, and +/-1.25V.

Slot - The number of the slot where I-8017H module is located. For example, choose 2 from the popup list if you have mounted an I-8017H module on slot 2.

Type of Value - Floating or Hex is available. The table below presents the list of ranges of the input voltage.

Type of Value	Hardware Input	Block Output Value
Floating	-10 ~ 10V	-10 ~ 10
Hex	-10 ~ 10V	-8192 ~ 8191
Floating	-5 ~ 5V	-5 ~ 5
Hex	-5 ~ 5V	-8192 ~ 8191
Floating	-2.5 ~ 2.5V	-2.5 ~ 2.5
Hex	-2.5 ~ 2.5V	-8192 ~ 8191
Floating	-1.25 ~ 1.25V	-1.25 ~ 1.25
Hex	-1.25 ~ 1.25V	-8192 ~ 8191

I-8024



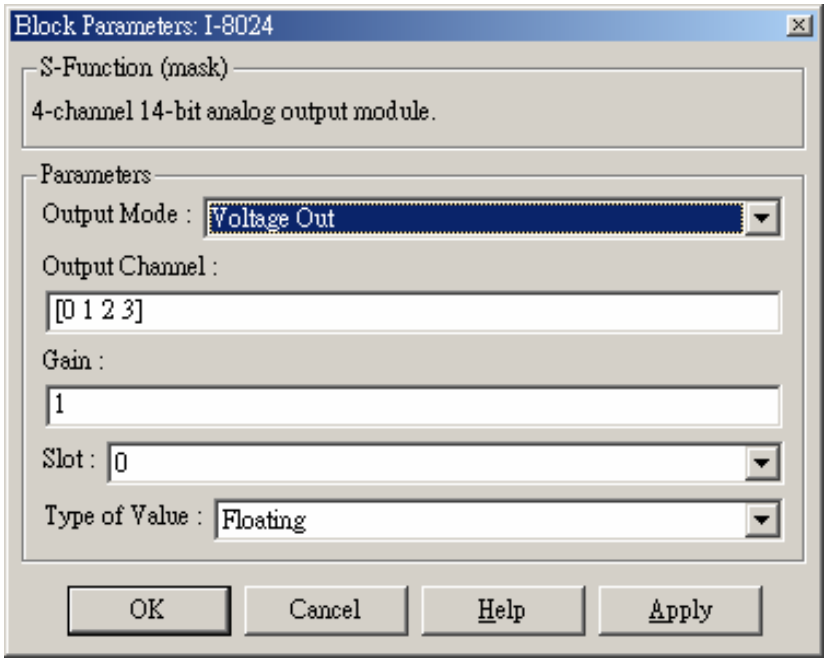
Description:

4-channel Isolated Analog Output Module.

Library:

AO

Dialog Box:



Driver Block Parameters:

Output Mode – *Voltage Out* or *Current Out* is available.

- Voltage Output -> +/-10V
- Current Output -> 0~20mA

Output channel - Enter numbers between 0 and 3. This block allows the selection of analog output lines in any order. The number of elements defines the number of analog outputs used. For example, to use the first 4 analog outputs, enter [0 1 2 3]

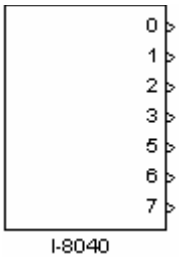
Gain - A multiplier. The default value is 1.

Slot - The number of the slot where I-8024 module is located. For example, choose 2 from the popup list if you have mounted an I-8024 module on slot 2.

Type of Value - Floating or Hex is available. The following table presents the difference between them.

Type of Value	Block Input Data Range	Hardware Output
Floating	-10 ~ 10 (Voltage Out) 0 ~ 20 (Current Out)	-10 ~ 10V (Voltage Out) 0 ~ 20mA (Current Out)
Hex	0 ~ 16383 (Both)	-10 ~ 10V (Voltage Out) 0 ~ 20mA (Current Out)

I-8040



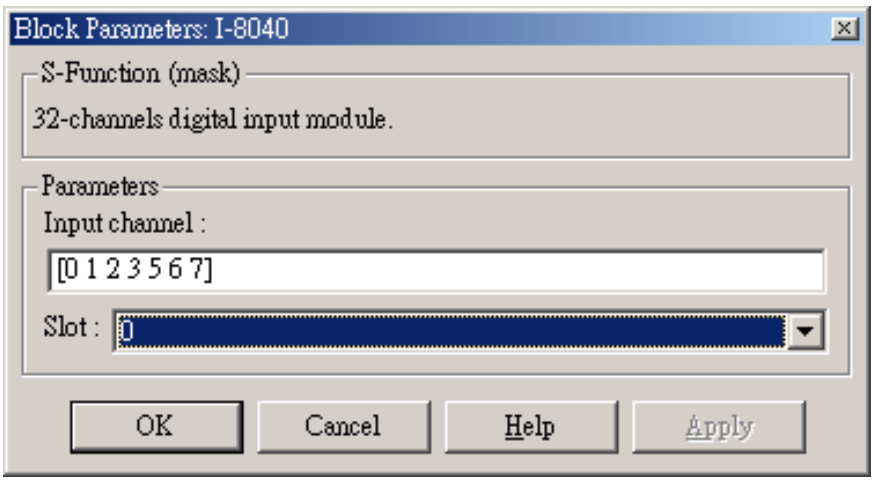
Description:

32-channel Isolated Digital Input Module.

Library:

DI

Dialog Box:



Driver Block Parameters:

Input channel - Enter numbers between 0 and 31. This block allows the selection of individual digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter

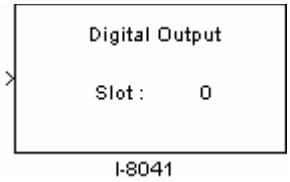
[0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8040 module is mounted. For example, choose 2 from the popup list if you have mounted an I-8040 module on slot 2.

Scaling Input to Output:

Hardware Input	Block Output Value
Below 3.5V	= 0
3.5V ~ 30V	= 1

I-8041



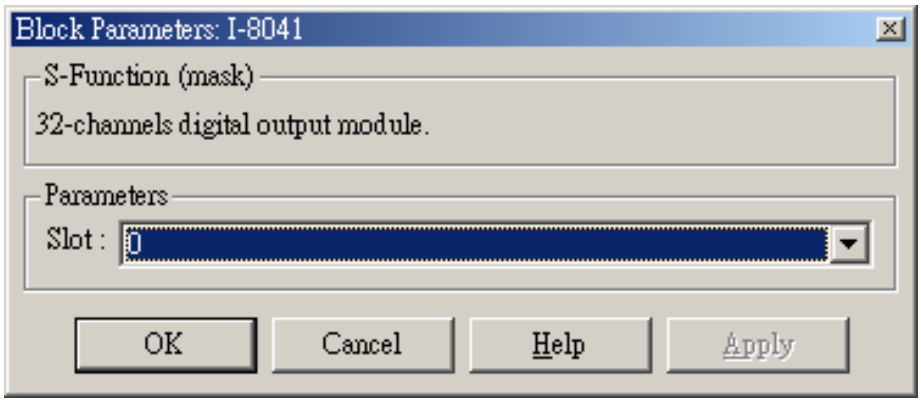
Description:

32-channel Isolated Digital Output Module.

Library:

DO

Dialog Box:



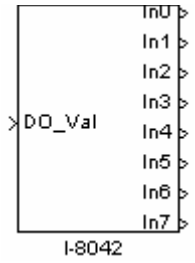
Driver Block Parameters:

Slot - The number of the slot where I-8041 module is located. For example, choose 2 from the popup list if you have mounted an I-8041 module on slot 2.

Scaling Input to Output:

BLOCK INPUT VALUE	HARDWARE OUTPUT
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on, and the others are off.
…	…
2^32 - 1	All channels are on.

I-8042



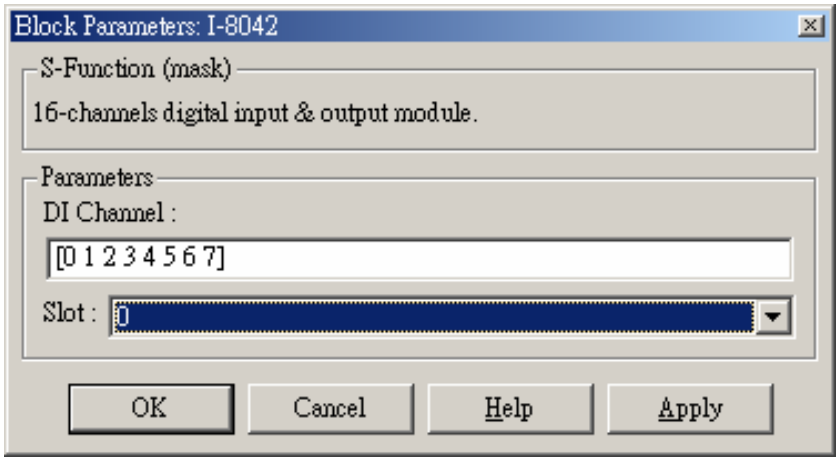
Description:

16-channel Isolated Digital Input & 16-channel Isolated Digital Output Module.

Library:

DIO

Dialog Box:



Driver Block Parameters:

DI Channel - Enter numbers between 0 and 15. This block allows the selection of digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter [0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8042 module is located. For example, choose 2 from the popup list if you have mounted an I-8042 module on slot 2.

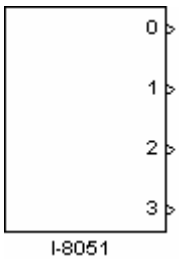
Scaling Input to Output (digital input):

Hardware Input	Block Output Value
Below 3.5V	= 0
3.5V ~ 30V	= 1

Scaling Input to Output (digital output):

Block Input Value	Hardware Output
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on, and the others are off.
…	…
65535	All channels are on.

I-8051



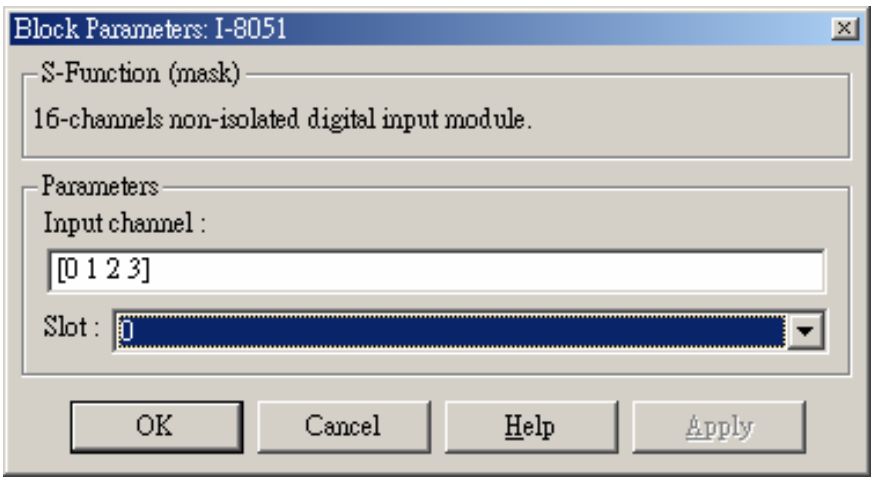
Description:

16-channel Non-isolated Digital Input Module.

Library:

DI

Dialog Box:



Driver Block Parameters:

Input channel - Enter numbers between 0 and 15. This block allows the selection of individual digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter

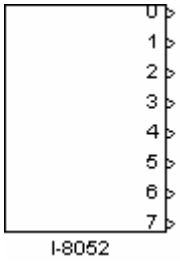
[0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8051 module is mounted. For example, choose 2 from the popup list if you have mounted an I-8051 module on slot 2.

Scaling Input to Output:

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

I-8052



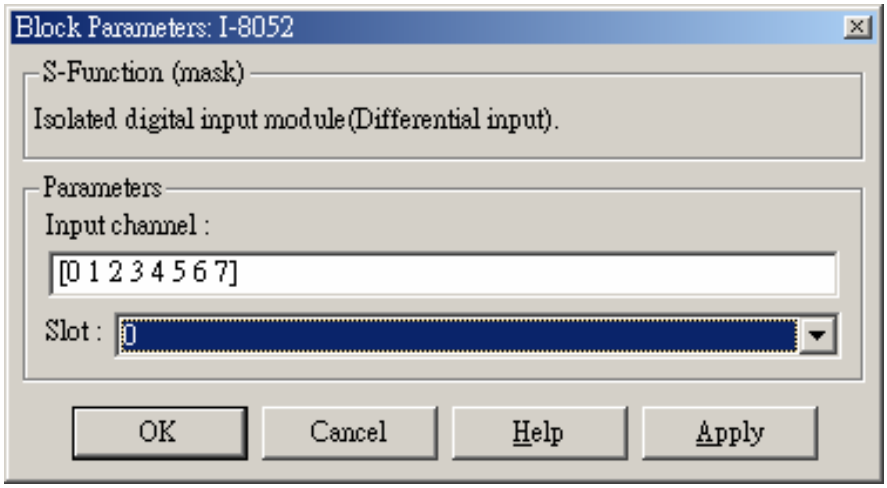
Description:

8-channel Isolated Digital Input Module (Differential input).

Library:

DI

Dialog Box:



Driver Block Parameters:

Input channel - Enter numbers between 0 and 7. This block allows the selection of individual digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter

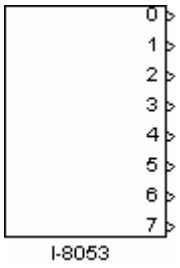
[0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8052 module is located. For example, select 2 from the popup list if you have mounted an I-8052 module on slot 2.

Scaling Input to Output:

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

I-8053



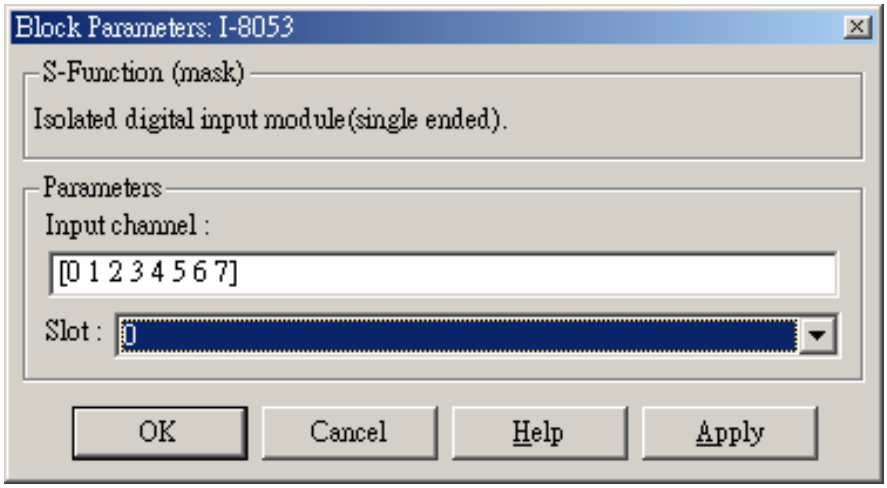
Description:

16-channel Isolated Digital Input Module (single ended).

Library:

DI

Dialog Box:



Driver Block Parameters:

Input channel - Enter numbers between 0 and 15. This block allows the selection of individual digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter

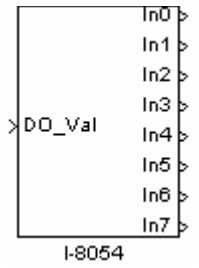
[0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8053 module is located. For example, select 2 from the popup list if you have mounted an I-8053 module on slot 2.

Scaling Input to Output:

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

I-8054



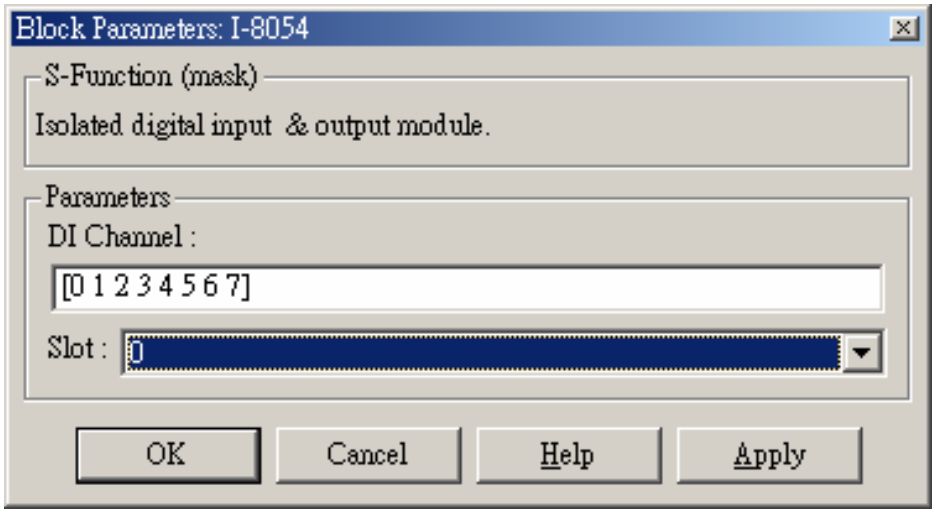
Description:

16-channel Isolated Digital I/O Module.

Library:

DIO

Dialog Box:



Driver Block Parameters:

DI Channel - Enter numbers between 0 and 7. This block allows the selection of digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter [0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8054 module is located. For example, choose 2 from the popup list if you have mounted an I-8054 module on slot 2.

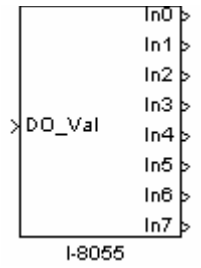
Scaling Input to Output (digital input):

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

Scaling Input to Output (digital output):

Block Input Value	Hardware Output
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on, and the others are off.
…	…
255	All channels are on.

I-8055



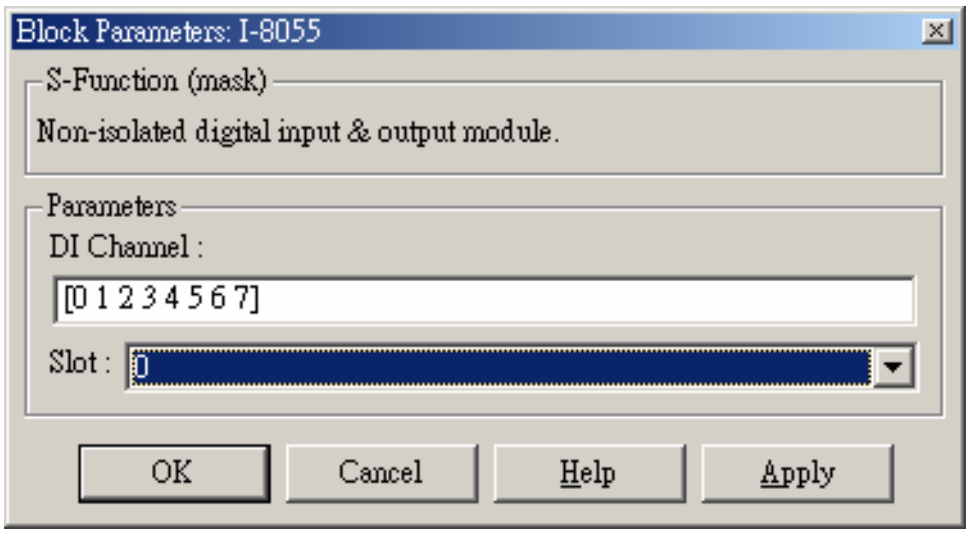
Description:

16-channel Non-isolated Digital I/O Module.

Library:

DIO

Dialog Box:



Driver Block Parameters:

DI Channel - Enter numbers between 0 and 7. This block allows the selection of digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter [0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8055 module is located. For example, choose 2 from the popup list if you have mounted an I-8055 module on slot 2.

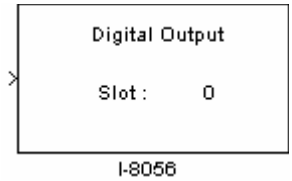
Scaling Input to Output (digital input):

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

Scaling Input to Output (digital output):

Block Input Value	Hardware Output
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on; the others are off.
…	…
255	All channels are on.

I-8056



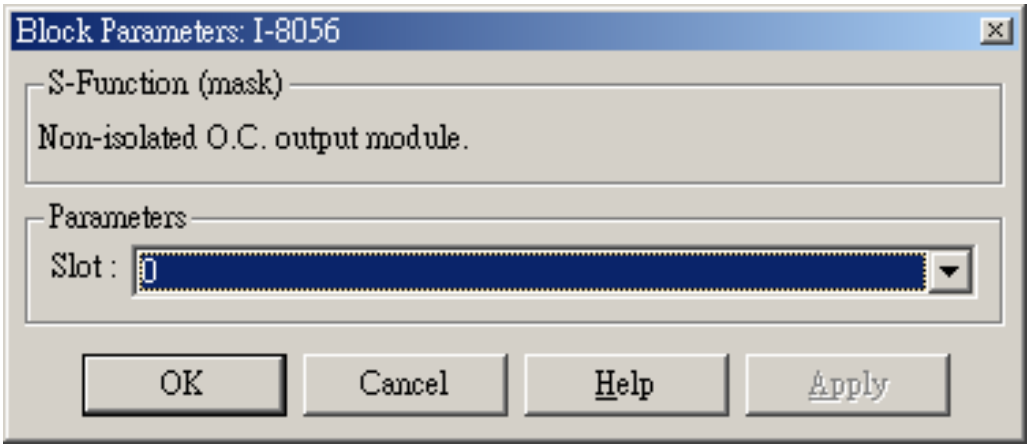
Description:

16-channel Non-isolated O.C. Output Module.

Library:

DO

Dialog Box:



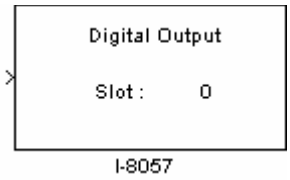
Driver Block Parameters:

Slot - The number of the slot where I-8056 module is located. For example, choose 2 from the popup list if you have mounted an I-8056 module on slot 2.

Scaling Input to Output:

BLOCK INPUT VALUE	HARDWARE OUTPUT
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on, and the others are off.
…	…
65535	All channels are on.

I-8057



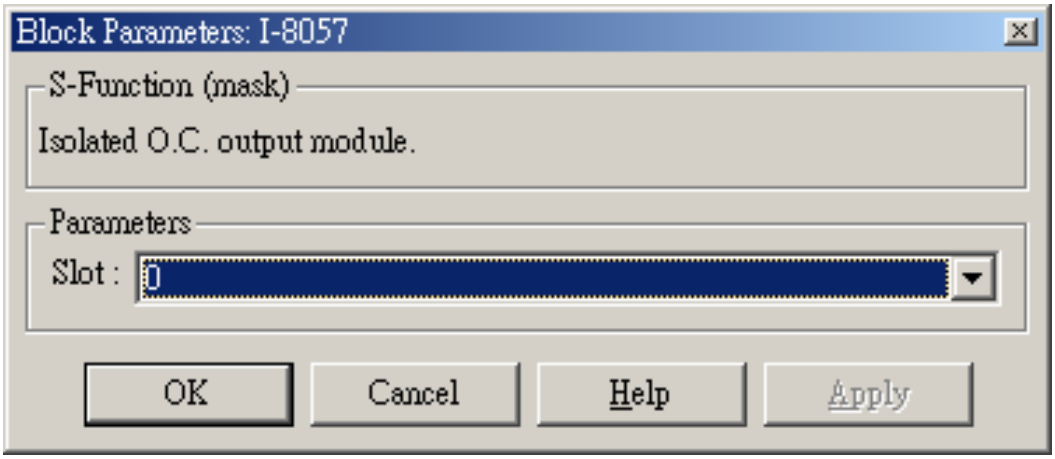
Description:

16-channel Isolated O.C. Output Module.

Library:

DO

Dialog Box:



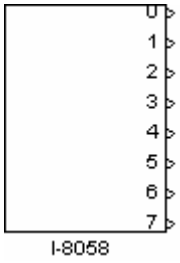
Driver Block Parameters:

Slot - The number of the slot where I-8057 module is located. For example, choose 2 from the popup list if you have mounted an I-8057 module on slot 2.

Scaling Input to Output:

BLOCK INPUT VALUE	HARDWARE OUTPUT
0	All channels are off.
1	Ch0 is on, and the others are off.
2	Ch1 is on, and the others are off.
…	…
65535	All channels are on.

I-8058



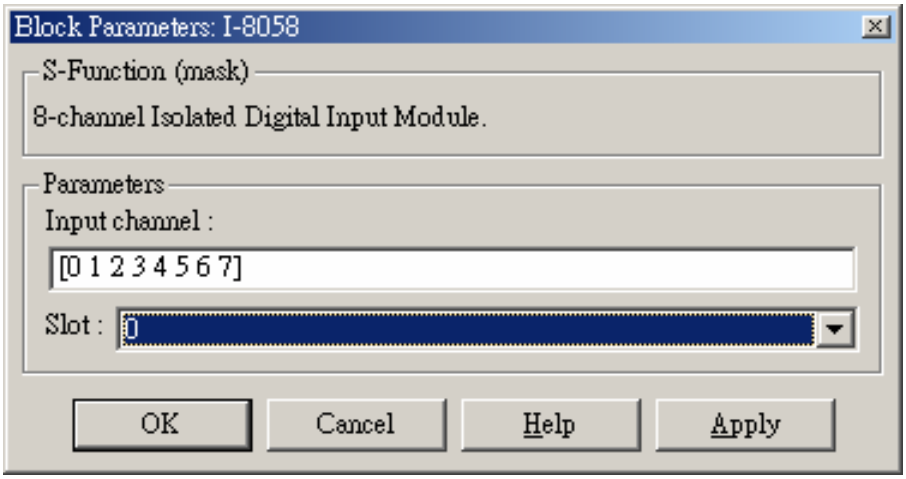
Description:

8-channel Isolated Digital Input Module.

Library:

DI

Dialog Box:



Driver Block Parameters:

Input channel - Enter numbers between 0 and 7. This block allows the selection of individual digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 8 digital inputs, enter

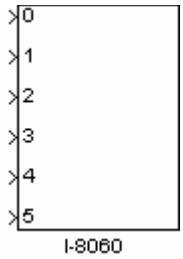
[0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8058 module is located. For example, choose 2 from the popup list if you have mounted an I-8058 module on slot 2.

Scaling Input to Output:

Hardware Input	Block Output Value
AC/DC 30V max.	= 0
AC/DC 80V mini	= 1

I-8060



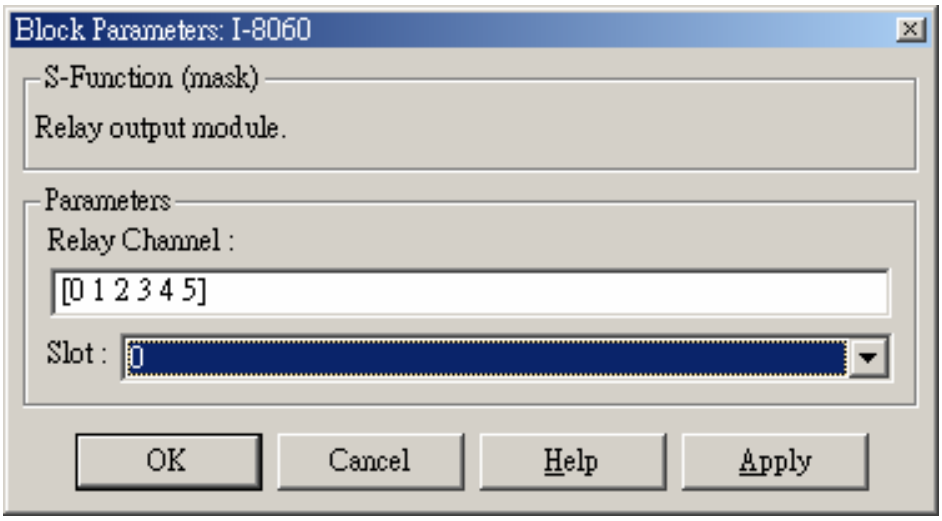
Description:

6-channel Relay Output Module.

Library:

RELAY

Dialog Box:



Driver Block Parameters:

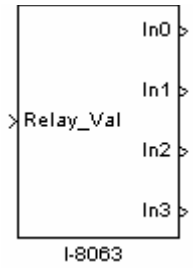
Relay Channel - Enter numbers between 0 and 5. This block allows the selection of relay output lines in any order. The number of elements defines the number of relay outputs used. For example, to use the first 6 relay outputs, enter [0 1 2 3 4 5]

Slot - The number of the slot where I-8060 module is located. For example, choose 2 from the popup list if you have mounted an I-8060 module on slot 2.

Scaling Input to Output:

Block Input Value	Hardware Output
> 0	Switch to NO
<=0	Switch to NC

I-8063



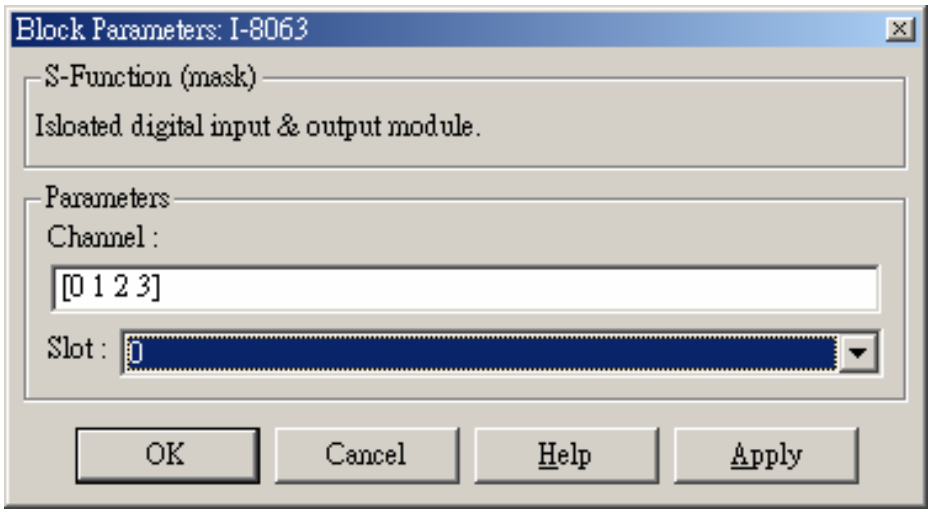
Description:

8-channel Isolated Digital I/O Module.

Library:

DIO

Dialog Box:



Driver Block Parameters:

DI Channel - Enter numbers between 0 and 3. This block allows the selection of digital input lines in any order. The number of elements defines the number of digital inputs used. For example, to use the first 4 digital inputs, enter [0 1 2 3]

Slot - The number of the slot where I-8063 module is located. For example, choose 2 from the popup list if you have mounted an I-8063 module on slot 2.

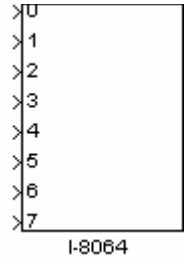
Scaling Input to Output (digital input):

Hardware Input	Block Output Value
Below 1V	= 0
3.5V ~ 30V	= 1

Scaling Input to Output (relay output):

Block Input Value	Hardware Output
0	All channels are switched to NO.
1	Ch0 is switched to NO, and the others remain NC.
2.	Ch1 is switched to NO, and the others remain NC.
…	…
7	All channels are switched to NC.

I-8064



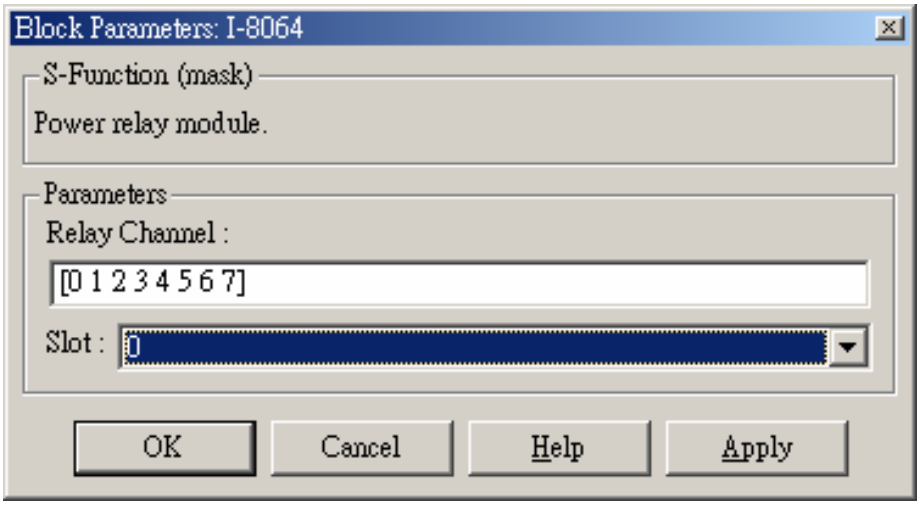
Description:

8-channel Power Relay Output Module.

Library:

RELAY

Dialog Box:



Driver Block Parameters:

Relay Channel - Enter numbers between 0 and 7. This block allows the selection of relay output lines in any order. The number of elements defines the number of relay outputs used. For example, to use the first 8 relay outputs, enter [0 1 2 3 4 5 6 7]

Slot - The number of the slot where I-8064 module is located. For example, choose 2 from the popup list if you have mounted an I-8064 module on slot 2.

Scaling Input to Output:

Block Input Value	Hardware Output
> 0	Switch to NO
<=0	Switch to off

I-8090

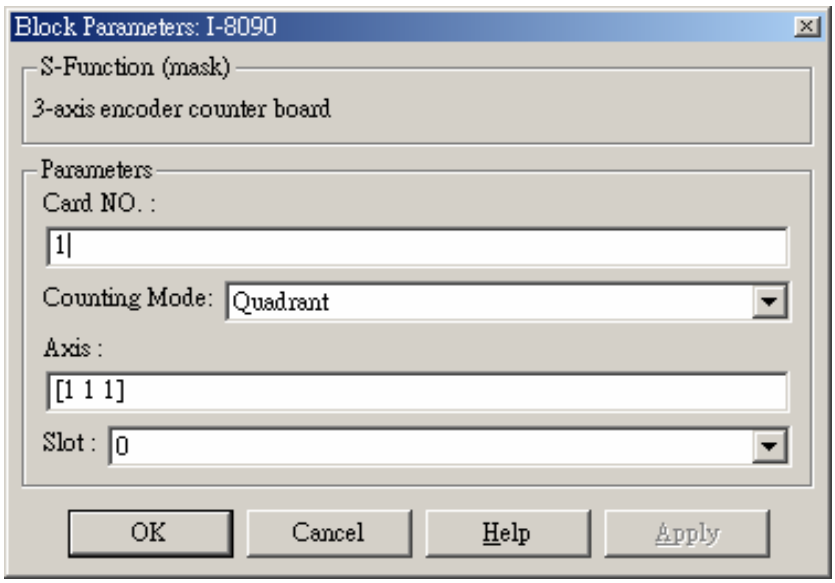
Function:

3-axis Encoder Input Module.

Library:

Encoder

Dialog Box:



Driver Block Parameters:

Card NO - The number is used to identify different I-8090 encoder input modules. A value between 0 and 19 is allowed.

Counting Mode - *Quadrant*, *CW/CCW*, and *pulse/direction* is available. This setting depends on the wire connections of the I-8090 encoder module.

Axis - The selected axis vector. The default value is [1 1 1]. Each of the elements in the vector is used to determine which axis is selected or not.

For example, if we enter [1 1 0] in the “Axis” field, the I-8090 driver block should look like as follows:

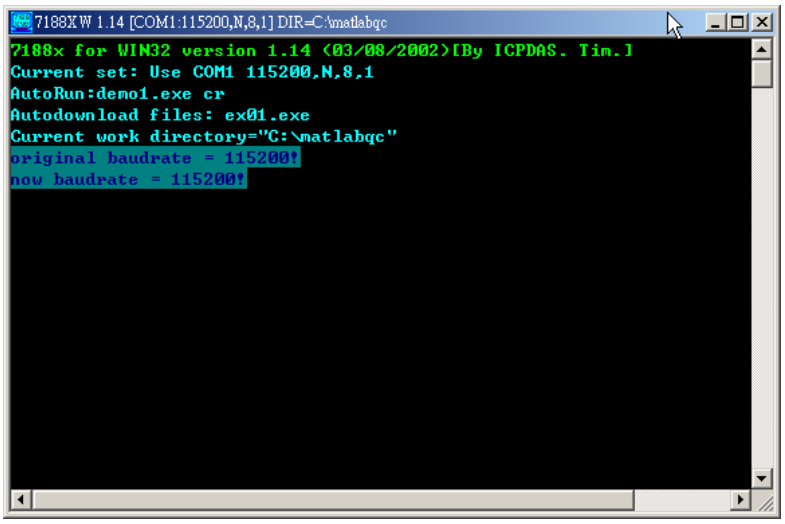
Slot - The number of the slot where I-8090 module is located. For example, choose 2 from the popup list if you have mounted an I-8090 module on slot 2.

Appendix A. Updating the OS of I-8438/8838

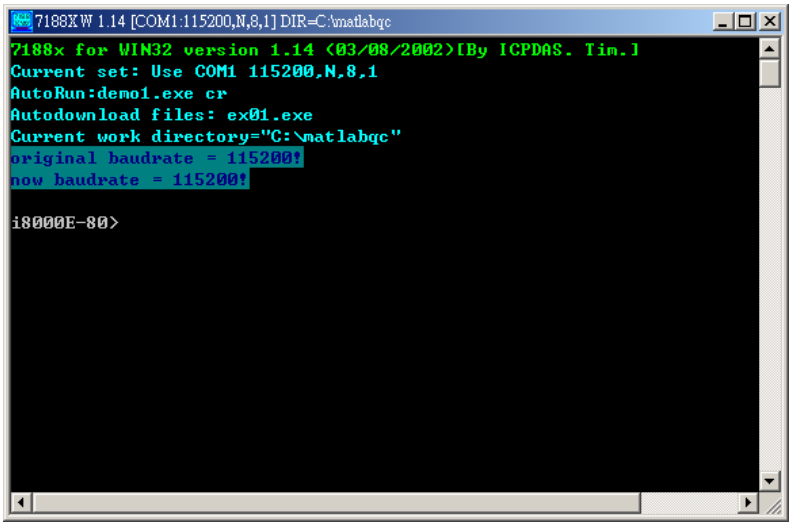
Before you start to update MiniOS7 on your I-8438/8838 embedded controller, 7188xw.exe is needed and can be found on the path CD:\Napdos\MATLAB\Firmware\1.00\. In addition, you can get the OS img file on the path, CD:\Napdos\MATLAB\OS_image\. To update MiniOS7 into a newer version, please follow these steps below. (**Note: You shouldn't turn off the I-8438/8838 embedded controller during the update, or unexpected damage may be caused.**)

Step 1. First of all, connect an rs232 cable from COM1 on the PC to COM1 of the I-8483/8838. Then turn on the I-8438/8838.

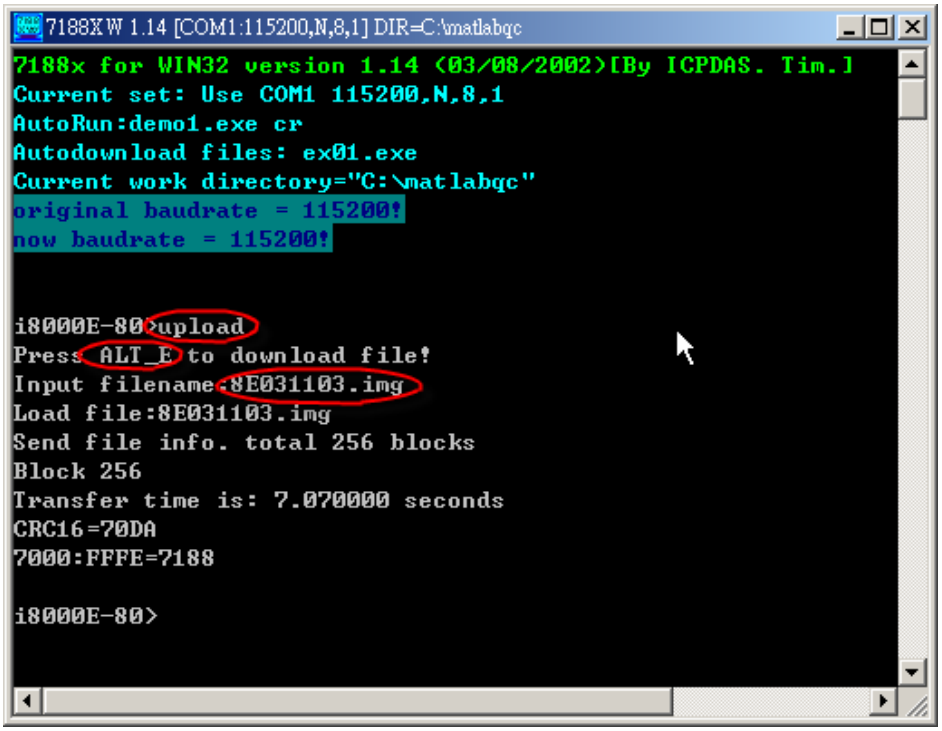
Step 2. Change to the directory where 7188xw.exe is located. Run 7188xw.exe and the following window shows up.



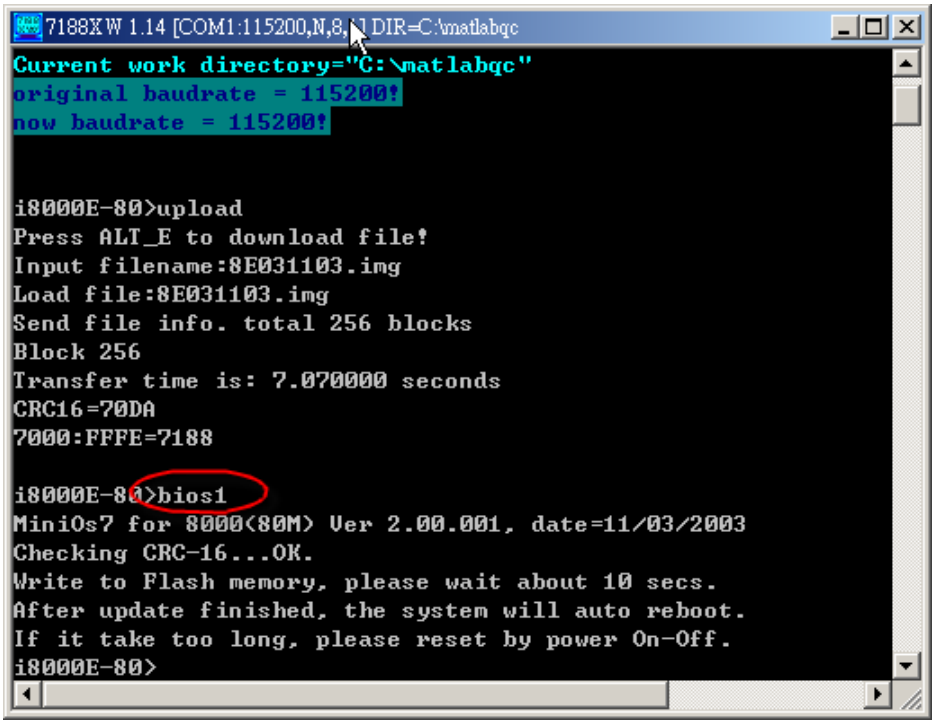
Step 3. Enter **Esc** to stop the execution of the firmware. When it was done, the window should look like this.



Step 4. Copy the firmware ***.img to the directory where 7188xw.exe is located. Then, enter **upload** at the prompt **i8000E-80>**. After the prompt message is displayed, press **ALT+E** to download the file ***.img.



Step 5. Then enter **bios1** and press Enter. The firmware updating starts automatically.



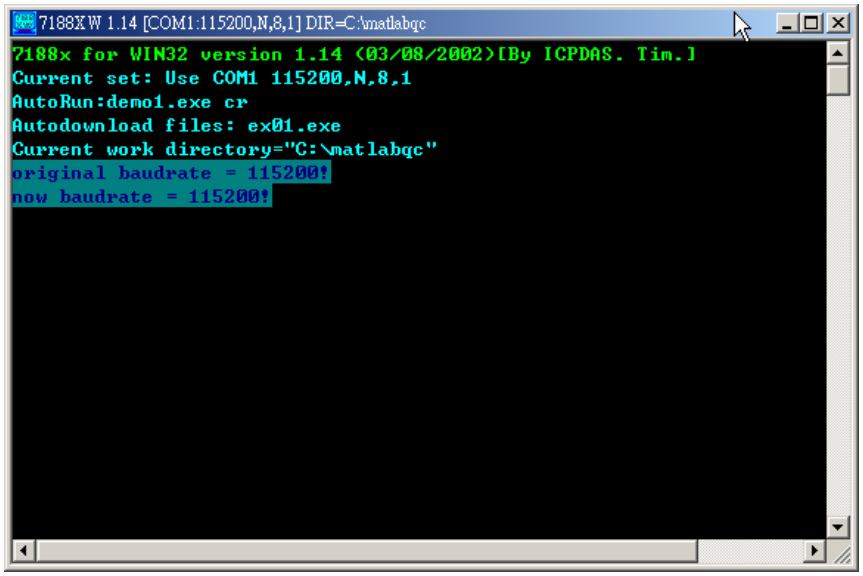
Appendix B. Updating the firmware

To update the firmware of your I-8438/8838 embedded controller, you can do as the following steps.

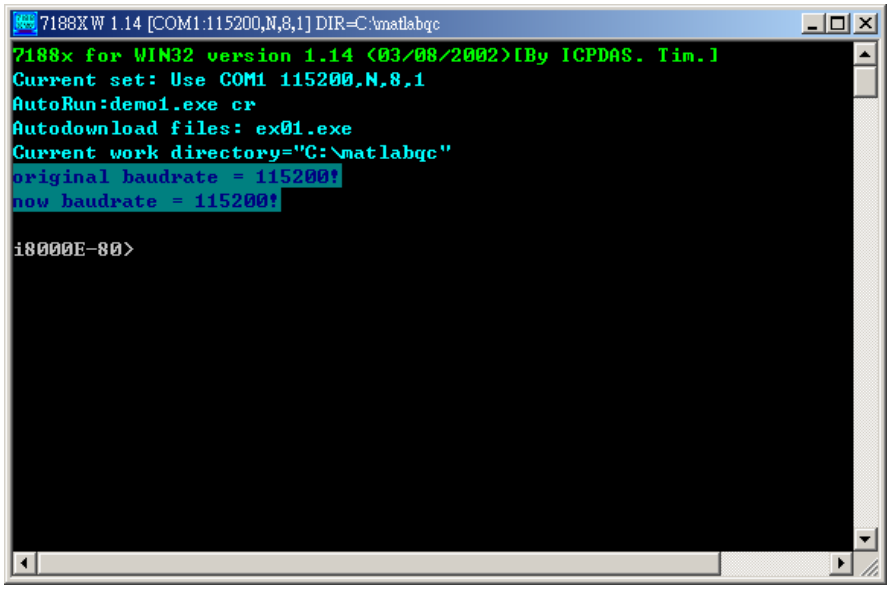
Step 1. At first, connect an rs232 cable from COM1 on PC to COM1 of the I-8438/8838 embedded controller.

Step 2. Turn on the I-8438/8838 embedded controller.

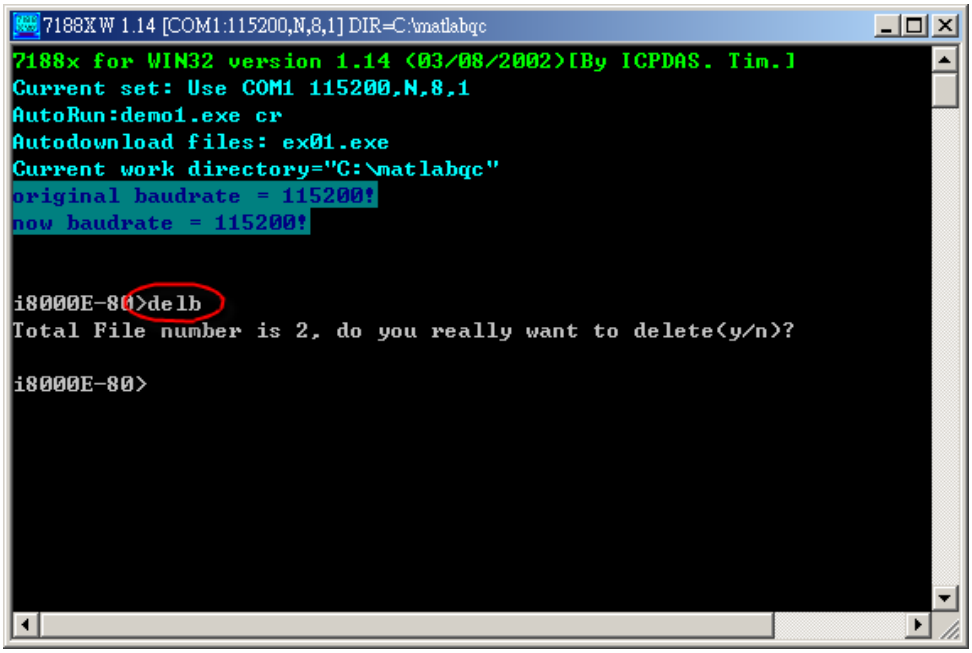
Step 3. Change to the directory where 7188xw.exe is located and run 7188xw.exe.



Step 4. Then Enter **Esc** to stop the execution of the firmware.

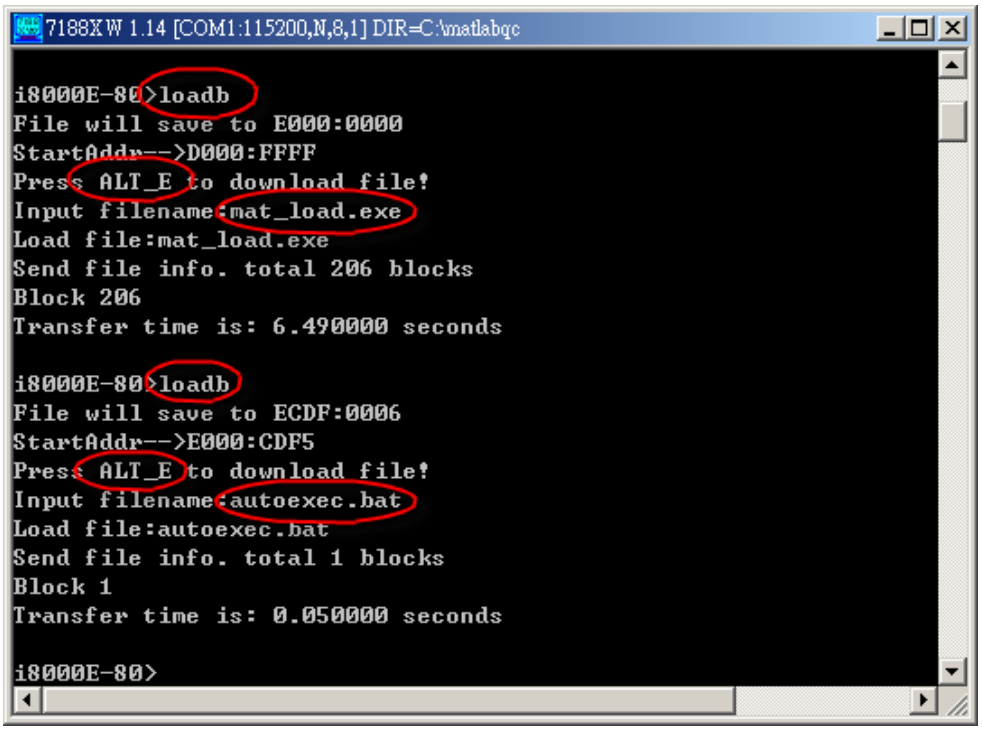


Step 5. Enter **delb** after the prompt of **i8000E-80>** to remove the original firmware. When the prompt message is displayed, press **y**.

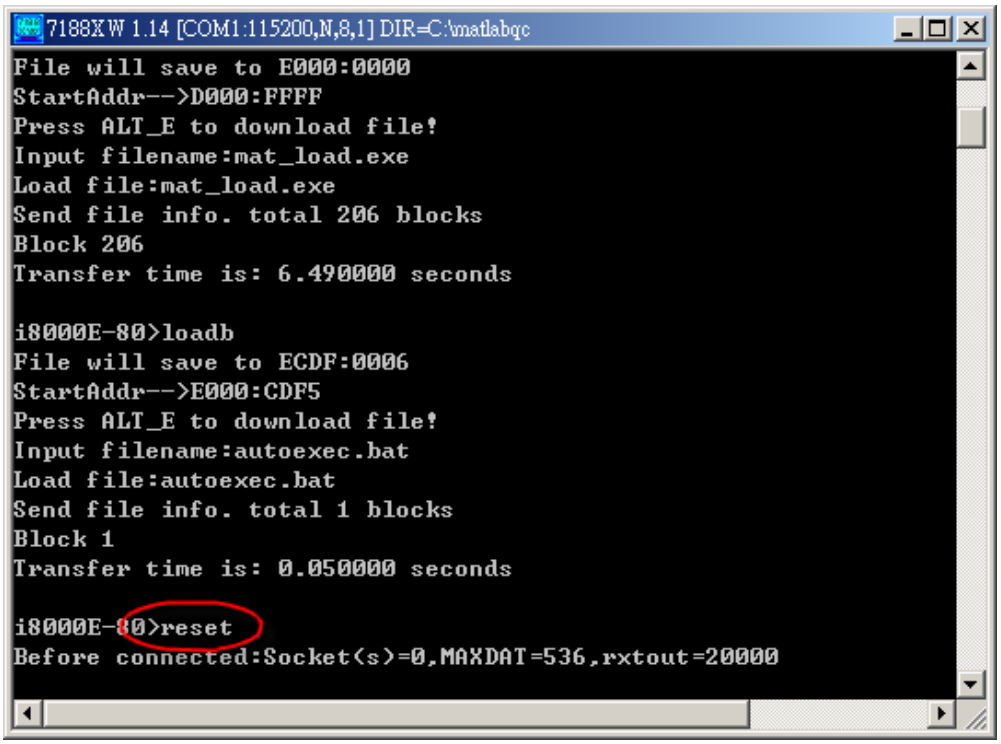


Step 6. Then copy the newer firmware **mat_load.exe** & **autoexec.bat** to the same directory.

Step 7. Enter **loadb** at the **i8000E>** prompt. When the prompt message is displayed, press **ALT+E** to enter the filename. Here we enter **mat_load.exe**. In a similar manner, download the file, **autoexec.bat**, to the I-8438/8838.



Step 8. Enter **reset** to restart the firmware.



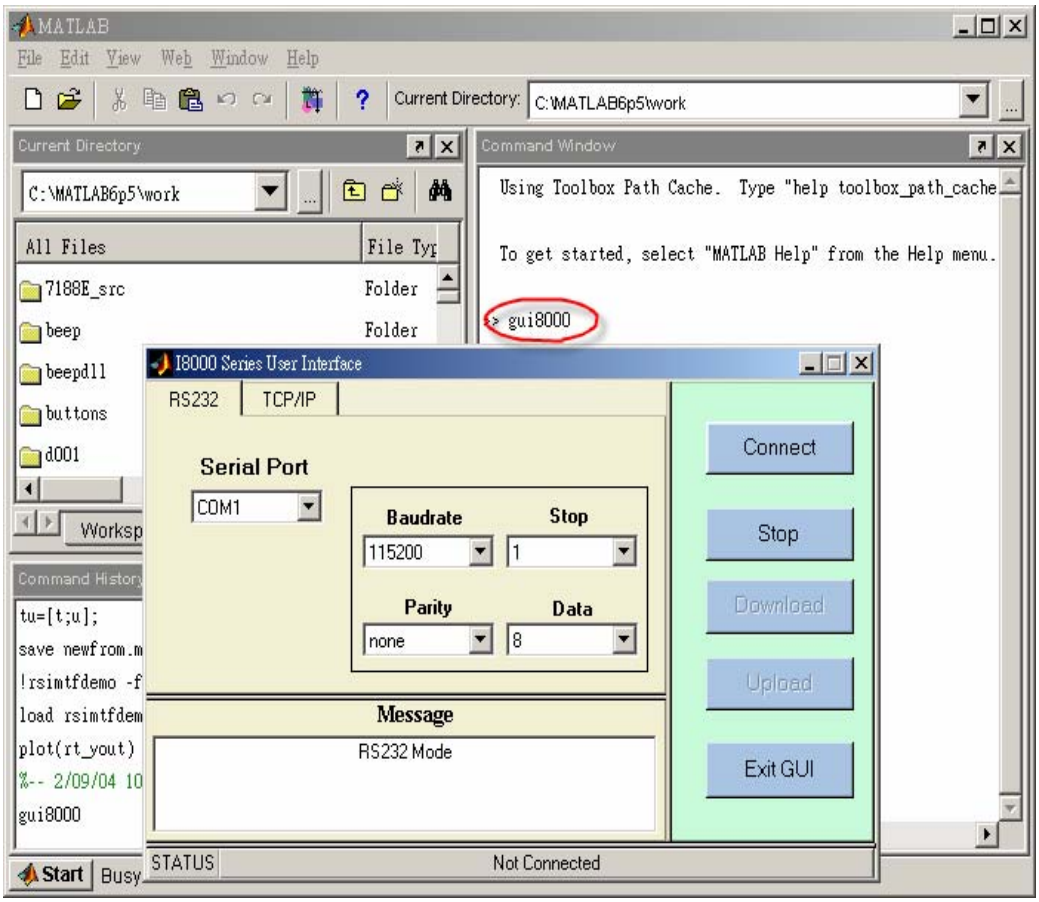
Appendix C. Change the IP of I-8438/8838

There provides two ways for users to change the IP, Mask and Gateway of I-8438/8838 embedded controller. One of them is through the GUI under MATLAB, and the other is through the push buttons on the S-MMI of the I-8438/8838. Next In the rest of this chapter, we will demonstrate you how to complete this work.

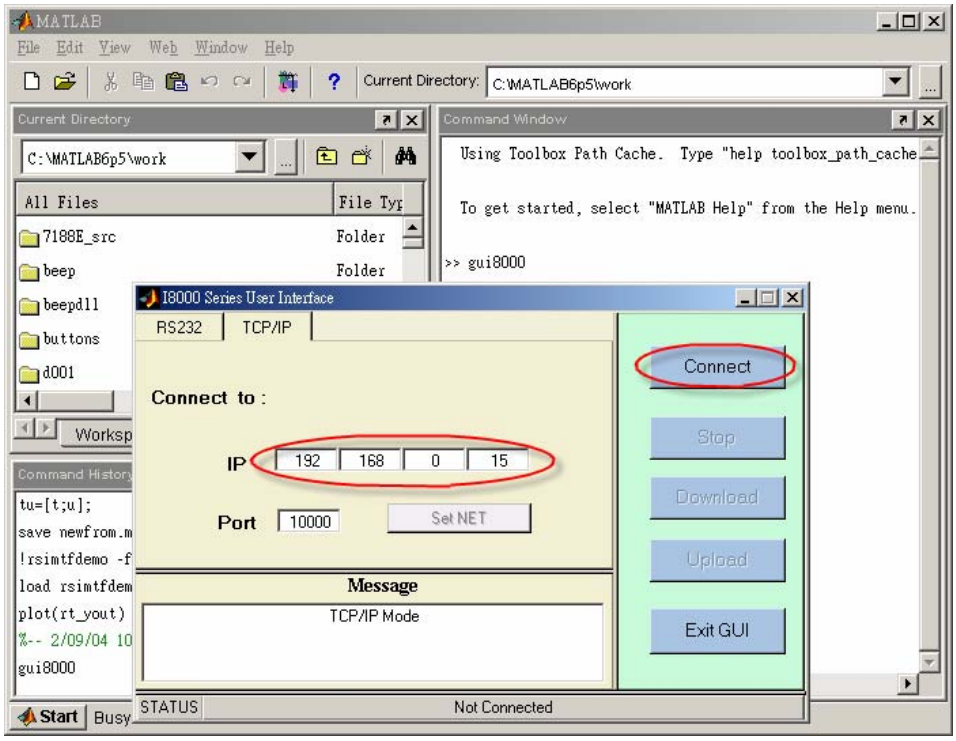
(Note: Before you continue to change the IP, Mask, and Gateway, make sure that the I-8438/8838 has been turned on.)

GUI under MATLAB

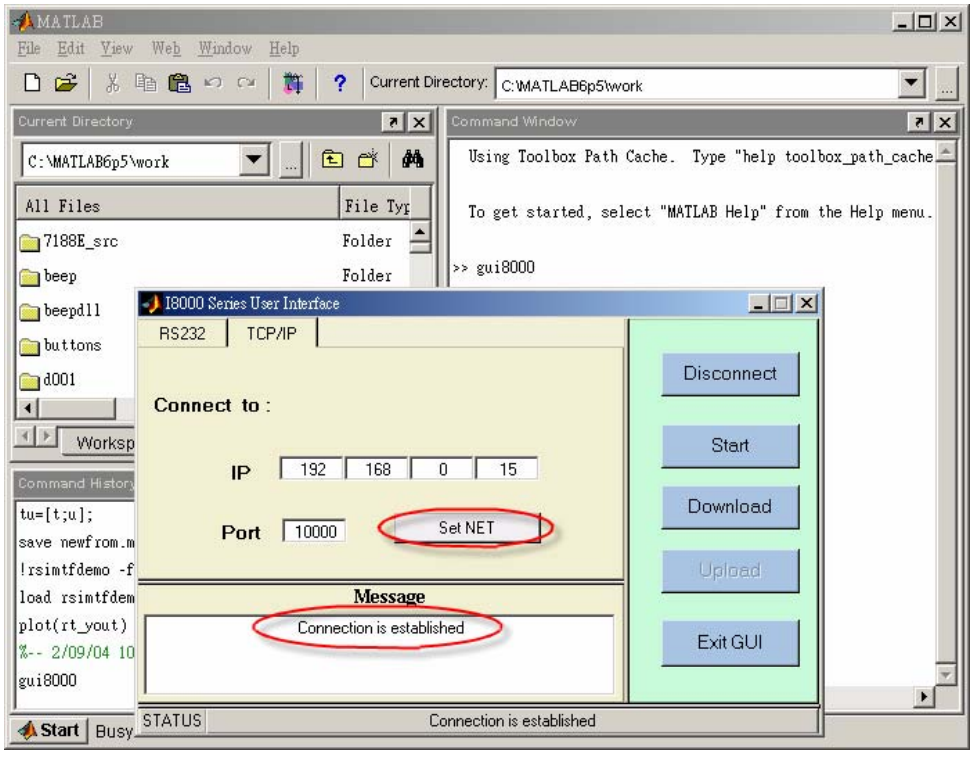
Step 1: Start MATLAB, and then enter **gui8000** at the MATLAB prompt.



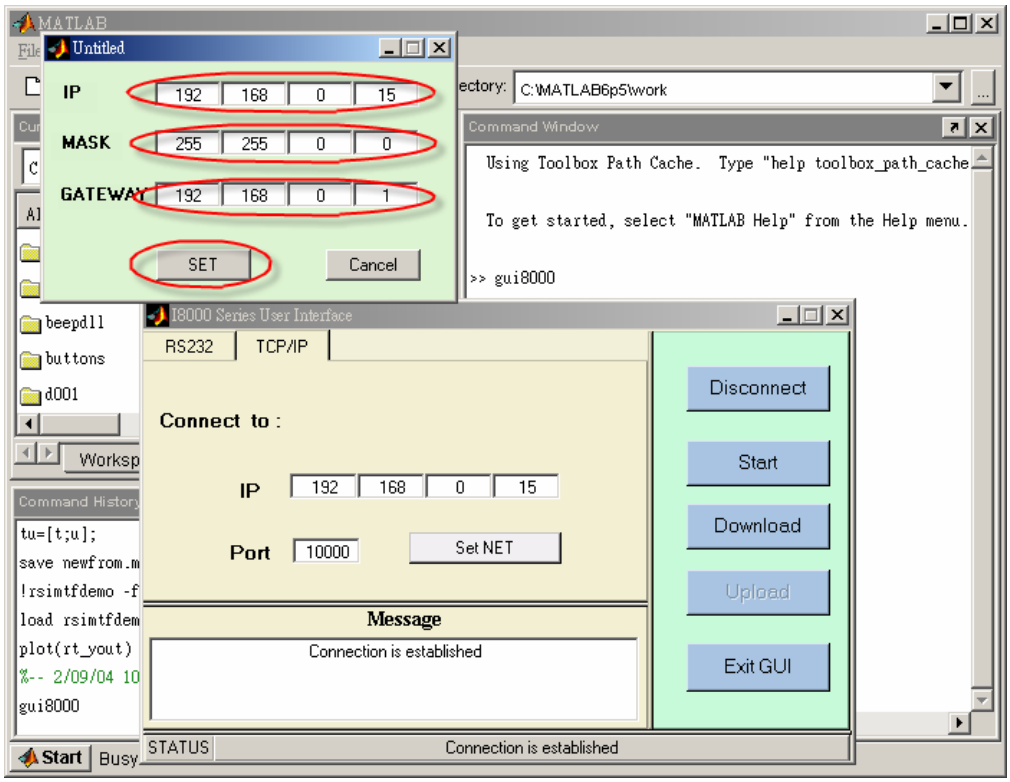
Step 2: On the TCP/IP page, enter **192.168.0.15** in the field 'IP' and **10000** in the field 'Port' respectively. Then click **Connect** to establish a connection with I-8438/8838.



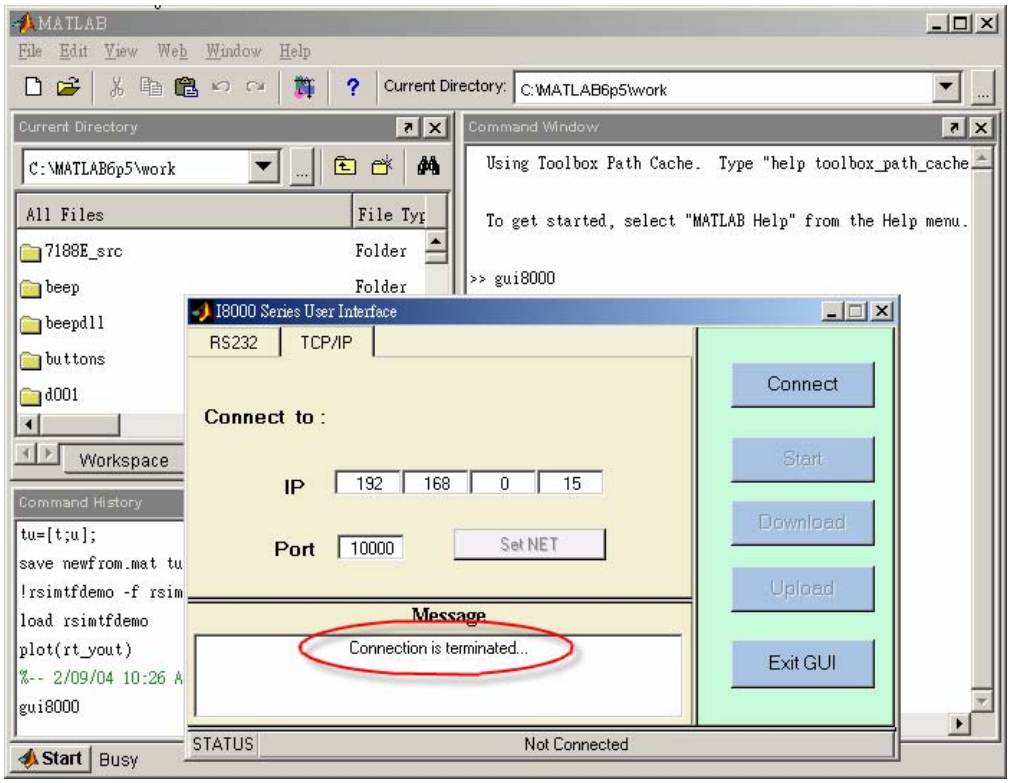
Step 3: When the connection was established successfully, the grayed button **SetNET** is enabled.



Step 4: Click SetNET to open the setting window as shown in the figure below. Enter intended ip, mask, gateway in the field 'IP', 'MASK', and 'GATEWAY' respectively. Then press **SET** to change the setting.

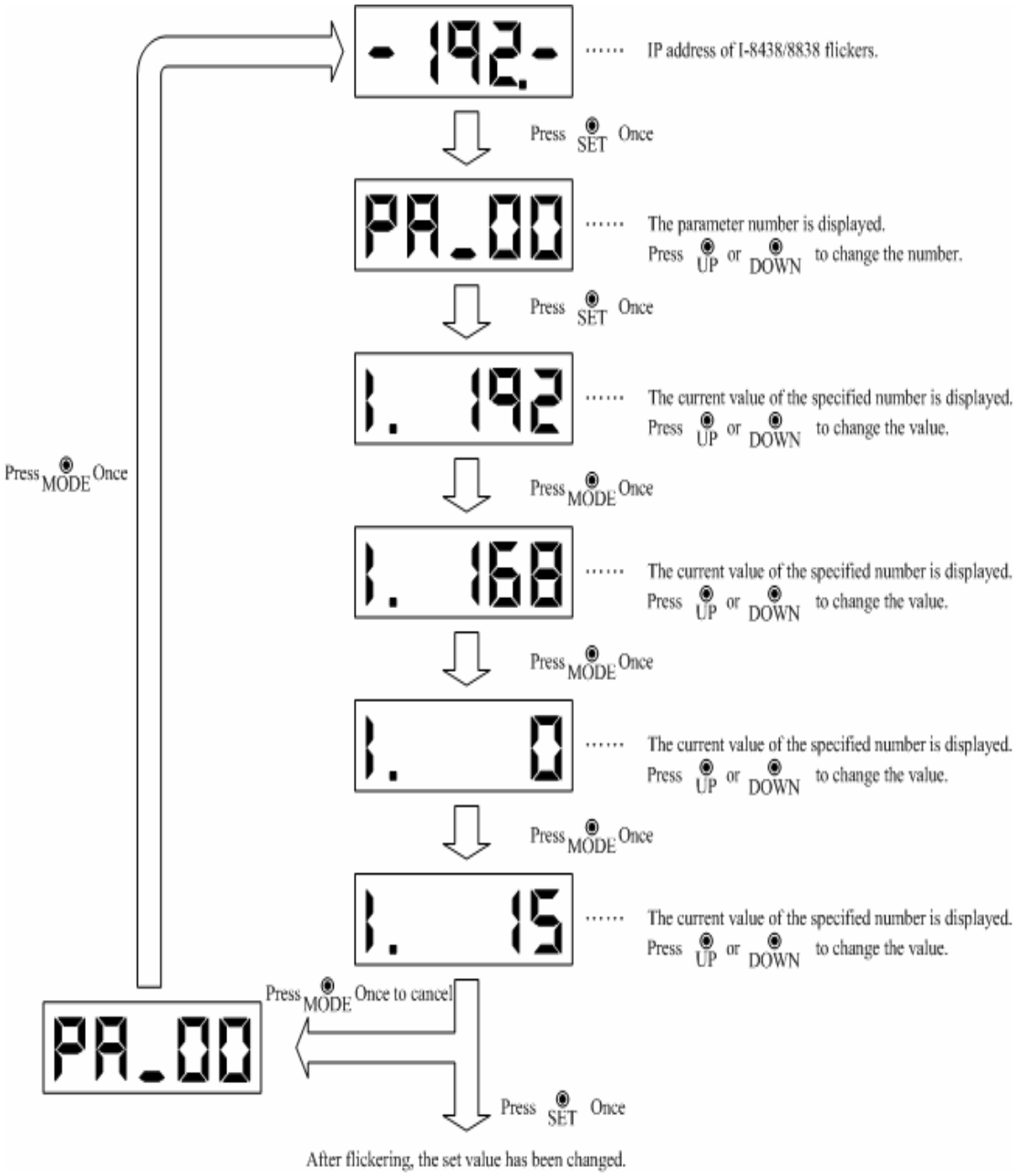


Step 5: If the process was completed successfully, the connection will be disconnected. And you will need to connect again.



Push Buttons on S-MMI

If you want to change ip, mask, and gateway of the I-8438/8838 via push buttons on S-MMI, you can follow the procedures shown in the figure below. We provide three parameters for users to set. They are PA_00, PA_01, and PA_02, which correspond with ip, mask, and gateway respectively. In the following figure, the alteration of ip is demonstrated. You can change the mask and gateway in a similar manner except different parameter.



Appendix D. Addition Function for New Driver Version

【New Function for Matlab Driver v1.1】

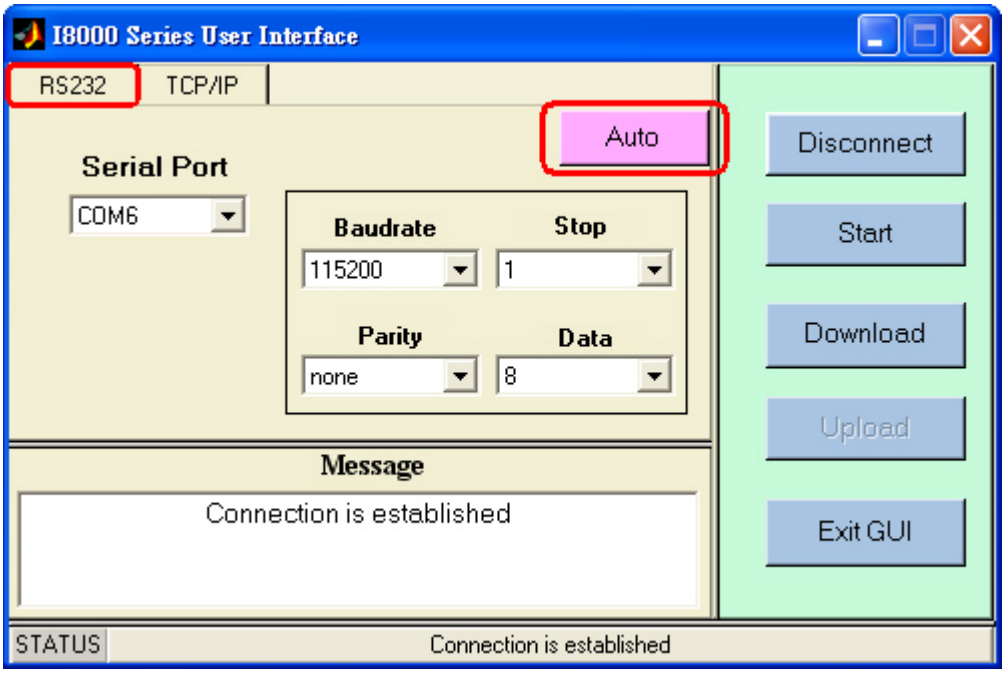
There are two new buttons -- "**Auto**" and "**eAuto**" provided in the GUI screen for RS-232 and Ethernet interface separately. After users complete connection and download file to the matlab controller, users can click "Auto" or "eAuto" button to execute the following continuous steps automatically：

(1) "**Run Program**"

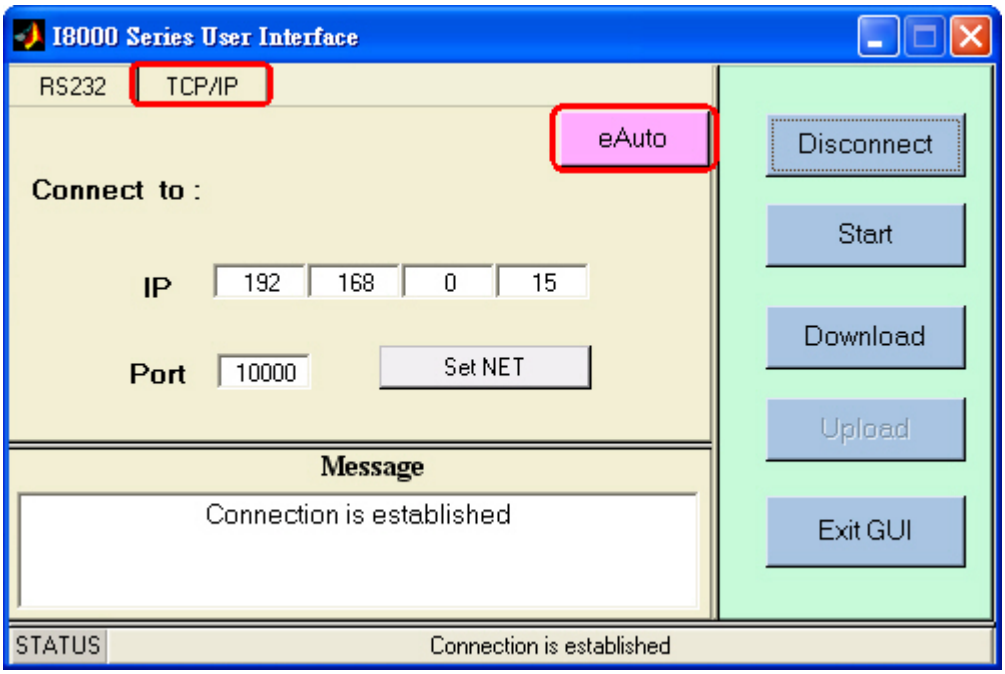
(2) "**Upload when stop time is up**"

(3) "**Close GUI**"

That will be more convenient for users to test the control algorithm during the development stage. These two Auto button are showed as the following figure.



RS-232 Interface

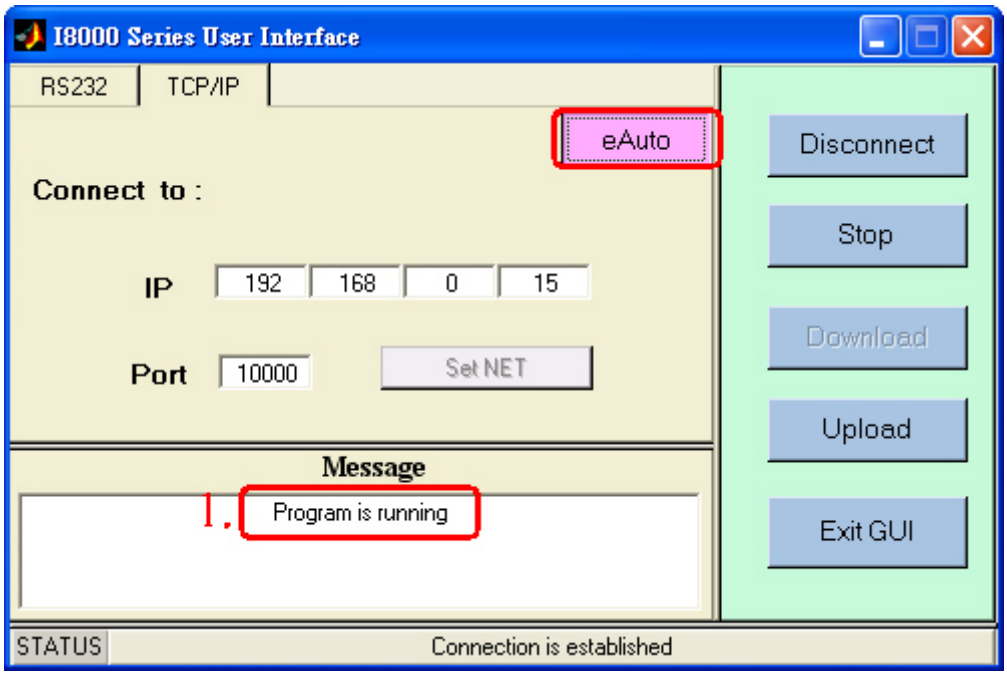


Ethernet Interface

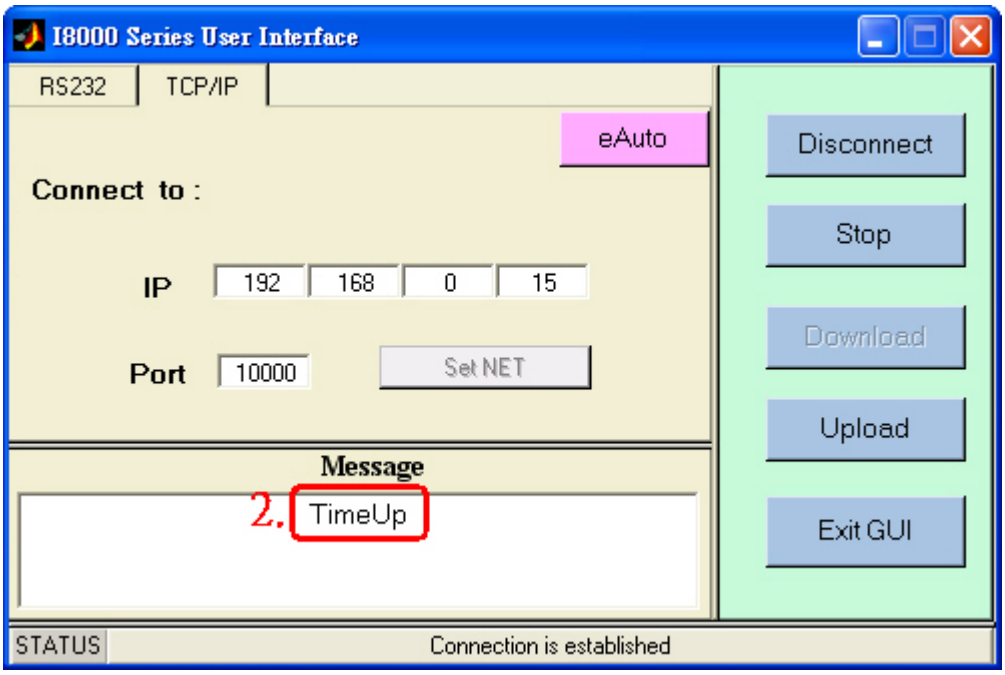
The whole process for **Auto** button function is described as following (Ethernet Interface Example)：

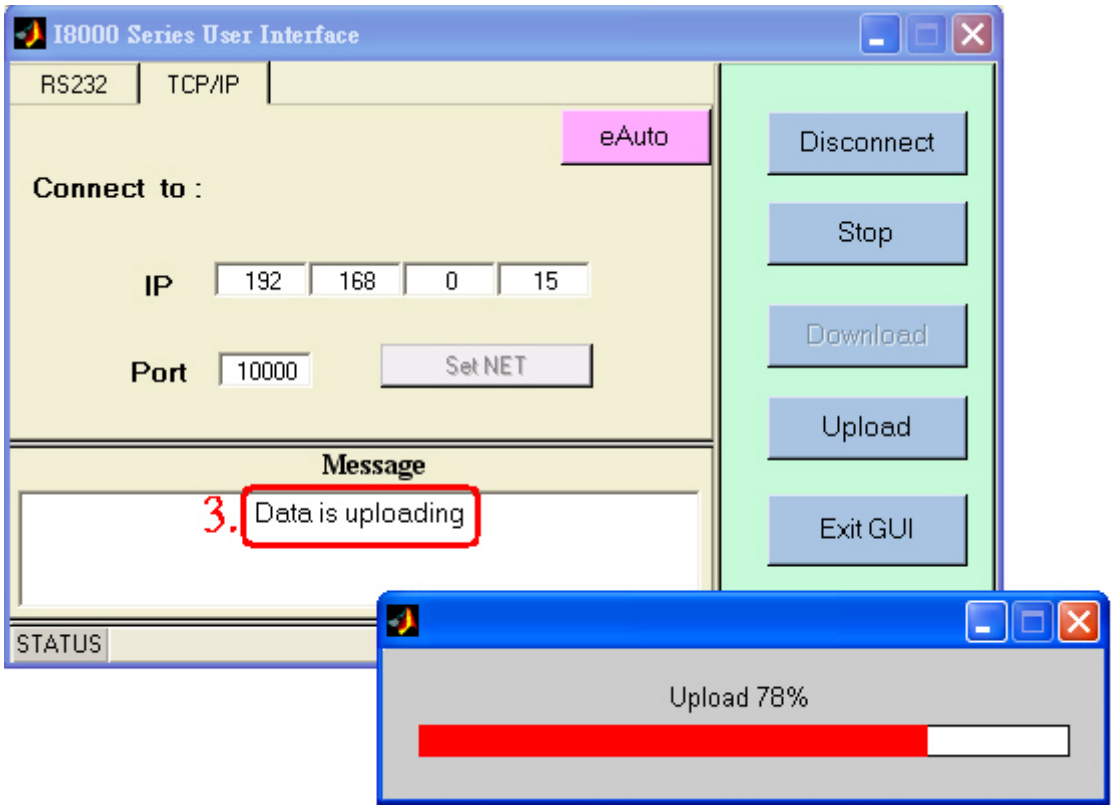
After users complete the connection and download file to the matlab controller, users can click Auto button in the GUI screen to execute the following steps：

(1) "**Run Program**"：

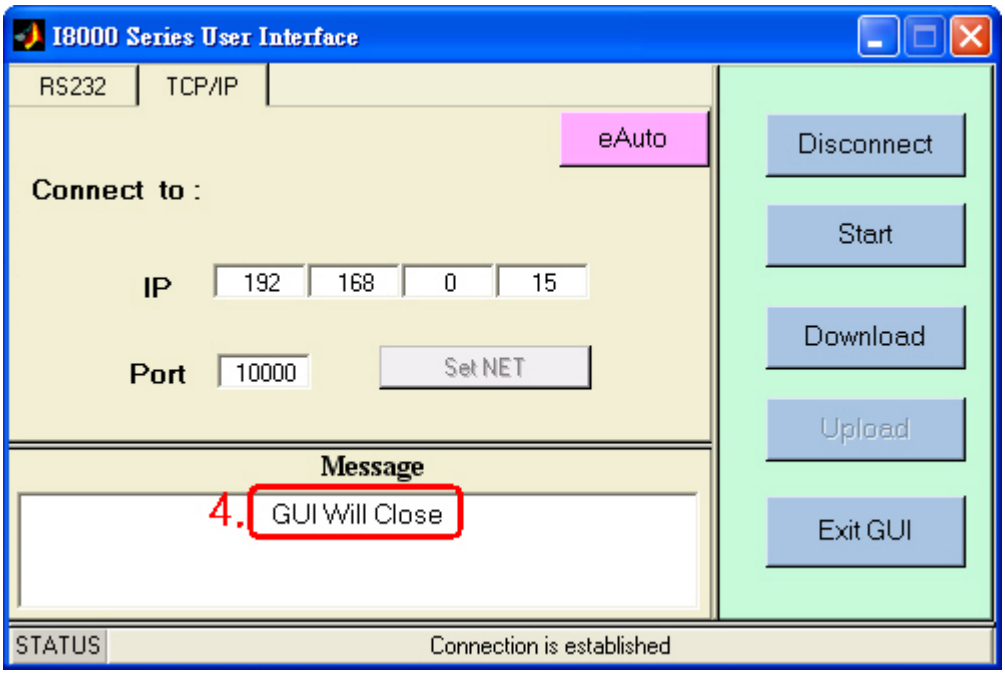


(2) "Upload when stop time is up" :





(3) "**Close GUI**" :


I8000 Series User Interface
RS232
TCP/IP
eAuto
Connect to :
IP
192
168
0
15
Port
10000
Set NET
Message
4.
GUI Will Close
Disconnect
Start
Download
Upload
Exit GUI
STATUS
Connection is established

Appendix E. History of Versions

Version	By	Date	Description of changes
1.0.0	Edward	15-June-04	1. First driver version
1.1.0	Edward	10-July-06	1. Add “Auto” function for RS-232 / Ethernet 2. RS-232 / Ethernet connection will be ok no matter what controller mode is. 3. Add RS-232 connection number to “COM 8” 4. Modify slow speed download file via Ethernet problem