



EcoCoder-AI

User Manual

V2.3

Revision History

Time	Version	Detail	Reviser
May. 2, 2019	V1.0	First version	David Wang
Sep. 20, 2019	V1.1	First page updated	David Wang
Nov. 10, 2019	V1.2	Add external mode	David Wang
Feb. 19, 2020	V2.0	Block update, detailed using instructions, external mode update	David Wang Luke Wang
Feb. 19, 2020	V2.1	Summary update	David Wang
Apr. 17, 2020	V2.2	Add 3 dependencies	David Wang
May 11, 2020	V2.3	Contact info updates	Zack Li

Contact us

Web: <http://www.ecotrons.com>

Email: info@ecotrons.com

ev-support@ecotrons.com

Address: 13115 Barton Rd, Ste H

Whittier, CA 90605 USA

Telephone: +1 562-758-3039 / +1 562-713-1105

Fax: +1 562-352-0552

Content

Chapter 1 Summary	6
Chapter 2 Software Installation	7
2.1 Computer Configuration	7
2.2 Operating System.....	7
2.3 MATLAB Version	7
2.4 Dependencies for EcoCoder-AI Installed in Linux.....	8
2.4.1 Ubuntu	8
2.4.2 Minicom	8
2.4.3 ROS.....	8
2.4.4 EcoSDK-S2	9
2.4.5 EcoSDK-XV.....	10
2.4.6 MATLAB.....	10
2.4.7 OpenSSH.....	11
2.4.8 Expect.....	11
2.4.9 Tcl	11
2.5 Install EcoCoder-AI in Linux	12
2.5.1 Preparation	12
2.5.2 Installation	12
2.6 Install EcoCoder-AI in Windows.....	13
2.6.1 Preparation	13
2.6.2 Installation	13
Chapter 3 Quick Start.....	15

3.1 Quick Start in Linux	15
3.1.1 Create Working Directory	15
3.1.2 Open the Model	15
3.1.3 Build the Model.....	16
3.1.4 Build Other Models	17
3.1.5 Run on PC	18
3.1.6 Run on ACU	21
3.2 Quick Start in Windows.....	24
Chapter 4 Library	25
4.1 EcoCoder Target Definition.....	25
4.2 ROS&CyberRT.....	26
4.2.1 Publish Message Module	26
4.2.2 Subscribe Message Module	27
4.2.3 ROS Info Module	28
4.3 Socket CAN Module	29
4.3.1 Receive CAN Raw with Trigger.....	29
4.3.2 Transmit CAN Message.....	30
4.3.3 Read CAN Message	31
4.3.4 Send CAN Message	32
Chapter 5 Define ROS Topic.....	34
Chapter 6 External Mode.....	35
6.1 Demo Directory.....	36
6.2 Make the Model Support External Mode.....	36
6.2.1 New Model Supports External Mode by Default.....	36

6.2.2	Use Command-Line to Convert the Old Model	36
6.2.3	Change the Configuration of the Old Model	36
6.3	IP Settings for External Mode	38
6.4	Using Different Ports.....	38
6.4.1	Change the Port in Simulink.....	38
6.4.2	Change the Port in Target.....	39
6.5	EcoCoder Build Model	39
6.6	Connect the Host and the Target.....	40
6.6.1	For MATLAB 2019a or Earlier Version	40
6.6.2	For MATLAB 2019b	41
6.7	Special Notes for External Mode	42

Chapter 1 Summary

EcoCoder-AI is an add-on library to Matlab/Simulink for AI controllers, which use Nvidia Xavier, or NXP S32V234 chips. EcoCoder-AI integrates auto code generation, compilation and on-the-fly calibration in one tool chain. With EcoCoder-AI, a Simulink model can be converted directly into a Linux-based executable program for the target controller and deployed to the target. It helps developers make most of the Simulink generic libraries and use a model-based method to develop Linux-based applications. EcoCoder-AI has been used for autonomous driving control systems.

Chapter 2 Software Installation

2.1 Computer Configuration

Requirement	Minimum Configuration
CPU	Intel(R) Core(TM)2 Duo CPU E4600 @2.40GHz
RAM	2G
Hard Disk	10G+ available space
Requirement	Recommended Configuration
System Language	English
CPU	Intel(R) Core(TM) i3-2120 CPU@3.30GHz
RAM	4G
Hard Disk	10G+ available space

2.2 Operating System

The operating systems that EcoCoder-AI supports:

- Ubuntu 14.04_amd64
- Ubuntu 18.04_amd64
- Windows XP
- Windows 7
- Windows 10

Cross-compile can only be implemented in a Linux system. Therefore, the code generated in Windows needs to be copied to a Linux system or the target to compile.

2.3 MATLAB Version

EcoCoder-AI is compatible with MATLAB of following versions:

- MATLAB R2014b 32-bit/64-bit
- MATLAB R2015a 32-bit/64-bit
- MATLAB R2015b 32-bit/64-bit

- MATLAB R2016a 64-bit
- MATLAB R2016b 64-bit
- MATLAB R2017a 64-bit
- MATLAB R2017b 64-bit
- MATLAB R2018a 64-bit
- MATLAB R2018b 64-bit
- MATLAB R2019a 64-bit
- MATLAB R2019b 64-bit

2.4 Dependencies for EcoCoder-AI Installed in Linux

2.4.1 Ubuntu

Ubuntu is an open-source GNU/Linux operating system for desktop applications. Created by Canonical Ltd., Ubuntu is based on Debian GNU/Linux and supports x86, amd64 (x64) and other architectures. For more details, please refer to [Ubuntu official website](#) or [Ubuntu - Wikipedia](#). For instructions on how to install a desktop Ubuntu, please refer to [Installation guides](#). For Linux beginners, we recommend [quick tutorials](#) to get started with command-line tools.

- If ACU is EAS2A01, please install Ubuntu 14.04_amd64.
- If ACU is EAXVA01, please install Ubuntu 18.04_amd64.

2.4.2 Minicom

Minicom is a text-based modem control and terminal emulation program for Unix-like operating systems. Minicom is commonly used as a remote serial console, which may be the last resort to access a computer if the LAN is down. For instructions, please refer to [minicom\(1\) - Linux man page](#) or [Minicom - Community Help Wiki - Ubuntu Documentation](#). On Ubuntu systems, you can use the following command to install Minicom:

```
$ sudo apt-get install minicom
```

2.4.3 ROS

ROS (Robot Operating System) is a robot software platform that provides functions similar to an operating system for heterogeneous computer clusters. ROS provides some standard operating system services such as hardware abstraction, low-level device control, common function implementation, interprocess messages, and packet management. ROS is based on a graph-like architecture whereby processes at different nodes can subscribe, publish, and aggregate various information (such as sensing, control, state, planning, etc.). ROS can be divided into two layers, the lower layer is the operating system layer described above, and the upper layer is the various software packages that contributed by different user groups to realize different functions, such as localization and mapping, action planning, sensing, simulation and so on. For information on how to install and use ROS, please refer to [Documentation - ROS Wiki](#).

- If ACU is EAS2A01, please install ROS Indigo under Ubuntu 14.04_amd64.
- If ACU is EAXVA01, please install ROS Melodic under Ubuntu 18.04_amd64.

2.4.4 EcoSDK-S2

- If ACU is EAS2A01, please install EcoSDK-S2.
- If ACU is EAXVA01, please refer to **2.4.5 EcoSDK-XV**.

EcoSDK-S2 provides users with a complete application development environment, including:

- Cross-development toolchain: consists of a cross-compiler, a cross-connector, a cross-debugger, and other tools for application development.
- System roots: EcoSDK-S2 contains 2 system roots: one for the development host, which contains the cross-development toolchain and other tools; the other for the target, which is a complete root file system for the target, contains development kits with header files and libraries.
- Environment configuration: using the script provided by EcoSDK-S2, users can configure a cross-development environment on the development host.
- Analysis Tools: A variety of userspace tools for analyzing user applications on the target system.

EcoSDK-S2 provides application developers with all the necessary tools to write applications based on the Linux operating system and ROS. For details, please refer to EcoSDK-S2 User Manual.

2.4.5 EcoSDK-XV

- If ACU is EAS2A01, please refer to **2.4.4 EcoSDK-S2**.
- If ACU is EAXVA01, please install EcoSDK-XV.

EcoSDK-XV is a cross-development tool for EAXVA01. It provides users with a complete application development environment, including:

- Cross-development toolchain: consists of a cross-compiler, a cross-connector, a cross-debugger, and other tools for application development.
- System roots: EcoSDK-XV contains 2 system roots: one for the development host, which contains the cross-development toolchain and other tools; the other for the target, which is a complete root file system for the target, contains development kits with header files and libraries.
- Environment configuration: using the script provided by EcoSDK-XV, users can configure a cross-development environment on the development host.
- Analysis Tools: A variety of userspace tools for analyzing user applications on the target system.

EcoSDK-XV provides application developers with all the necessary tools to write applications based on the Linux operating system and ROS. For details, please refer to EcoSDK-XV User Manual.

2.4.6 MATLAB

MATLAB is a commercial mathematics software produced by MathWorks. MATLAB provides an advanced computing language and interactive environment for algorithm development, data visualization, data analysis, and numerical computing. In addition to common functions such as matrix operations, drawing functions/data images, MATLAB can also be used to create user interfaces and call programs written in other languages (including C, C++, Java, Python, and FORTRAN). MATLAB is not only used for numerical calculations, but also suitable for different fields of application, such as control system design and analysis, image processing, signal

processing and communication, financial modeling and analysis, with the help of many additional toolboxes. As integrated with MATLAB, Simulink provides a model-based development environment for system simulation, dynamic/embedded system development, etc. For information on how to install and use MATLAB, see [MATLAB Documentation - MathWorks](#). For information on how to use Simulink, see [Simulink Documentation - MathWorks](#).

The components you need to install when installing MATLAB are:

- MATLAB
- Simulink
- MATLAB Coder
- Simulink Coder
- Embedded Coder

We recommend you install the following components as well:

- Stateflow

2.4.7 OpenSSH

To make sure EcoCoder-AI can enable users to calibrate the model using Simulink external mode, the installation of OpenSSH is suggested. To install it, use the command below in Ubuntu:

```
$ sudo apt-get install openssh-client
```

2.4.8 Expect

To make sure EcoCoder-AI can enable users to calibrate the model using Simulink external mode, the installation of Expect is suggested. To install it, use the command below in Ubuntu:

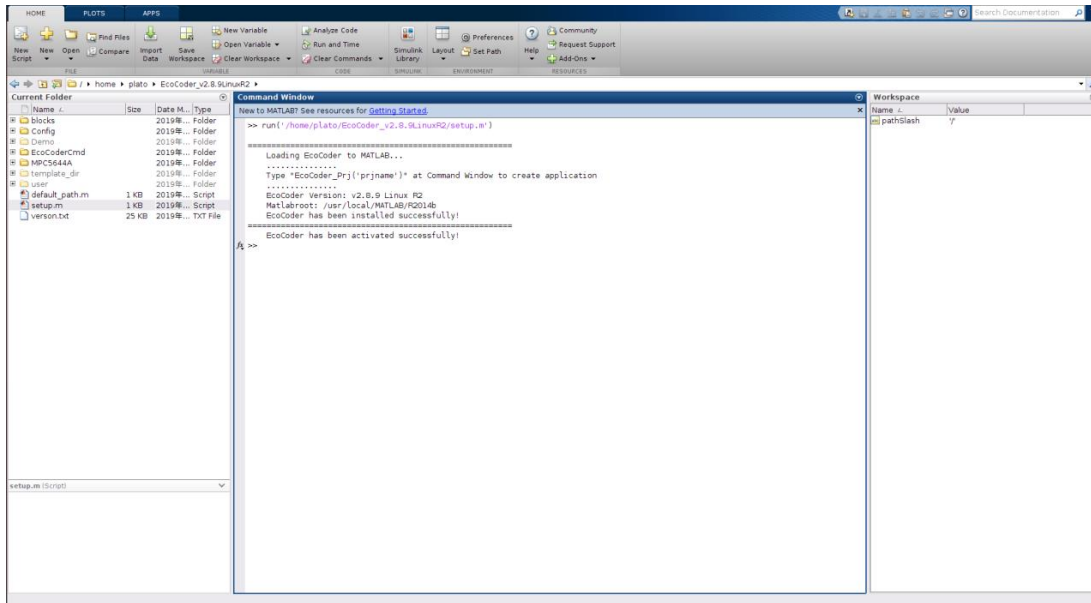
```
$ sudo apt-get install expect
```

2.4.9 Tcl

To make sure EcoCoder-AI can enable users to calibrate the model using Simulink external mode, the installation of Tcl is suggested. To install it, use the command below in Ubuntu:

```
$ sudo apt-get install tcl
```

2.5 Install EcoCoder-AI in Linux



A dongle is needed to use EcoCoder-AI. Please contact us to obtain a dongle. The dongle needs to be plugged in when using EcoCoder-AI.

2.5.1 Preparation

Extract the installer to the user directory. We'll take `EcoCoder_v2.8.9LinuxR2.zip` as an example.

```
$ mkdir -p ~/EcoCoder_v2.8.9LinuxR2
```

```
$ sudo unzip <path to package>/EcoCoder_v2.8.9LinuxR2.zip -d ~/
```

2.5.2 Installation

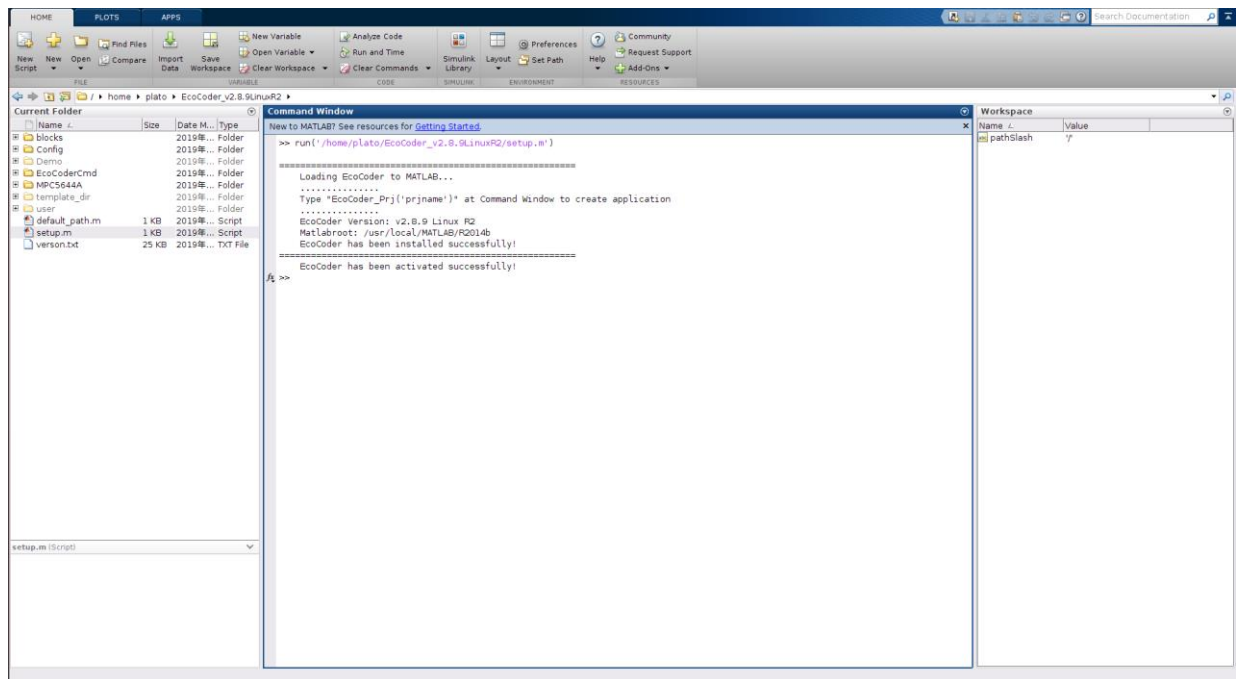
Switch to root and launch MATLAB.

```
$ sudo su root
```

```
# matlab
```

In MATLAB, change the working directory to where EcoCoder-AI is extracted to. In this example, it is `"/home/plato/EcoCoder_v2.8.9LinuxR2"`, drag the `setup.m` in this directory to the Command Window.

In the Command Window, "EcoCoder has been installed successfully!" and "EcoCoder has been activated successfully!" should appear as shown below, indicating that EcoCoder-AI has been installed and activated successfully.



Note: If you **relaunch** MATLAB, you need to repeat the installation process from the beginning.

2.6 Install EcoCoder-AI in Windows

2.6.1 Preparation

Extract the installer to the user directory.

2.6.2 Installation

Launch MATLAB. Change the MATLAB working directory to where EcoCoder-AI is extracted to. Drag the `setup.m` in this directory to the Command Window.

In the Command Window, "EcoCoder has been installed successfully!" and "EcoCoder has been activated successfully!" should appear, indicating that EcoCoder-AI has been installed and activated successfully.

Chapter 3 Quick Start

3.1 Quick Start in Linux

There is a demo project in EcoCoder-AI installation directory, you can get started with EcoCoder-AI with the help of the demo.

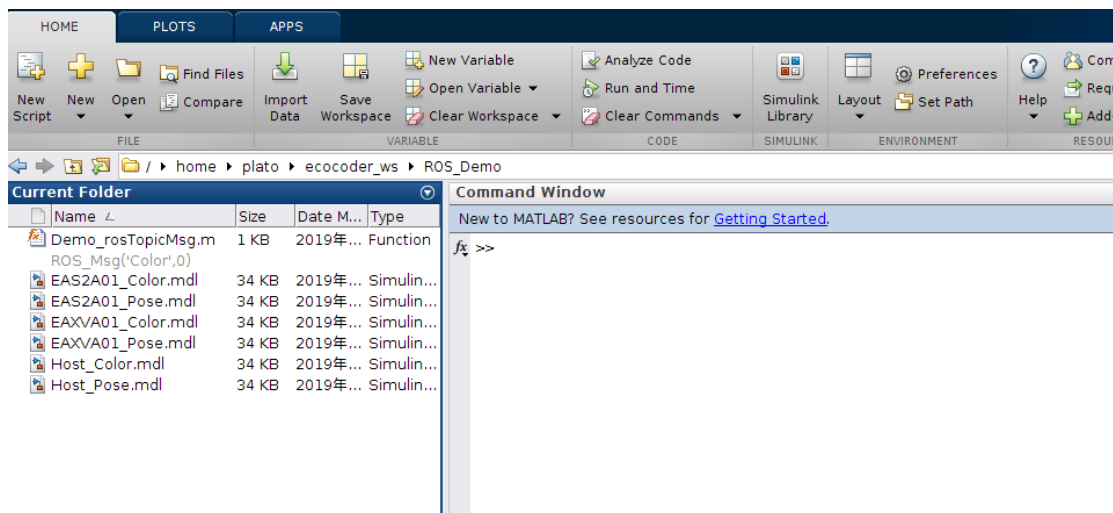
Take EAXVA01 as an example.

3.1.1 Create Working Directory

Create an empty directory, copy the demo project into this directory, and set `ROS_Demo` as MATLAB working directory.

```
$ mkdir $HOME/ecocoder_ws
```

```
$ sudo cp -r <path to installation directory>/Demo/ROS_Demo $HOME/ecocoder_ws/
```

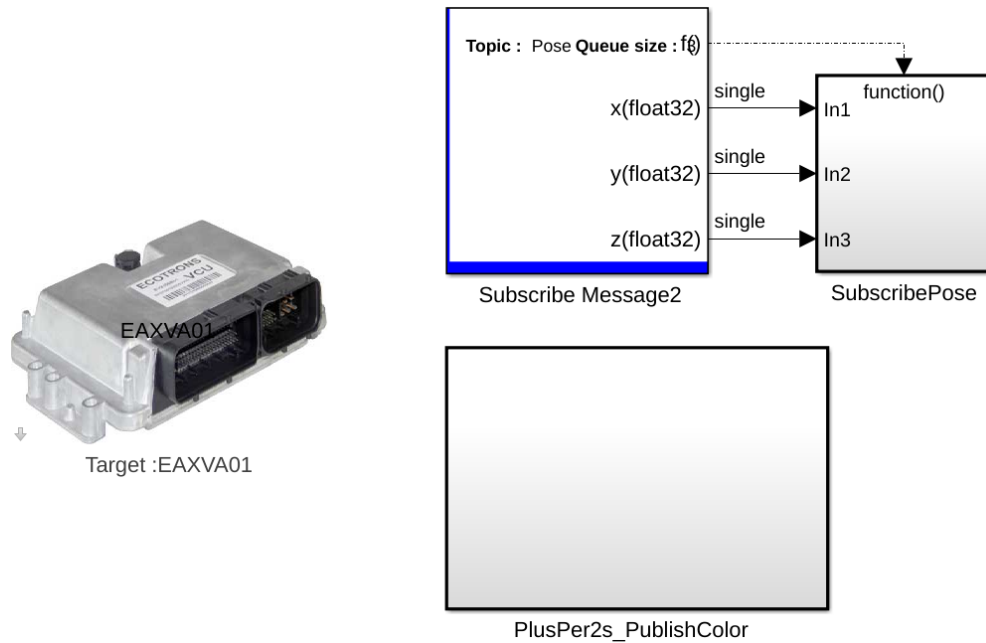
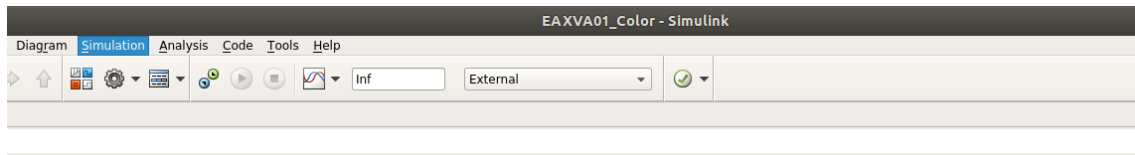


3.1.2 Open the Model

In `ROS_Demo` directory, double click `EAXVA01_Color.mdl` to open the model as shown below. The model is designed based on ROS and it can run on EAXVA01. The main functions of the model are:

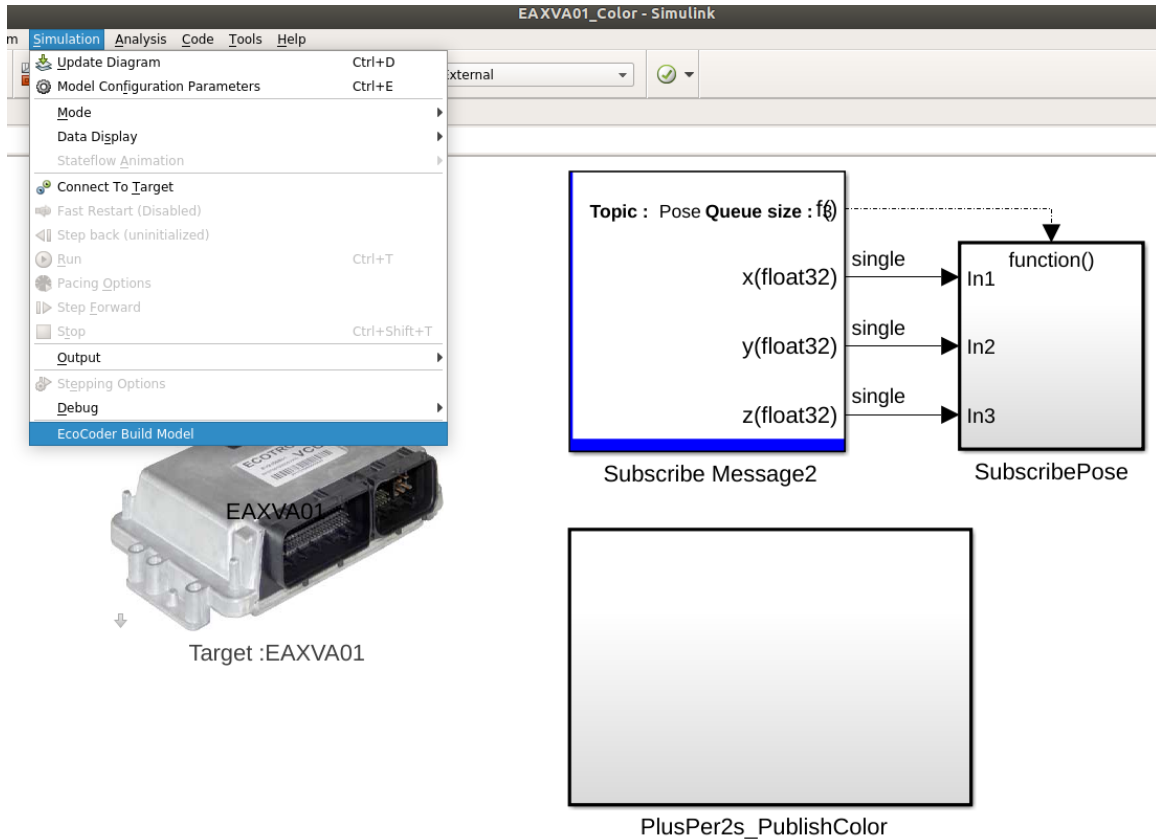
- To subscribe to the “Pose” topic, obtain the values of x, y, and z in the “Pose” message, and broadcast them through the console.

- To post a message to “Color” topic every 2 seconds, with the values of a, b, c and d in the message incremented by one, and broadcast the new values through the console.



3.1.3 Build the Model

The executable files can be generated by clicking **Simulation** in the menu, then clicking **EcoCoder Build Model**, as shown below. After the building process, a directory called "Target_out" will be generated in the current folder, with a file named "EAXVA01_Color_0" in it. EAXVA01_Color_0 is an executable program based on ROS, which can run on an EAXVA01 device.



When you build the model, if the EAXVA01 device is not connected with the host (in this case, Linux) through ethernet or its IP address and port are not set up in Simulink (you will see how to set it up in the following chapter), you will get an error alert, saying that the target controller is not connected. You may ignore this error right now.

3.1.4 Build Other Models

Open "EAXVA01_Pose.mdl" as step 3.1.2. The model is designed based ROS and it can run on EAXVA01. The main functions of the model are:

- To subscribe to "Color" topic, obtain the values of a, b, c and d in the "Color" message, and broadcast these values in the console.
- Post a message to the "Pose" topic every 2 seconds, with the values of x, y and z in the "Pose" message incremented by one and broadcast these values through console.

Build the model as step 3.1.3, you will get "EAXVA01_Pose_0".

The other two models in this demo, "Host_Color.mdl" and "Host_Pose.mdl", have the same functions as "EAXVA01_Color.mdl" and "EAXVA01_Pose.mdl" respectively. But the executable files these two models generate are meant to run on PC.

"EAS2A01_Color.mdl" and "EAS2A01_Pose.mdl" have the same functions as "EAXVA01_Color.mdl" and "EAXVA01_Pose.mdl" respectively. But the executable files these two models generate are meant to run on EAS2A01.

3.1.5 Run on PC

3.1.5.1 EAXVA01

Open a terminal, go to "Target_out" directory, initialize ROS runtime environment, and launch roscore. If you see the texts as below, roscore is launched successfully.

```
$ cd $HOME/ecocoder_ws/ROS_Demo/Target_out/
```

```
$ source /opt/ros/melodic/setup.sh
```

```
$ roscore
```

```
plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ source /opt/ros/melodic/setup.sh
plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ roscore
... logging to /home/plato/.ros/log/98bdfd40-8b31-11e9-b58f-5ce0c58a9cd3/roslauch-T450s-18835.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://T450s:41501/
ros_comm version 1.14.3

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.3

NODES

auto-starting new master
process[rosmaster]: started with pid [18846]
ROS_MASTER_URI=http://T450s:11311/

setting /run_id to 98bdfd40-8b31-11e9-b58f-5ce0c58a9cd3
process[rosout-1]: started with pid [18857]
started core service [/rosout]
```

Open a new terminal, initialize ROS runtime environment, and launch “Host_Pose_0”.

```
$ source /opt/ros/melodic/setup.sh
```

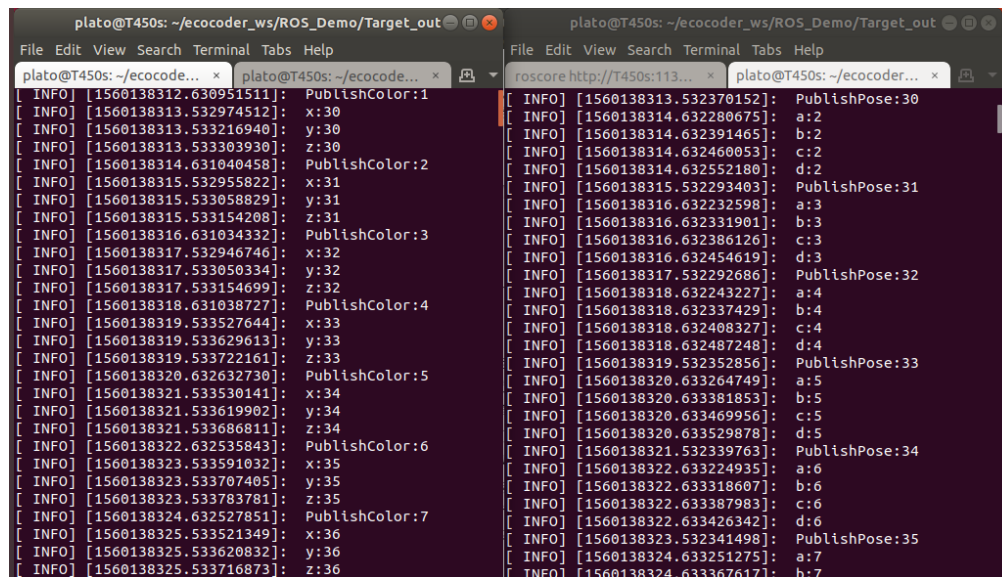
```
$ ./Host_Pose_0
```

Open a new terminal, initialize ROS runtime environment, and launch “Host_Color_0”

```
$ source /opt/ros/melodic/setup.sh
```

```
$ ./Host_Color_0
```

You can see that Host_Pose is posting messages to topic “Pose” and the messages are subscribed and received by Host_Color; Host_Color is posting messages to topic “Color” and the messages are subscribed and received by Host_Pose.



```
plato@T450s: ~/ecocoder_ws/ROS_Demo/Target_out
[ INFO ] [1560138312.630951511]: PublishColor:1
[ INFO ] [1560138313.532974512]: x:30
[ INFO ] [1560138313.533216940]: y:30
[ INFO ] [1560138313.533303930]: z:30
[ INFO ] [1560138314.631040458]: PublishColor:2
[ INFO ] [1560138315.532955822]: x:31
[ INFO ] [1560138315.533058829]: y:31
[ INFO ] [1560138315.533154208]: z:31
[ INFO ] [1560138316.631034332]: PublishColor:3
[ INFO ] [1560138317.532946746]: x:32
[ INFO ] [1560138317.533050334]: y:32
[ INFO ] [1560138317.533154699]: z:32
[ INFO ] [1560138318.631038727]: PublishColor:4
[ INFO ] [1560138319.533527644]: x:33
[ INFO ] [1560138319.533629613]: y:33
[ INFO ] [1560138319.533722161]: z:33
[ INFO ] [1560138320.632632730]: PublishColor:5
[ INFO ] [1560138321.533530141]: x:34
[ INFO ] [1560138321.533619902]: y:34
[ INFO ] [1560138321.533686811]: z:34
[ INFO ] [1560138322.632535843]: PublishColor:6
[ INFO ] [1560138323.533591032]: x:35
[ INFO ] [1560138323.533707405]: y:35
[ INFO ] [1560138323.533783781]: z:35
[ INFO ] [1560138324.632527851]: PublishColor:7
[ INFO ] [1560138325.533521349]: x:36
[ INFO ] [1560138325.533620832]: y:36
[ INFO ] [1560138325.533716873]: z:36

plato@T450s: ~/ecocoder_ws/ROS_Demo/Target_out
[ INFO ] [1560138313.532370152]: PublishPose:30
[ INFO ] [1560138314.632280675]: a:2
[ INFO ] [1560138314.632391465]: b:2
[ INFO ] [1560138314.632460053]: c:2
[ INFO ] [1560138314.632552180]: d:2
[ INFO ] [1560138315.532293403]: PublishPose:31
[ INFO ] [1560138316.632232598]: a:3
[ INFO ] [1560138316.632331901]: b:3
[ INFO ] [1560138316.632386126]: c:3
[ INFO ] [1560138316.632454619]: d:3
[ INFO ] [1560138317.532292686]: PublishPose:32
[ INFO ] [1560138318.632243227]: a:4
[ INFO ] [1560138318.632337429]: b:4
[ INFO ] [1560138318.632408327]: c:4
[ INFO ] [1560138318.632487248]: d:4
[ INFO ] [1560138319.532352856]: PublishPose:33
[ INFO ] [1560138320.633264749]: a:5
[ INFO ] [1560138320.633381853]: b:5
[ INFO ] [1560138320.633469956]: c:5
[ INFO ] [1560138320.633529878]: d:5
[ INFO ] [1560138321.532339763]: PublishPose:34
[ INFO ] [1560138322.633224935]: a:6
[ INFO ] [1560138322.633318607]: b:6
[ INFO ] [1560138322.633387983]: c:6
[ INFO ] [1560138322.633426342]: d:6
[ INFO ] [1560138323.532341498]: PublishPose:35
[ INFO ] [1560138324.633251275]: a:7
[ INFO ] [1560138324.633367617]: b:7
```

3.1.5.2 EAS2A01

Open a terminal, go to “Target_out” directory, initialize ROS runtime environment, and launch roscore. If you see the texts as below, roscore is launched successfully.

```
$ cd $HOME/ecocoder_ws/ROS_Demo/Target_out/
```

```
$ source /opt/ros/indigo/setup.sh
```

```
$ roscore
```

```
roscore http://jock-T420:11311/
root@jock-T420:/home/jock x roscore http://jock-T420:11311/ x
jock@jock-T420:~/ecocoder_ws/ROS_Demo/Target_out$ source /opt/ros/indigo/setup.s
h
jock@jock-T420:~/ecocoder_ws/ROS_Demo/Target_out$ roscore
... logging to /home/jock/.ros/log/b632fa9a-6b17-11e9-9f9a-8c705a663674/roslaun
h-jock-T420-11661.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://jock-T420:45623/
ros_comm version 1.11.21

SUMMARY
=====

PARAMETERS
* /rostdistro: indigo
* /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [11673]
ROS_MASTER_URI=http://jock-T420:11311/

setting /run_id to b632fa9a-6b17-11e9-9f9a-8c705a663674
process[rosout-1]: started with pid [11686]
started core service [/rosout]
```

Open a new terminal, initialize ROS runtime environment, and launch “Host_Pose_0”.

```
$ source /opt/ros/indigo/setup.sh
```

```
$ ./Host_Pose_0
```

Open a new terminal, initialize ROS runtime environment, and launch “Host_Color_0”

```
$ source /opt/ros/indigo/setup.sh
```

```
$ ./Host_Color_0
```

You can see that Host_Pose is posting messages to topic “Pose” and the messages are subscribed and received by Host_Color; Host_Color is posting messages to topic “Color” and the messages are subscribed and received by Host_Pose.

```

root@jock-T420: ~/ecocoder_ws/ROS_Demo/Target_out
[INFO] [1556609383.621586206]: b: 34
[INFO] [1556609383.621696412]: c: 34
[INFO] [1556609383.622092539]: d: 34
[INFO] [1556609384.278436174]: PublishPose:27
[INFO] [1556609385.621370620]: a: 35
[INFO] [1556609385.621474914]: b: 35
[INFO] [1556609385.621564961]: c: 35
[INFO] [1556609385.621649836]: d: 35
[INFO] [1556609386.278427715]: PublishPose:28
[INFO] [1556609387.628811041]: a: 36
[INFO] [1556609387.628887216]: b: 36
[INFO] [1556609387.628925020]: c: 36
[INFO] [1556609387.628973934]: d: 36
[INFO] [1556609388.283816092]: PublishPose:29
[INFO] [1556609389.628797553]: a: 37
[INFO] [1556609389.628905751]: b: 37
[INFO] [1556609389.628998375]: c: 37
[INFO] [1556609389.629178570]: d: 37
[INFO] [1556609390.283910221]: PublishPose:30

root@jock-T420: ~/ecocoder_ws/ROS_Demo/Target_out
[INFO] [1556609404.374321486]: x:37
[INFO] [1556609404.374832929]: y:37
[INFO] [1556609404.374956801]: z:37
[INFO] [1556609405.699359070]: PublishColor:45
[INFO] [1556609406.374316984]: x:38
[INFO] [1556609406.374437188]: y:38
[INFO] [1556609406.374655831]: z:38
[INFO] [1556609407.718019669]: PublishColor:46
[INFO] [1556609408.388031170]: x:39
[INFO] [1556609408.388541979]: y:39
[INFO] [1556609408.388655792]: z:39
[INFO] [1556609409.720599639]: PublishColor:47
[INFO] [1556609410.397987130]: x:40
[INFO] [1556609410.398035241]: y:40
[INFO] [1556609410.398080464]: z:40
[INFO] [1556609411.723061459]: PublishColor:48
[INFO] [1556609412.397996736]: x:41
[INFO] [1556609412.398234056]: y:41
[INFO] [1556609412.398445648]: z:41

```

3.1.6 Run on ACU

3.1.6.1 EAXVA01

Prepare an EAXVA01 device and obtain the IP address of EAXVA01 through the serial terminal. In the following instructions, <target_ip> represents the IP address of the target controller EAXVA01. As with how to obtain the IP address of the device, please refer to EAXVA01 Datasheet.

Open a terminal, go to the directory "Target_out", and transmit "EAXVA01_Pose_0" and "EAXVA01_Color_0" to the EAXVA01 device. Afterwards, log in remotely to the EAXVA01 device, initialize the ROS runtime environment, and launch roscore.

```
$ cd $HOME/ecocoder_ws/ROS_Demo/Target_out/
```

```
$ scp EAXVA01_Pose_0 EAXVA01_Color_0 nvidia@<target_ip>:/home/nvidia
```

The password of EAXVA01 by default is nvidia.

```
$ ssh nvidia@<target_ip>
```

```
# source /opt/ros/melodic/setup.sh
```

```
# roscore
```

```

plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ scp EAXVA01_Color_0 EAXVA01_Pose_0 nvidia@192.168.1.123:/home/nvidia
nvidia@192.168.1.123's password:
EAXVA01_Color_0          100% 1157KB  10.7MB/s  00:00
EAXVA01_Pose_0          100% 1181KB  10.9MB/s  00:00
plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ ssh nvidia@192.168.1.123
nvidia@192.168.1.123's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.9.140 aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with

   snap install microk8s --classic
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

28 packages can be updated.
1 update is a security update.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Thu Jun  6 17:37:40 2019 from 192.168.1.120

```

```

nvidia@EAXVA01:~$ source /opt/ros/melodic/setup.sh
nvidia@EAXVA01:~$ roscore
... logging to /home/nvidia/.ros/log/bfe12216-883f-11e9-a3f5-00044bcb90b4/roslaunch-EAXVA01-7650.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://EAXVA01:43757/
ros_comm version 1.14.3

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.3

NODES

auto-starting new master
process[master]: started with pid [7660]
ROS_MASTER_URI=http://EAXVA01:11311/

setting /run_id to bfe12216-883f-11e9-a3f5-00044bcb90b4
process[rosout-1]: started with pid [7671]
started core service [/rosout]

```

Open a new terminal, remotely log in to the EAXVA01 device, initialize the ROS runtime environment, and start "EAXVA01_Pose_0".

```
$ ssh nvidia@<target_ip>
```

The password of EAXVA01 by default is nvidia.

```
# source /opt/ros/melodic/setup.sh
```

```
# ./EAXVA01_Pose_0
```

Open a new terminal, remotely log in to the EAXVA01 device, initialize the ROS runtime environment, and start "EAXVA01_Color_0".

```
$ ssh nvidia@<target_ip>
```

The password of EAXVA01 by default is nvidia.

```
$ source /opt/ros/melodic/setup.sh
```

```
$/EAXVA01_Color_0
```

```
plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ ssh nvidia@192.168.1.123
nvidia@192.168.1.123's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.9.140 aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with
   snap install microk8s --classic
 This system has been minimized by removing packages and content that are
 not required on a system that users do not log into.
 To restore this content, you can run the 'unminimize' command.
 28 packages can be updated.
 1 update is a security update.
 Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check
 your Internet connection or proxy settings

Last login: Thu Jun  6 17:26:04 2019 from 192.168.1.120
nvidia@EAXVA01:~$ source /opt/ros/melodic/setup.sh
nvidia@EAXVA01:~$ ./EAXVA01_Color_0
[ INFO] [1559813919.551246176]: PublishColor:1
[ INFO] [1559813921.562289168]: PublishColor:2
[ INFO] [1559813921.718359792]: x:3
[ INFO] [1559813921.719567728]: y:3
[ INFO] [1559813921.720668880]: z:3
[ INFO] [1559813923.565345136]: PublishColor:3
[ INFO] [1559813923.718486032]: x:4
[ INFO] [1559813923.719246944]: y:4
[ INFO] [1559813923.719789456]: z:4
[ INFO] [1559813925.567839660]: PublishColor:4
[ INFO] [1559813925.723314896]: x:5
[ INFO] [1559813925.726818096]: y:5
[ INFO] [1559813925.726703504]: z:5
[ INFO] [1559813927.573187920]: PublishColor:5
[ INFO] [1559813927.727148976]: x:6
[ INFO] [1559813927.728092000]: y:6

plato@T450s:~/ecocoder_ws/ROS_Demo/Target_out$ ssh nvidia@192.168.1.123
nvidia@192.168.1.123's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.9.140 aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with
   snap install microk8s --classic
 This system has been minimized by removing packages and content that are
 not required on a system that users do not log into.
 To restore this content, you can run the 'unminimize' command.
 28 packages can be updated.
 1 update is a security update.
 Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check
 your Internet connection or proxy settings

Last login: Thu Jun  6 17:37:28 2019 from 192.168.1.120
nvidia@EAXVA01:~$ source /opt/ros/melodic/setup.sh
nvidia@EAXVA01:~$ ./EAXVA01_Pose_0
[ INFO] [1559813917.697596976]: PublishPose:1
[ INFO] [1559813919.711695472]: PublishPose:2
[ INFO] [1559813921.564331728]: a:2
[ INFO] [1559813921.565072208]: b:2
[ INFO] [1559813921.565235088]: c:2
[ INFO] [1559813921.565358384]: d:2
[ INFO] [1559813921.716504304]: PublishPose:3
[ INFO] [1559813923.567278960]: a:3
[ INFO] [1559813923.567872112]: b:3
[ INFO] [1559813923.568451920]: c:3
[ INFO] [1559813923.568644144]: d:3
[ INFO] [1559813923.718532272]: PublishPose:4
[ INFO] [1559813925.569874608]: a:4
[ INFO] [1559813925.570958128]: b:4
[ INFO] [1559813925.571674832]: c:4
[ INFO] [1559813925.572546000]: d:4
```

You should see the same results as on PC. Host_Pose is posting messages to topic "Pose" and the messages are subscribed and received by Host_Color; Host_Color is posting messages to topic "Color" and the messages are subscribed and received by Host_Pose.

3.1.6.2 EAS2A01

Prepare an EAS2A01 device and obtain the IP address of EAS2A01 through the serial terminal. In the following instructions, <target_ip> represents the IP address of the target controller EAS2A01. As with how to obtain the IP address of the device, please refer to EAS2A01 Datasheet.

Open a terminal, go to the directory "Target_out", and transmit "EAS2A01_Pose_0" and "EAS2A01_Color_0" to the EAS2A01 device. Afterwards, log in remotely to the EAS2A01 device, initialize the ROS runtime environment, and launch roscore.

```
$ cd $HOME/ecocoder_ws/ROS_Demo/Target_out/
```

```
$ scp EAS2A01_Pose_0 EAS2A01_Color_0 root@<target_ip>:/home/root
```

```
$ telnet <target_ip>
```

```
# source /opt/ros/indigo/setup.sh
```

```
# roscore
```

Open a new terminal, remotely log in to the EAS2A01 device, initialize the ROS runtime environment, and start "EAS2A01_Pose_0".

```
$ telnet <target_ip>
```

```
# source /opt/ros/indigo/setup.sh
```

```
# ./EAS2A01_Pose_0
```

Open a new terminal, remotely log in to the EAS2A01 device, initialize the ROS runtime environment, and start "EAS2A01_Color_0".

```
$ telnet <target_ip>
```

```
$ source /opt/ros/indigo/setup.sh
```

```
$ ./EAS2A01_Color_0
```

You should see the same results as on PC. Host_Pose is posting messages to topic "Pose" and the messages are subscribed and received by Host_Color; Host_Color is posting messages to topic "Color" and the messages are subscribed and received by Host_Pose.

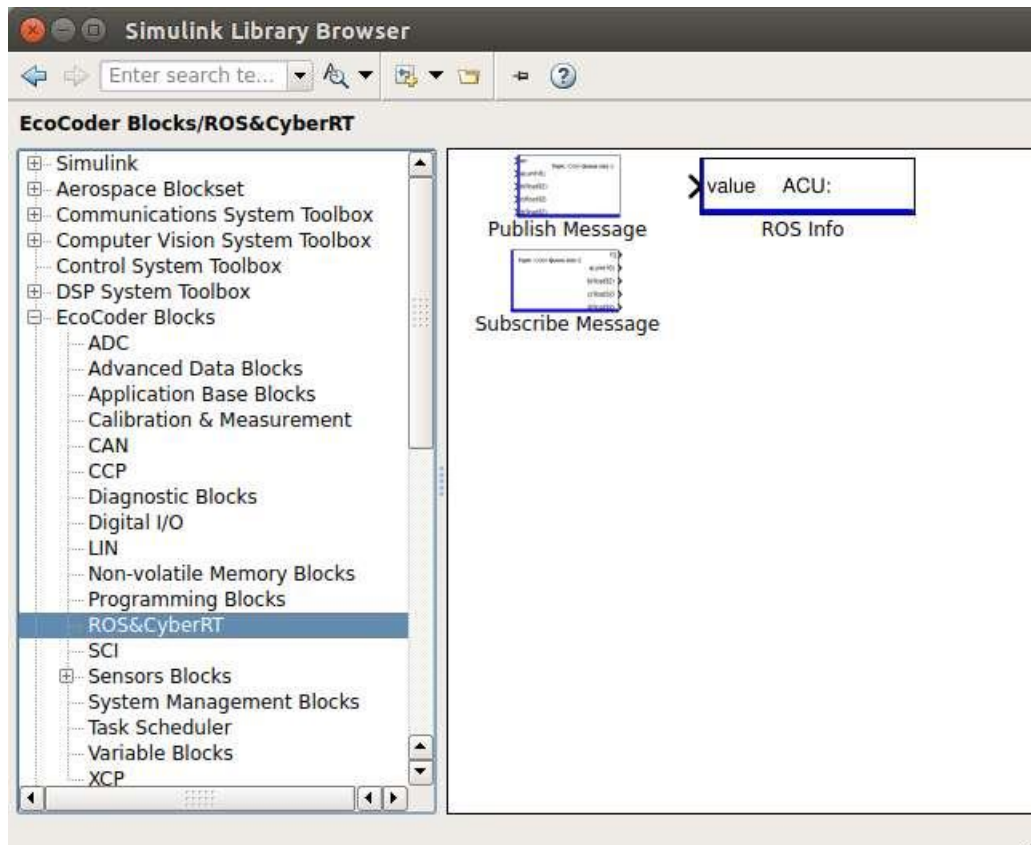
3.2 Quick Start in Windows

It's almost the same to design a model, open a demo model in the Windows platform, compared with that in Linux. Demo models are in the Demo folder in the EcoCoder directory.

Cross-compile is not supported in the Windows platform. Therefore, you need to copy the generated code to an x86-based Linux and cross-compile the code, or you can copy the generated code to the target and compile them there. Please refer to EcoSDK User Manual.

Chapter 4 Library

After installing the EcoCoder-AI, many extension libraries will be added to the Simulink library, which can be used to access the hardware interface developed by Ecotron, or connect the Simulink model to the ROS system. Click the “Simulink Library” button in the MATLAB interface, and the dialog box shown below will pop up. You can see that “EcoCoder Blocks” has been added.



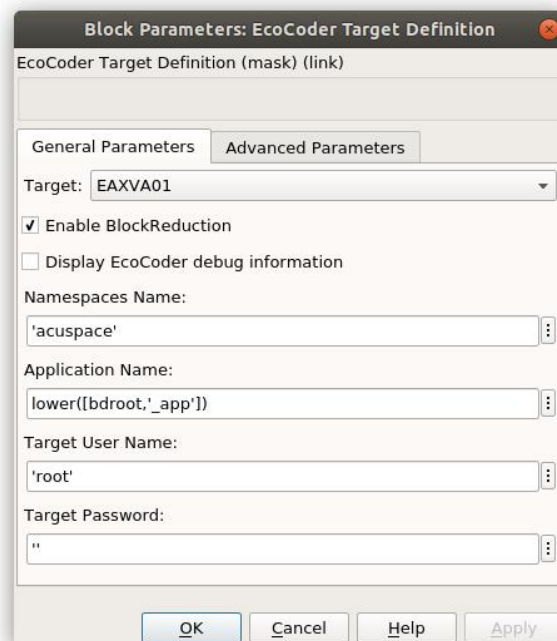
4.1 EcoCoder Target Definition

The EcoCoder Target Definition is used to define the target controller.

This module must be added, otherwise an error will occur during compilation.

- **Parameters:**
 - General Parameters:
 - Target: you can choose your target hardware (Host is PC).
 - Enable BlockReduction: enable automatic optimize code

- Display EcoCoder debug information: checking this will display debug information
 - Namespaces Name: enter the name of ROS Package here
 - Application Name: enter the name of ROS Node here
 - Target User Name: user name of the target in external mode, used for SSH login
 - Target Password: corresponding password for the username above in external mode
- Advanced Parameters: under development



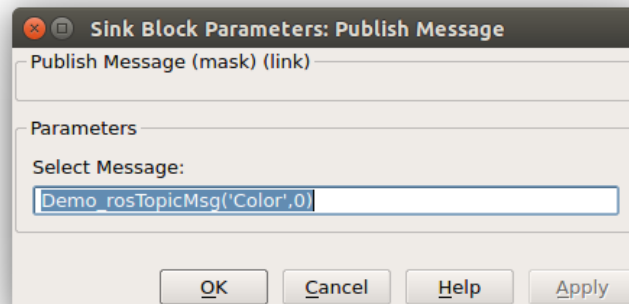
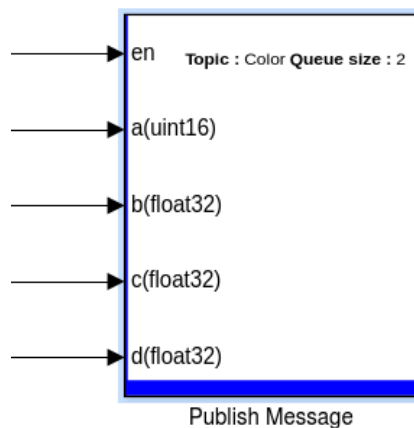
4.2 ROS&CyberRT

4.2.1 Publish Message Module

The Publish Message Module is based on ROS. It is used for posting a message to a topic. The specified m file can be selected according to the input parameter, then it will define a topic, determine the parameter of the message according to the name of the topic and the content of the m file, and change the input interface of the module.

For example, if you double-click the module, a dialog box will pop up. Change the contents of the input box as shown in the picture, then click the “Apply” button. The module will first search for the file named “Demo_rosTopicMsg.m” in the current working directory. If it can’t find the file, it will search again in "<path of installation directory>/blocks/My_libs". If the file is found, it will read the contents of the file and search for the keyword "Color", define a topic named "Color" and a message named "Color"; according to the "Color" keyword Content, determine the parameter name and parameter type of the "Color" message, and change the input interface of the module according to the parameters of the message. For information on how to define ROS topics via m files, please refer to [Define a ROS Topic](#).

- Parameter:
 - en: enables or disables the module, 1 is enable, 0 is disable
 - Select Message: assign the m file this module depends on and the name of topic and message it will create

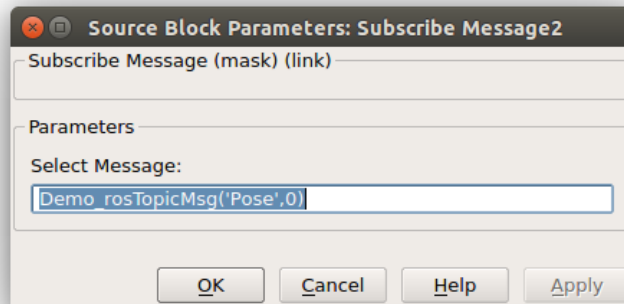
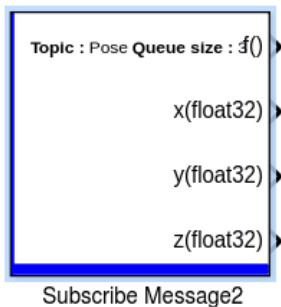


4.2.2 Subscribe Message Module

The Subscribe Message Module is based on ROS and subscribes to a message on the topic and calls a subfunction that processes the value of each parameter in the message. The specified m file can be selected according to the input parameter, then it will define a message and subscribe to the topic of the same name, and change the output interface parameter of the module according to the content of the m file and the name of the topic.

For example, if you double-click the module, a dialog box will pop up. Change the contents of the input box as shown in the figure, then click the “Apply” button. The module will first search for the file named “Demo_rosTopicMsg.m” in the current working directory. If the search fails, it will search “<path of installation directory>/blocks/My_libs” for the file. If the file is found, it will read the contents of the file, search for the “Pose” keyword, and define a message named “Pose” and subscribe to the message from the topic named “Pose”; the following content of the Pose keyword determines the parameter name and parameter type of the “Pose” message, and changes the output interface of the module according to the parameters of the message. Then the module will output the values of the parameters in the subscribed message to other modules through these interfaces and call the subfunction connected by f(). For information on how to define ROS topics via m files, please refer to [Define a ROS Topic](#).

- Parameter:
 - f(): the module it links will be called after subscribing the message
 - Select Message: assign the m file this module depends on and the name of topic and message it will create

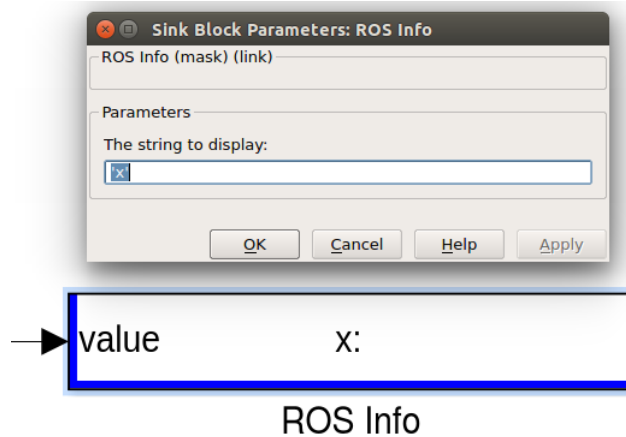


4.2.3 ROS Info Module

The ROS Info Module outputs the value of a variable through the console. For example, if you double-click the block, a dialog box will pop up, enter a character or string (usually the name of the variable to be connected to the input interface of this block), the block will output “[INFO]

[1556623925.873084964] in the console: x : \$value", where "\$value" represents the value of the variable connected to the input interface of the block.

- Parameter:
 - The string to display: the name of the variable that links to input interface, which will show up in the console

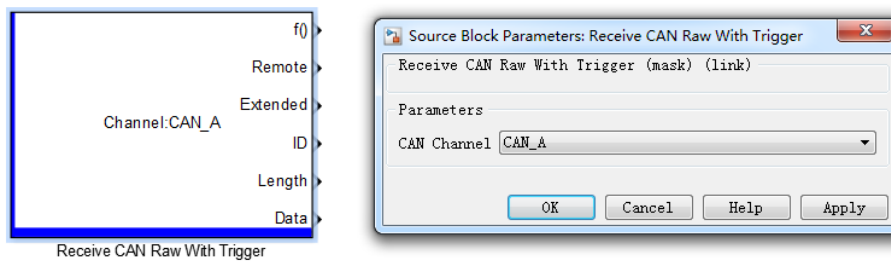


4.3 Socket CAN Module

This is the CAN communication module, it calls the socket CAN interface of Linux, to realize CAN message sending and receiving. And it supports to load the dbc file directly, for rapid development of CAN system design.

4.3.1 Receive CAN Raw with Trigger

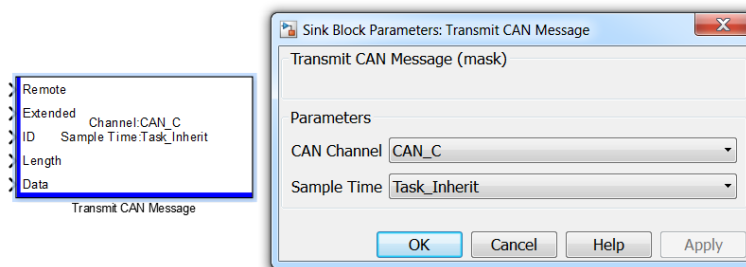
This block uses trigger type for receiving the CAN message, when ACU receives the CAN message, it will be triggered for processing the task through the socket CAN interface.



- Parameter:
 - CAN Channel: define the CAN channel for this block
- Output
 - f(): the module it links will be called after subscribing the message
 - Remote: message type, 1 for remote frame, 0 for data frame
 - Extended: message type, 1 for extended frame, 0 for standard frame
 - ID: message ID
 - Length: message length
 - Data: message data

4.3.2 Transmit CAN Message

This block is used to send CAN messages. After calling this module, the socket CAN interface of ACU will be called to send the CAN message.

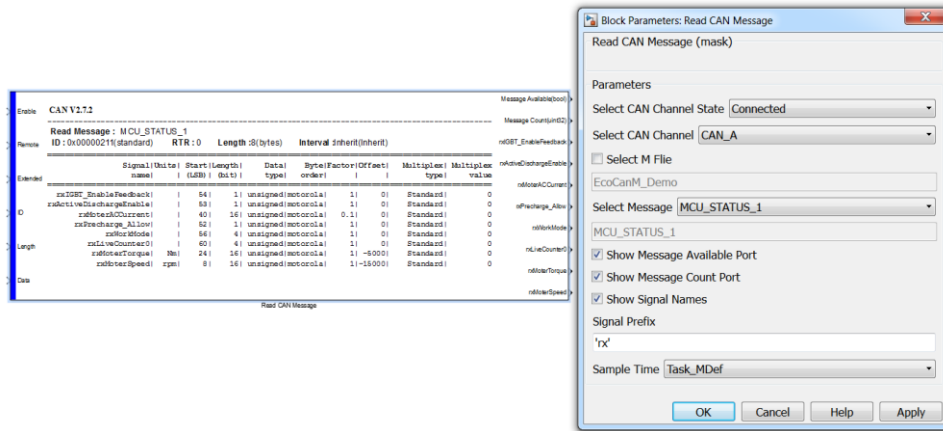


- Parameter:
 - CAN Channel: define the CAN channel for this block
 - Sample Time: define the sampling time for this block
- Output
 - Remote: message type, 1 for remote frame, 0 for data frame
 - Extended: message type, 1 for extended frame, 0 for standard frame
 - ID: message ID
 - Length: message length
 - Data: message data

4.3.3 Read CAN Message

This module is used to parse CAN messages. The CAN message will be parsed into signals, and the signal type will be automatically inherited backwards.

The module can't load dbc files directly, but it can load m files converted from dbc files by using EcoCAN. (free SW developed by Ecotron)

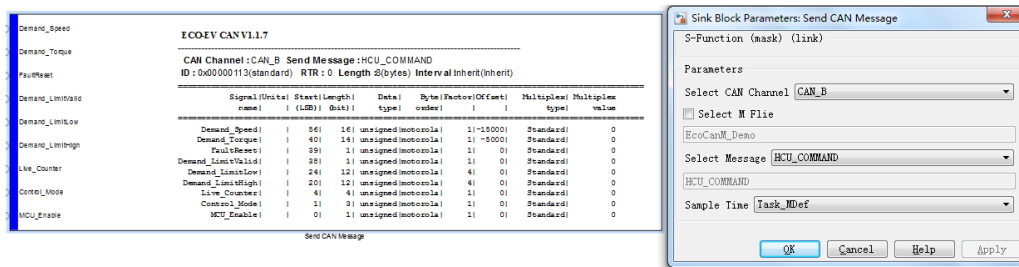


- Parameter:
 - Select CAN Channel State: for ACU customers, please select 'Disconnected'
 - Select CAN Channel: define the CAN channel for this block
 - Select M file: load that m file which is converted from dbc file here, the m file has to be located in the same folder with the model
 - Select Message: select the message here
 - Show Message Available port: 1 means it is receiving the message
 - Show Message Count Port: each time a new message is received, the counter is incremented by 1
 - Show Signal Names: when enabled, the names of the signals will be displayed on the output line
 - Signal prefix: the variable prefix on the signal line
 - Sample time: define the sampling time for this block
- Input (The inputs will appear when you set 'select CAN Channel State' to 'disconnected')
 - Enable: Enable this block

- Remote: message type, 1 for remote frame, 0 for data frame
- Extended: message type, 1 for extended frame, 0 for standard frame
- ID: message ID
- Length: message length
- Data: message data
- Output
 - For each signal, the value of the signal is the physical value

4.3.4 Send CAN Message

This module is used to package CAN messages. CAN signals will be packaged and sent, the signal type is inherited, the module can't load dbc files directly, but it can load m files converted from dbc files by using EcoCAN. (free SW developed by Ecotron)



- Parameter:
 - Select CAN Channel State: Select the channel state with the 'Connected' and 'Disconnected' options.
When 'Connected' is selected, there is no output port, and the message information will be directly sent to the selected channel.
When 'Disconnected' is selected, the module appears with output ports from which the message is exported.
 - Select CAN Channel: define the CAN channel for this block
 - Select M file: load that m file which is converted from dbc file here, the m file has to be located in the same folder with the model
 - Select Message: select the message here
 - Sample time: define the sampling time for this block

- Input
 - For each signal, the value of the signal should be the physical value

- Output (The outputs will appear when you set 'select CAN Channel State' to 'disconnected')
 - Remote: message type, 1 for remote frame, 0 for data frame
 - Extended: message type, 1 for extended frame, 0 for standard frame
 - ID: message ID
 - Length: message length
 - Data: message data

Chapter 5 Define ROS Topic

We will use an example to show how to define a ROS topic. You should create a m file under the project directory, and name it “Demo_rosTopicMsg.m”. Then edit the file like below:

```
function ROSMsg = ROS_TopicMsg_Demo(msgname,type)
    if(255==type)
        ROSMsg = struct;
        ROSMsg.num=2;
        ROSMsg.list= cell(1, ROSMsg.num);
        ROSMsg.list{1}='Color';
        ROSMsg.list{2}='Pose';
    else
        ROSMsg = struct;
        switch msgname
            case 'Color'
                ROSMsg = struct;
                ROSMsg.topic_name = 'Color';
                ROSMsg.msg_queue_size = 2;

                ROSMsg.fields{1}.data_name = 'a';
                ROSMsg.fields{1}.data_type = 'uint16';

                ROSMsg.fields{2}.data_name = 'b';
                ROSMsg.fields{2}.data_type = 'float32';

                ROSMsg.fields{3}.data_name = 'c';
                ROSMsg.fields{3}.data_type = 'float32';

                ROSMsg.fields{4}.data_name = 'd';
                ROSMsg.fields{4}.data_type = 'float32';
            case 'Pose'
                ROSMsg = struct;
                ROSMsg.topic_name = 'Pose';
                ROSMsg.msg_queue_size =3;

                ROSMsg.fields{1}.data_name = 'x';
                ROSMsg.fields{1}.data_type = 'float32';

                ROSMsg.fields{2}.data_name = 'y';
                ROSMsg.fields{2}.data_type = 'float32';

                ROSMsg.fields{3}.data_name = 'z';
                ROSMsg.fields{3}.data_type = 'float32';
        end
    end
end
```

In this file, we define two topics and messages both named “Color” and “Pose”. The buffer of topic “Color” can store up to 2 “Color” messages, while that of “Pose” is 3. Message “Color” has four parameters: uint16 a, float32 b, float32 c, float32 d. Message “Pose” has 3 parameters: float32 x, float32 y, float32 z.

Chapter 6 External Mode

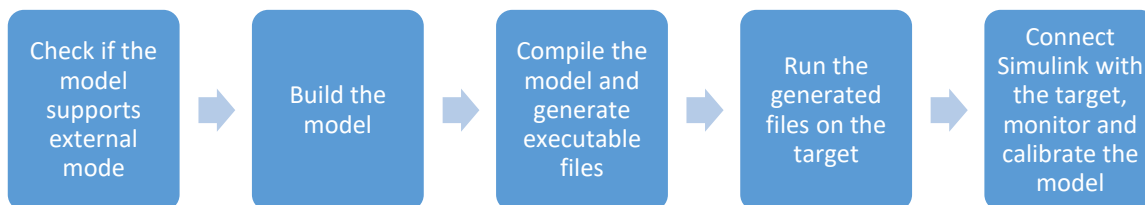
External mode enables the communication between two systems, namely the host and the target. The host is the computer that you run MATLAB and Simulink on. The target is the ACU hardware on which we will run the executable files.

The host will require the target to receive the changed model parameters or send out the messages to the host, by sending request messages to the target. The target will take action according to the requests from the host. The external mode is based on the C/S structure, wherein Simulink is the client and the target is the server.

Using external mode, after you change the model parameters, Simulink will send the changed parameters to the target in real-time, which enables the developers to do on-the-fly calibration. Thanks to the external mode, You can also monitor the input and the output of all the blocks and subsystems, without adding any interfaces. By connecting a Scope to the input and output interfaces, one can monitor their real-time values when the model is running on the target, which enables the developers to do on-the-fly measurement.

Besides calibration and measurement based on external mode, EcoCoder-AI also supports automatic cross-compiling on the host, automatic transmitting the generated executable files to the target, and automatic running the executable files on the target. After the developers finish building the model, they can implement cross-compile, copying files to the target and running the model by one single click. Afterward, one can click on the “Connect” button, the connection between the host and the target is built.

The process of using external mode is as follows:



6.1 Demo Directory

EcoCoder-AI contains demo models, in which developers can find the demo for all the function blocks. Users can use the demo models to perform an evaluation, or develop their own application model using the demo. **Demo_External.zip** is the one user can find in the demo directory for the external mode.

6.2 Make the Model Support External Mode

6.2.1 New Model Supports External Mode by Default

After installing EcoCoder-AI, users can drag the EcoCoder Target Definition block to the model, then this model supports external mode by default.

6.2.2 Use Command-Line to Convert the Old Model

If the old model doesn't support external mode, users can convert the model into one that supports external mode, by command line. Please make sure you back up the old model before conversion. Users can use this command in MATLAB command window to convert all the models in "desPath" directory:

```
EcoCoder_AutoEnAllExtMode(desPath, ipStr)
```

Or users can convert a single model using this command:

```
EcoCoder_AutoEnOneExtMode mdlPath, ipStr
```

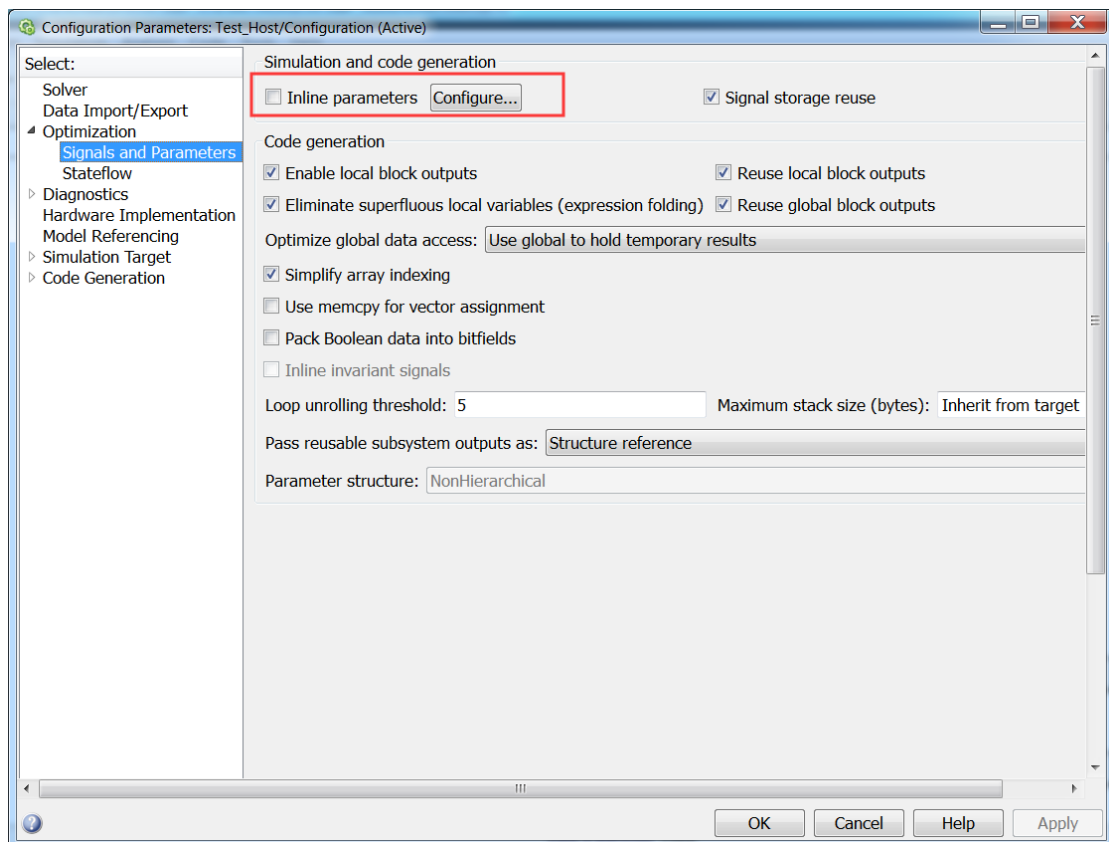
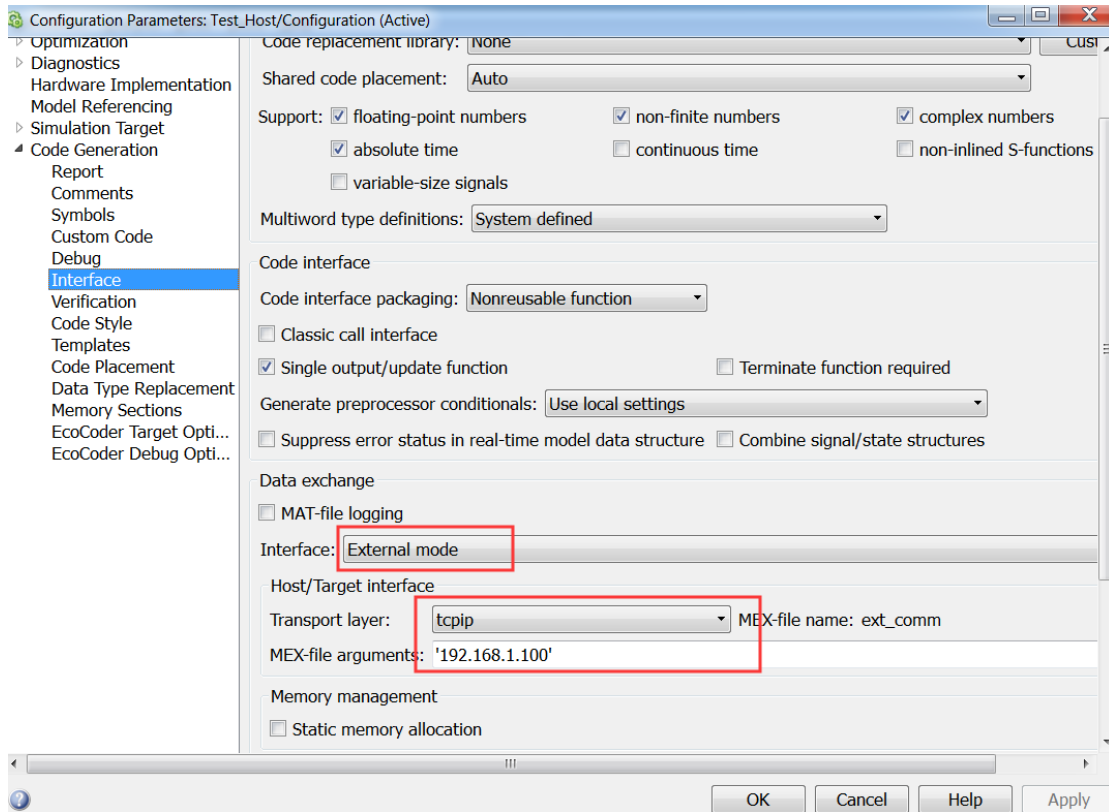
For example:

```
EcoCoder_AutoEnAllExtMode(pwd, '192.168.31.61')
```

```
EcoCoder_AutoEnOneExtMode('Test_Host.mdl', '192.168.31.60')
```

6.2.3 Change the Configuration of the Old Model

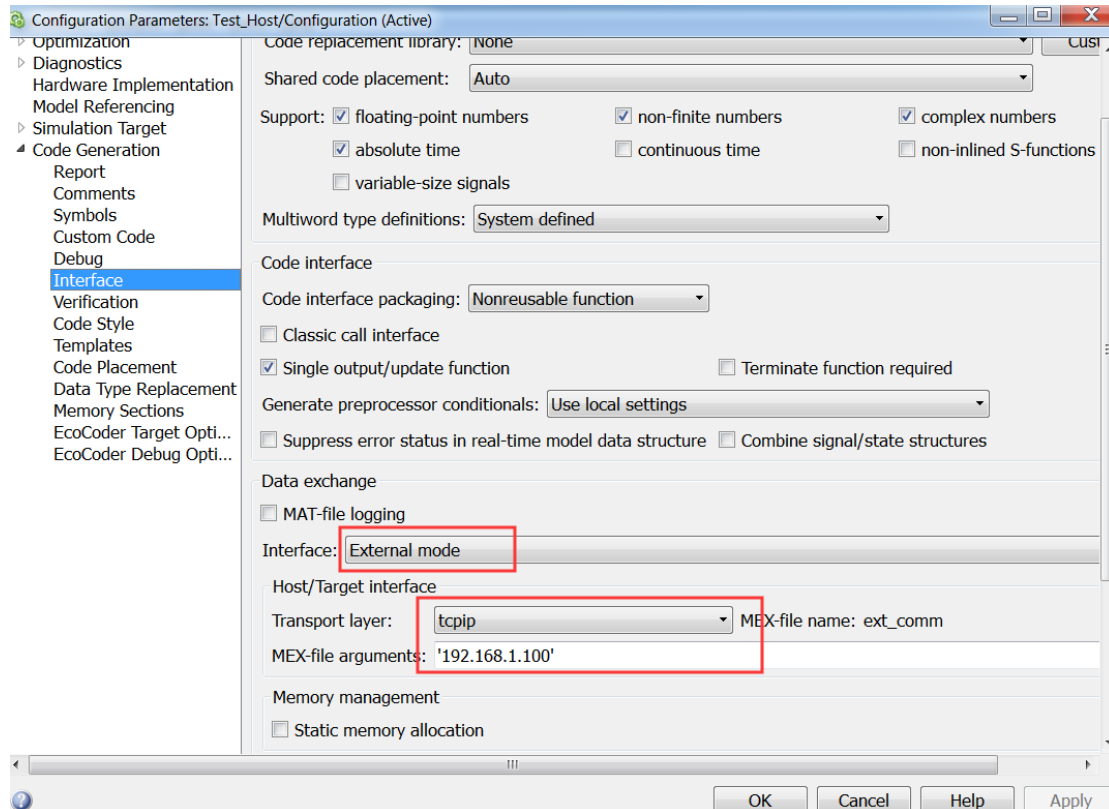
If the old model doesn't support external mode, users can convert it by changing the configuration as well. The process is as follows:



6.3 IP Settings for External Mode

External mode uses TCP/IP protocol to connect the host and the target. Therefore, in the model configuration, users need to fill in the Target IP address in the “MEX-file argument” textbox.

Target IP address needs to be in the same local network as the host and make sure the connection between them is good.

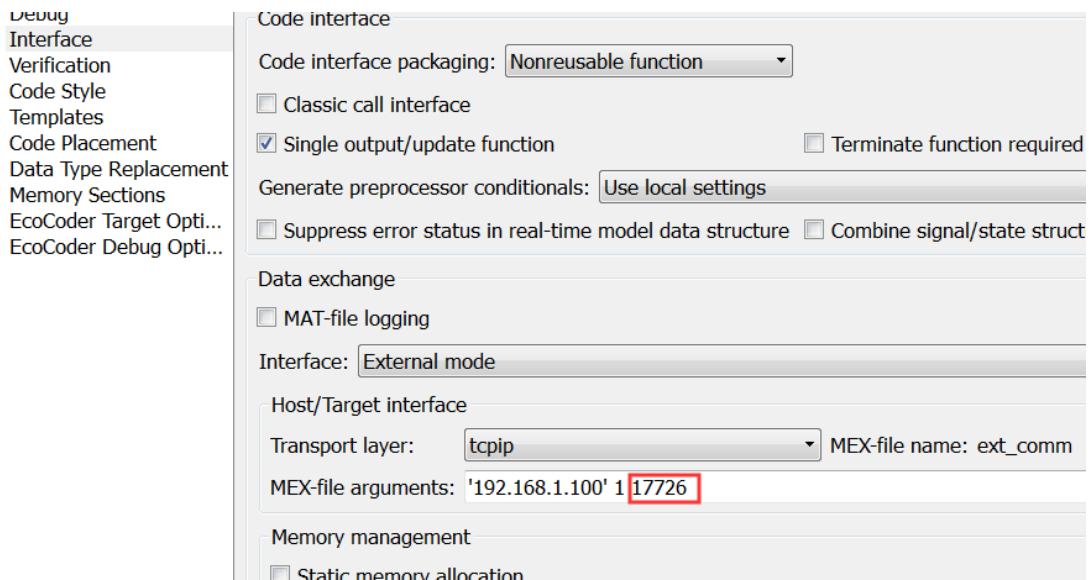


6.4 Using Different Ports

If there are multiple applications/models running in the target, and more than one of them need to connect to the host using the external mode, users need to configure different Port with the same IP address for them. The configuration should be done both in Simulink and the target.

6.4.1 Change the Port in Simulink

Users can change the port as follows. By default, the port is 17725. Users can change it into a number between 256 and 65535, but make sure it doesn't conflict with any other application.



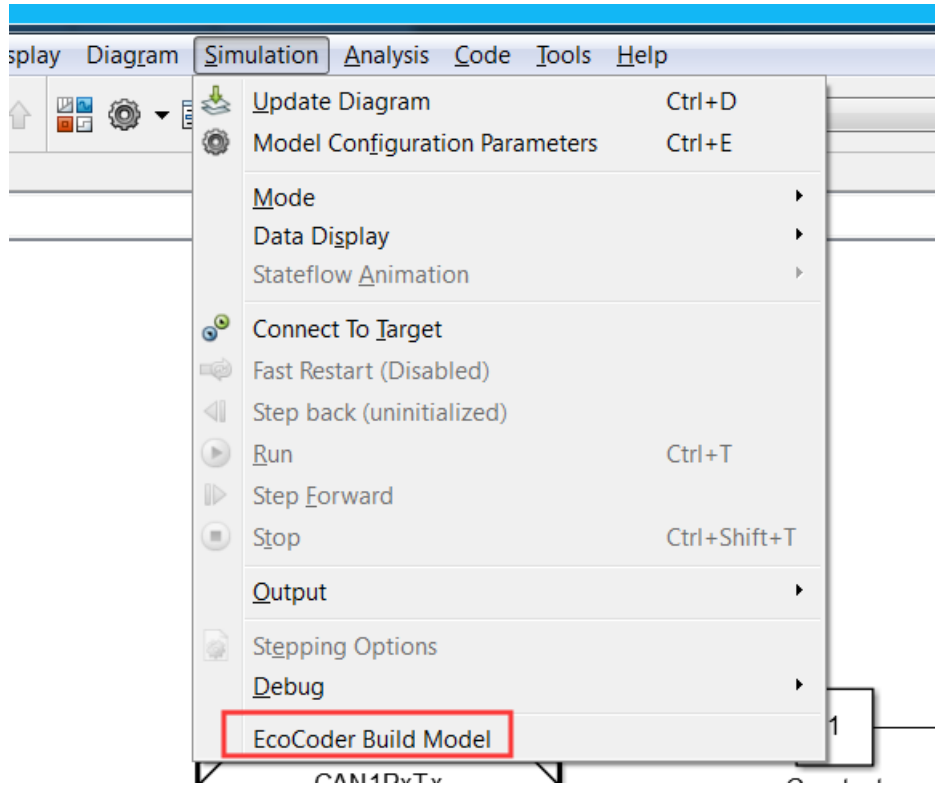
6.4.2 Change the Port in Target

Suppose the program that you want to run in the target is named as test_host_app, if you want it to use port 17726, you can type in the console on the target:

```
./test_host_app -port 17726
```

6.5 EcoCoder Build Model

When you finish designing the model, you need to generate the code from the model. In the external mode, the **Build Model Button** on tool bar is hidden, you need to use **Simulation -> EcoCoder Build Model** to build the model, as shown below.



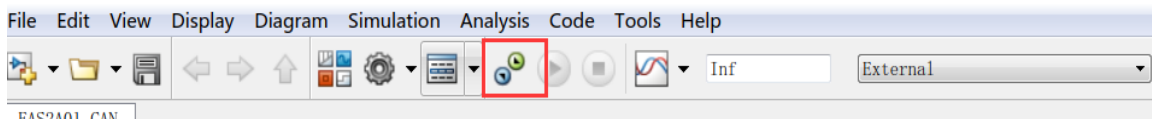
After the code is generated, EcoCoder will open the folder where the project files are located. If EcoSDK is installed in the host, EcoCoder will call the cross-compile tool to cross-compile the model and generate the executable files. If the port is configured correctly and the communication between the host and the target is well established, EcoCoder will copy the executable files to the target and run the program as well.

6.6 Connect the Host and the Target

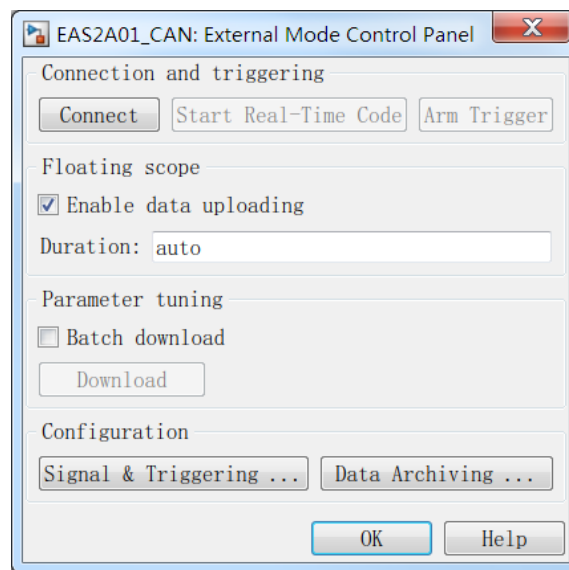
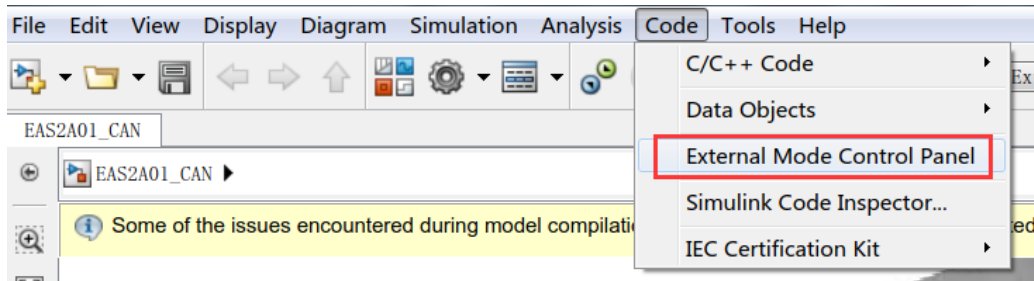
6.6.1 For MATLAB 2019a or Earlier Version

After the executable files are generated, users can establish the connection between the host and the target. Before connecting, make sure the executable files are running in the target.

You can click the “Connect” button in the Simulink toolbar to connect Simulink with the target.



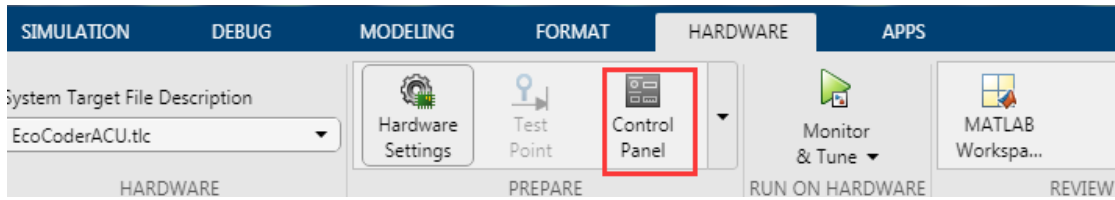
After the connection is successful, you can manage the external mode connection through “External Mode Control Panel” in the menu, you can also connect or disconnect with the target through the panel.



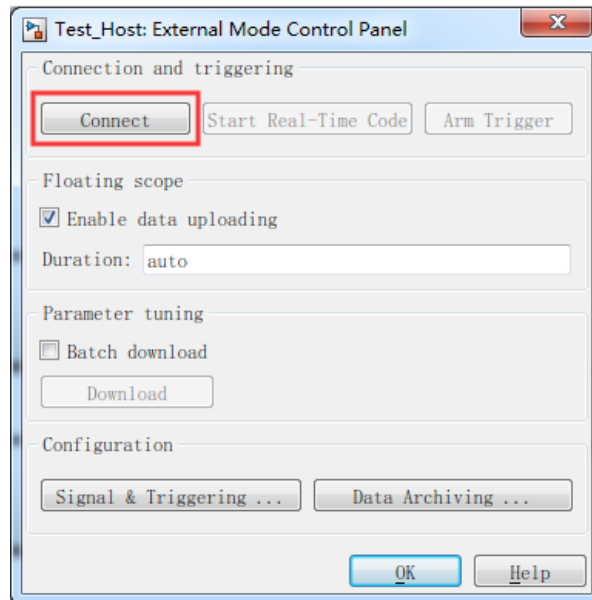
In the window above, you can see there is a “Batch download” checkbox. If it’s checked, you need to click the “Download” button to download the data to the target. If it’s not checked, the data will be sent to the target automatically after you change the model parameters.

6.6.2 For MATLAB 2019b

In MATLAB 2019b, all the external mode-related interfaces are located in the Hardware section, shown as follows:



You need to find the External Model Control Panel to connect or disconnect, as shown below.



6.7 Special Notes for External Mode

1. If there is any Model Reference in your model, parameter monitor and change are not supported in the model reference. You can monitor the parameters by setting it as an output of the model reference, and you can change the parameters by setting it as an input of the model reference.
2. Right now, on the target, you can **only run one model that supports external mode**.
3. When using the external mode, users can change the parameters in the function-call subsystems triggered by **Subscribe Message Block** and **Receive CAN Raw with Trigger Block**, but users cannot monitor the parameters in them. If you want to monitor the parameters in these kinds of subsystems, you can consider setting these parameters as the outputs of these subsystems.
4. When you convert the model that contains model references, make sure the configuration is consistent. You can drag **EcoCoder Target Definition Block** to the model to make it consistent. Please **do not put EcoCoder Target Definition Block in the function-call subsystems**.