EcoCoder Manual

V4.9.1

Copyright ECOTRON LLC

All Rights Reserved

Contact us:

Web: http://www.ecotron.ai
Email: info@ecotron.ai
support@ecotron.ai

Address: 13115 Barton Rd, STE H
Whittier, CA, 90605
United States

Tel: +1 562-758-3039
+1 562-713-1105

Date	Version	Comments	Editor
6/20/2019	4.8.2		Ted Wang
7/24/2019	4.8.3	NVM Demo	David Wang
8/11/2019	4.8.4	Model Reference Section	Ted Wang
05/11/2020	4.8.5	Contact info update	Zack Li
07/10/2020	4.8.6	Screenshot update	Yibo Wang
07/16/2020	4.8.7	NVM update	Jake Li
07/30/2020	4.8.8	Add Task Monitor	Jake Li
01/29/2021	4.8.9	Add more EcoCoder blocks	Michael Hu, Peter Zhu
02/11/2021	4.9.0	Update “build model” button, FNVM demo	Jake Li
04/12/2021	4.9.1	Update “Simulink C-code generation issue”	Eric Huo

Contents

CHAPTER 1 GENERAL INFORMATION 14

1.1 About EcoCoder 14

1.2 System Requirements 15

1.3 MATLAB Installation Requirements 15

1.4 Supported MATLAB Version 16

1.5 Developer's Kit 16

CHAPTER 2 ECOCODER DEVELOPMENT ENVIRONMENT 18

2.1 Software Installation List 18

2.2 CodeWarrior Installation 20

2.3 MinGW-GCC Compiler Installation 20

2.4 C++ Compiler Installation 23

2.4.1 Installation of Compiler for MATLAB 32-Bit 24

2.4.2 Compiler Selection for MATLAB 64-Bit 25

2.5 EcoCoder Installation 25

2.6 Link S32DS_Power_Win32 to EcoCoder 29

2.7 Link HighTec TriCore Tool Chain to EcoCoder 32

2.8 Link CS+ to EcoCoder 34

2.9 Activate EcoCoder 34

2.9.1 Get Key File 35

2.9.2 Activate EcoCoder by License (.dat) File 37

CHAPTER 3 QUICK START ON APPLICATION SOFTWARE 39

CHAPTER 4 ECOCODER LIBRARY 42

4.1 EcoCoder Target Definition 42

4.2 ADC Analog Inputs 44
4.2.1 Read ADC Value 44
4.2.2 Read Fixed-Point ADC Volt 45
4.2.3 Read Float ADC Volt 47

4.3 CAN Communication 49
4.3.1 CAN Channel Definition 49
4.3.2 CAN Wake-up Frame Definition 51
4.3.3 Read Fixed-Point CAN Message 52
4.3.4 Read CAN Message 55
4.3.5 Send Fixed-Point CAN Message 57
4.3.6 Send CAN Message 59
4.3.7 Receive CAN Message 61
4.3.8 Transmit CAN Message 62
4.3.9 Send CAN Data 63
4.3.10 CAN Receive Counter 63
4.3.11 Set CAN Mode 64
4.3.12 Unpack CAN Data to Signals 66
4.3.13 Pack Signals to CAN Data 67
4.3.14 Read CAN Error State 67
4.3.15 Receive CAN Raw with Trigger 68
4.3.16 Recover CAN Bus Off 68
4.3.17 CAN MESSAGE2NormalType 69
4.3.18 CAN NormalType2NormalType 70
4.3.19 Receive CANFD Raw With Trigger 71
4.3.20 Transmit CANFD Message 72
4.3.21 Disable CAN transceiver and Wake-up 72

4.4 Serial Communication Interface (SCI) Block 73
4.4.1 SCI Definition 73
4.4.2 Read SCI Data 74
4.4.3 Send SCI Data 74

4.5 Digital I/O 75
4.5.1 Switch Input 75
4.5.2 KeyOn Input 76
4.5.3 Switch Output 78
4.5.4 IPM Read 79
4.5.5 PWM IO Frequency Range Definition 79
4.5.6 IPWM Read 80
4.5.7 IPWM Interrupt Handler Definition 80
4.5.8 IPWM Read with Interrupt Handler 81
4.5.9 PWM Definition 82
4.5.10 PWM Output 83
4.5.11 PPM Read 84
4.5.12 Switch Internal 84
4.5.13 WakeUp Input 85
4.5.14 H-bridge Definition 86
4.5.15 H-bridge Output 87
4.5.16 INA226 Definition 88
4.5.17 Get INA226 Current 89
4.5.18 Get INA226 Shunt Voltage 89
4.5.19 Quadrature Decoder Definition 90
4.5.20 Quadrature Decoder Input 90
4.5.21 Quadrature Decoder Set Counter 91
4.5.22 Get Switch Output Sense Current 92

4.6 LIN Communication 92
4.6.1 LIN Channel Definition 92
4.6.2 LIN Get Status 93
4.6.3 LIN Receive Data 94
4.6.4 LIN Transmit Data 96

4.7 Task Scheduler 97
4.7.1 Task Trigger 97
4.7.2 Task Monitor 98

4.8 Non-Volatile Memory Blocks 99
4.8.1 Fixed NVM Definition 99

4.8.2 Read Fixed NVM 101
4.8.3 Write Fixed NVM 102
4.8.4 NVM Definition 102
4.8.5 NVM Variable Definition 103
4.8.6 Read NVM 104
4.8.7 Write NVM 105
4.8.8 Store All NVM Data 107
4.8.9 Restore All NVM Data 108
4.8.10 Get Fixed NVM Variable Address 108

4.9 Diagnostic Blocks 109
4.9.1 Hardware Output DTC 109
4.9.2 DTC Parser 111
4.9.3 Software Core Diagnostic 112
4.9.4 Clear H-bridge DTC 112
4.9.5 Clear TLF35584 All DTC 113
4.9.6 Get TLF35584 Summary DTC 113
4.9.7 Get TLF35584Specific DTC Total Number 114
4.9.8 Get TLF35584 Specific DTC 114

4.10 Calibration & Measurement 115
4.10.1 Calibration Definition 115
4.10.2 Read Calibration 116
4.10.3 Write Measurement 117
4.10.4 Write and Read Measurement 118
4.10.5 Override Probe 119
4.10.6 1-D Lookup Table 121
4.10.7 2-D Lookup Table 122
4.10.8 Calibration Data Check 123

4.11 System Management Blocks 124
4.11.1 Power Management Example 124
4.11.2 Shutdown Power 126
4.11.3 Shutdown Power Hold 127
4.11.4 Set ECU Mode 127
4.11.5 ECU Master Chip Wake-Up Definition 128

4.11.6 Watchdog Definition 129
4.11.7 Service Software Watchdog 130
4.11.8 Software Reset 131
4.11.9 Software Reset Hold 131
4.11.10 Read System Free Counter 132
4.11.11 Power Control Output 132
4.11.12 Stack Overflow Detection Definition 133
4.11.13 Detect Stack Overflow 133
4.11.14 Read BootID 134
4.11.15 Hardware Reset 134

4.12 CCP 134
4.12.1 Fixed CCP Slave Definition 134
4.12.2 CCP/CAL Seed&Key Security Definition 136
4.12.3 CCP DAQ Seed&Key Security Definition 137
4.12.4 CCP PGM Seed&Key Security Definition 137
4.12.5 CCP Generate Seed Demo 138
4.12.6 CCP Get Seed Trigger 138
4.12.7 CCP Set Seed 139

4.13 Sensors Blocks 139
4.13.1 Read Gyro Hex Value 139
4.13.2 Read Gyro Phy Value 140

4.14 Programming Blocks 140
4.14.1 Online Programming Definition 140
4.14.2 Programming Seed&Key Definition 143
4.14.3 Entry UDS Programming 145

4.15 Advanced Data Blocks 145
4.15.1 Read OTP 145
4.15.2 Read OTP (Input port) 146
4.15.3 Write OTP 147
4.15.4 Write OTP (Input port) 149
4.15.5 Read Data by Address 150
4.15.6 Read Data by Address (Input port) 151

4.15.7 Read String Value 152
4.15.8 Read EEPROM 152
4.15.9 Write EEPROM 153
4.15.10 EEPROM Emulation Definition 154
4.15.11 Clear ALL EEPROM Emulation Record 154
4.15.12 Clear One EEPROM Emulation Record 155
4.15.13 Read EEPROM Emulation Record 155
4.15.14 Write EEPROM Emulation Record 156
4.15.15 Read Signals from EEPROM Emulation Record 157
4.15.16 Write Signals to EEPROM Emulation Record 158
4.15.17 EEPROM Emulation Area Need to Erase 158
4.15.18 Erase EEPROM Emulation Area 159
4.15.19 Program First Run Flag 160
4.15.20 Write RAM Data by Address 160
4.15.21 Get General Variable Address 161

4.16 Application Base Blocks 162
4.16.1 Rising Edge 162
4.16.2 Falling Edge 162
4.16.3 Online Programming by SoftReset 162
4.16.4 Online Programing By HardReset 164
4.16.5 dt – time step length 165
4.16.6 Rising Edge Debounce 165
4.16.7 Falling Edge Debounce 166
4.16.8 PT1 Filter 166
4.16.9 Hysteresis 167
4.16.10 SR Flip Flop 167

4.17 Volatile Variable 168
4.17.1 Read Const Volatile Variable 168

4.18 XCP Module 169
4.18.1 XCP Slave Definition 169

4.19 FlexRay Module 169
4.19.1 FlexRay Definition 169

4.19.2 FlexRay Get Current State 171
4.19.3 FlexRay Setup Network Control 171
4.19.4 FlexRay Force to Halt 172
4.19.5 FlexRay Restore to Default Configuration 172
4.19.6 Read FlexRay Message 173
4.19.7 Write FlexRay Message 173

4.20 Ethernet 174
4.20.1 Ethernet Definition 174
4.20.2 Ethernet Handler 175
4.20.3 Ethernet Init 175

4.21 TCP Protocol Blocks 175
4.21.1 TCP Server Definition 175
4.21.2 TCP Server Init 176
4.21.3 TCP Client Definition 177
4.21.4 TCP Client Connect 177
4.21.5 TCP Close 178
4.21.6 TCP Abort 178
4.21.7 TCP State 179
4.21.8 TCP Receive 180
4.21.9 TCP Transmit 180

4.22 UDP Blocks 181
4.22.1 UDP Definition 181
4.22.2 UDP Init 182
4.22.3 UDP Receive 182
4.22.4 UDP Transmit 183

4.23 SPI Blocks 184
4.23.1 SPI Definition 184
4.23.2 SPI Master Exchange Data 185
4.23.3 SPI Slave Exchange Data 186

4.24 Model Reference 187
4.24.1 Configurations for Parent Models and Referenced Models 188
4.24.2 Configuration Reference 189

4.24.3 Copy Parent Model Configuration File to Referenced Model 191
4.24.4 EcoCoder Blocks in Model Reference 191

CHAPTER 5 CAN THEORY OF ECOTRON 194

5.1 Introduction 194

5.2 CAN Implementation 194
5.2.1 Convert DBC to m File 195
5.2.2 EcoCoder CAN Blocks 197
5.2.3 Select m file 197
5.2.4 Select Message 198
5.2.5 Select Sample Time 199

CHAPTER 6 MEMORY MANAGEMENT 201

6.1 Introduction 201

6.2 Storage device 201

6.3 Data Storage 201
6.3.1 Calibration/Measurement Variable 201
6.3.2 Non-Volatile Variable 202

CHAPTER 7 CUSTOM VARIABLE TYPE 203

7.1 Customize Variable Types 203

7.2 Add Variables to Workspace 204

7.3 Customize Calibration Variables 206

7.4 Customize measurement Variables 207

7.5 Customize NVM Variables 208

7.6 Save the Variables to M file 209

7.7 Load M file to Workspace 210

7.8 Model Example 211

CHAPTER 8 PROGRAMMING VCU WITH ECOFLASH 212

CHAPTER 9 MEASUREMENT AND CALIBRATION WITH ECOCAL 213

CHAPTER 10 UNINSTALL ECOCODER 214

10.1 Uninstall EcoCoder from MATLAB 214

10.2 Uninstall EcoCoder from Windows System 215

CHAPTER 11 FAQS 216

11.1 Q1. The m file exported from DBC by 'EcoCAN' can't be used 216

11.2 Q2. Model created by 'EcoCoder_Prj', emulation or code generation error 216

11.3 Q3. 'CAN' module is blank after being configured 216

11.4 Q4. EcoCoder Loader Pop-up error 217

11.5 Q5. How to update application model to be compatible with updated EcoCoder .. 217

11.6 Q6. Is there a way to get rid of popping up folder of generated file? 219

11.7 Q7. Is there a way to access project file in C code? 220

11.8 Q8. C code generation by Simulink is not accepted by compilers. 221

APPENDIX A - NONVOLATILE VARIABLES 223

Non-volatile Variables 223

Fixed Non-volatile Variables 223

Fixed NVM Demo 223

APPENDIX-B BATTERY INPUT 226

APPENDIX C-MAB(MATHWORKS ADVISORY BOARD) 227

Chapter 1 General Information

1.1 About EcoCoder

EcoCoder is an advanced auto code generation library added on top of Simulink generic libraries. It links the user's Simulink models directly to Ecotron target controller.

EcoCoder encapsulates the low level driver software, or basic software, also abstracts the specific hardware, like Freescale or Infineon microprocessor-based controllers. It enables the controls engineer to develop their control systems completely in MATLAB/Simulink environment.

Plus, EcoCoder is only an add-on package on top of Simulink. It enables engineers to maximize the usage of Simulink generic library. It adds the necessary library blocks which bridge the gap between application software and the specific controller hardware. Meaning the application software will not be dependent on the specific hardware, and you can port your models to any other hardware which supports the Simulink. In short, you are not stuck with EcoCoder by using it.

Features:

- Auto-code generation of Simulink/Stateflow models using Embedded Coder/Stateflow Coder
- Calibration using EcoCAL or other CCP based software
- Programming using EcoFlash through CAN bus
- OTA upgrade of application software
- Available for both prototyping and production
- Manual C-code integration is available in addition to model-based design (MBD) with Simulink/EcoCoder

Benefits:

- Control engineers can be freed from time-consuming learning curve of hardware, C programming, and specific microprocessor settings
- Responsive support services from Ecotron

  Application software development is isolated from a specific hardware, and it has transparency and easy migration to other platforms.

1.2 System Requirements

OS	Windows XP/Windows 7/Windows 10
CPU	Intel CORE 2 Duo or higher
Memory	2 GB or higher
Hard drive	1 GB free hard disk space

1.3 MATLAB Installation Requirements

Mandatory Components:

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

Highly recommended components to be installed:

- Stateflow
- Stateflow Coder

1.4 Supported MATLAB Version

- MATLAB R2010b 32-bit/64-bit
- MATLAB R2011a 32-bit/64-bit
- MATLAB R2011b 32-bit/64-bit
- MATLAB R2012a 32-bit/64-bit
- MATLAB R2012b 32-bit/64-bit
- MATLAB R2013a 32-bit/64-bit
- MATLAB R2013b 32-bit/64-bit
- MATLAB R2014a 32-bit/64-bit
- 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

Note: some of the MATLAB versions (old) may require extra configurations to make the EcoCoder work. Contact us if you have compatibility issues.

1.5 Developer's Kit

- VCU
- Test Harness*
- Ecotron CAN (USB-CAN Adapter) *

* Test harness is available from Ecotron; however, users can also make their own by using recommended connector parts.

* Ecotron CAN needs to be compatible with CAN Calibration Protocol (CCP). Third party adaptors like Kvaser

or PeakCAN should be compatible with Ecotron products.

Ecotron USB-CAN Adaptor

VCU

Test Harness

Chapter 2 EcoCoder Development Environment

2.1 Software Installation List

Please install software tools in the following order:

1. Integrated development environment for generating executable files

Main Chip	Integrated development environment
Infineon TC27x	HighTec TriCore Tool Chain
NXP SPC57xx	*S32DS_Power_Win32_v2017.R1_b171019.exe*
NXP SPC56xx	*CodeWarrior for MPC55xxMPC56xx v2.10.exe*
Renesas RH850	*CS+(CSPlus_CC_Package_V70000.EXE)*

2. Compiler for generating DLL file (optional)

Compiler	Integrated development environment
Tdm64-gcc-4.9.2.exe	GCC compiler can generate DLL file, to set calibration, measurement, and program flashing permission. Support all MATLAB versions to generate DLL file.

3. *EcoFlash Vxxxx Setup.exe*
4. *EcoCAL Vxxxx Setup.exe*, or INCA
5. Ecotron USB-CAN adapter Driver, or other CAN adapter Driver, such as Kvaser product
6. Stateflow Coder (optional)

Compiler	Supported MATLAB version
C++ Compiler	Support Stateflow of MATLAB 32/64-bit
Lcc-win32	Support Stateflow of MATLAB 32-bit
MinGW-GCC	Support Stateflow of some MATLAB versions Please see the corresponding relationship between the MATLAB version and the supported MinGW version

7. EcoCoder Setup.msi

Compilers and its Applications:

Compiler or integrated development environment utility	Application
HighTec TriCore Tool Chain	Compilation and link; HEX file generation for target EH2175A
S32DS_Power_Win32_v2017.R1_b171019.exe	Compilation and link; MOT file generation for NXP MPC5744 based units.
CodeWarrior for MPC55xxMPC56xx v2.10.exe	Compilation and link; MOT file generation for target EV2206B03, ET3206A
C++ Compiler	Support Stateflow of 32-bit and 64-bit MATLAB
Lcc-win32	Support Stateflow of 32-bit MATLAB
WinGW-GCC	1. Generate DLL file for calibration, measurement and programming permission, support all versions of MATLAB 2. Support Stateflow of some MATLAB versions, you can go to MATLAB official website to check whether it supports specific MATLAB version
CS+(CSPlus_CC_Package_V70000.EXE)	Compilation and link, MOT file generation for target GWRH850

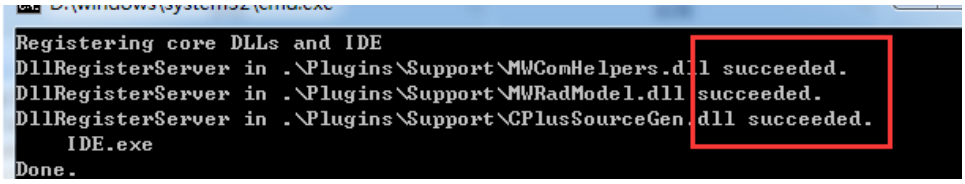
The corresponding relationship between the MATLAB version and the supported MinGW version:

MATLAB version	Supported MinGW version
MATLAB2015a or below	Not support
MATLAB2015b	Support MinGW 4.9.2 (Distributor: TDM-GCC)
MATLAB 2016a	Support MinGW 4.9.2 (Distributor: TDM-GCC)
MATLAB 2016b	Support MinGW 4.9.2 (Distributor: TDM-GCC)
MATLAB 2017a	Support MinGW 4.9.2 (Distributor: TDM-GCC)
MATLAB 2017b	Support MinGW 5.3 (Distributor: TDM-GCC)
MATLAB 2018a	Support MinGW 5.3 (Distributor: TDM-GCC)

2.2 CodeWarrior Installation

The installation instruction is for *CodeWarrior MPC55xxMPC56xx v2.10.exe*, if you installed other CodeWarrior version, please do the following after installation:

1. Run the “regserves.bat” file in the installation directory “Freescale\GW for MPC55xx and MPC56xx 2.10\bin”. When the window appears, press any key to exit.

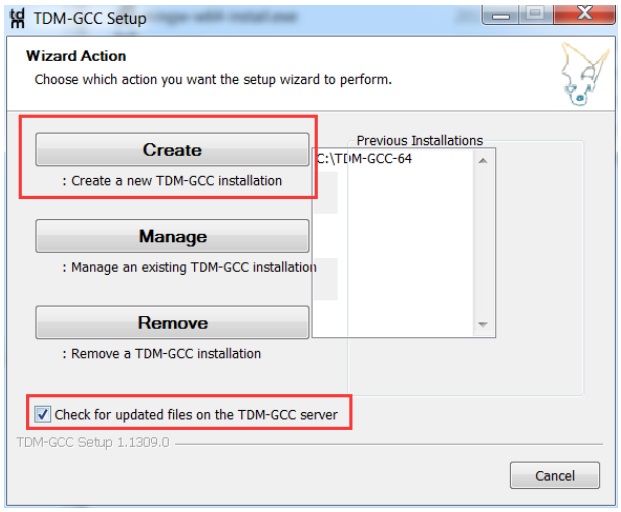


2. If you follow the step 1 and other versions are called by default during compilation, you will need to uninstall other versions of CodeWarrior.

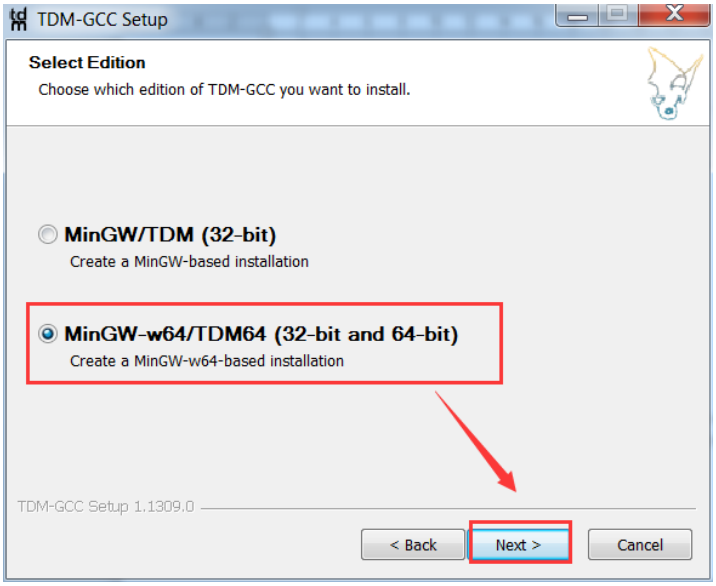
2.3 MinGW-GCC Compiler Installation

The MinGW-GCC compiler can be installed via TDM-GCC. TDM-GCC is a compiler integration package for Windows that combines the latest version of the GCC toolset and includes API of open source MinGW or MinGW-w64. The installation steps are as follows:

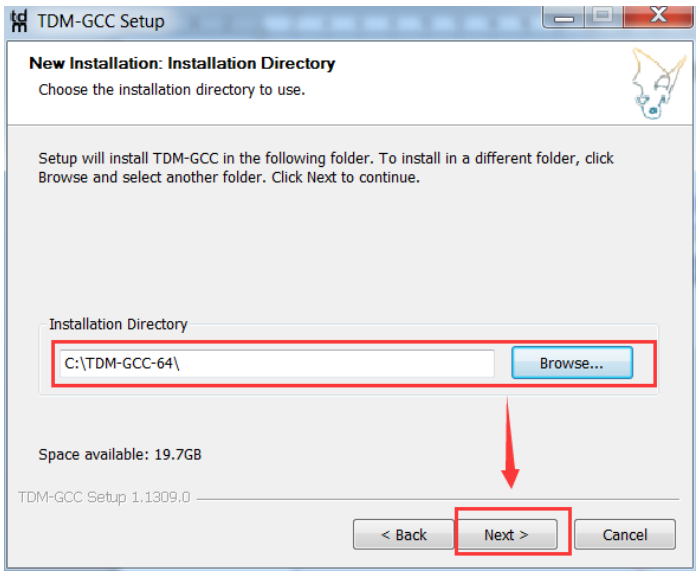
1. First check “Check for updated files on the TDM-GCC server”



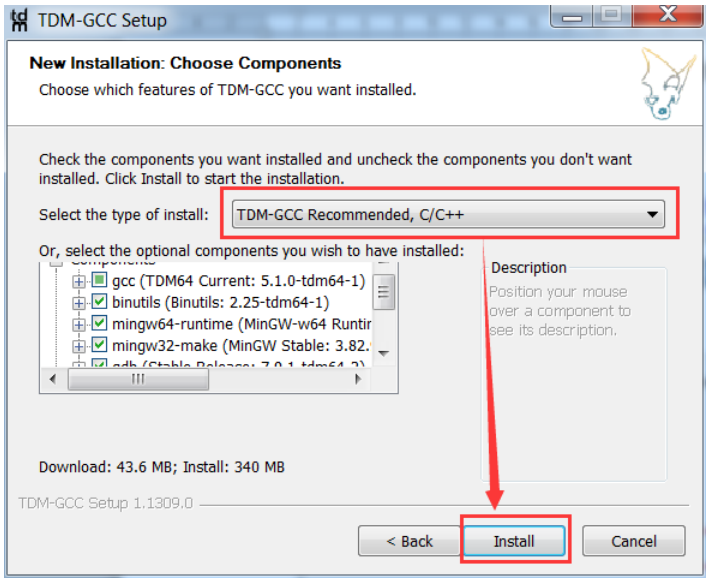
2. Choose MinGW-w64



3. Choose the installation directory

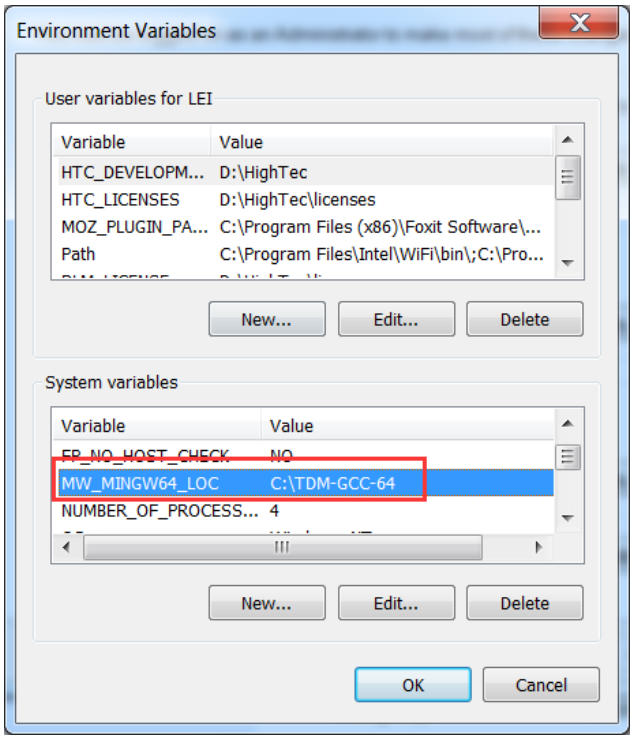


4. Choose TDM-GCC Recommended, C/C++

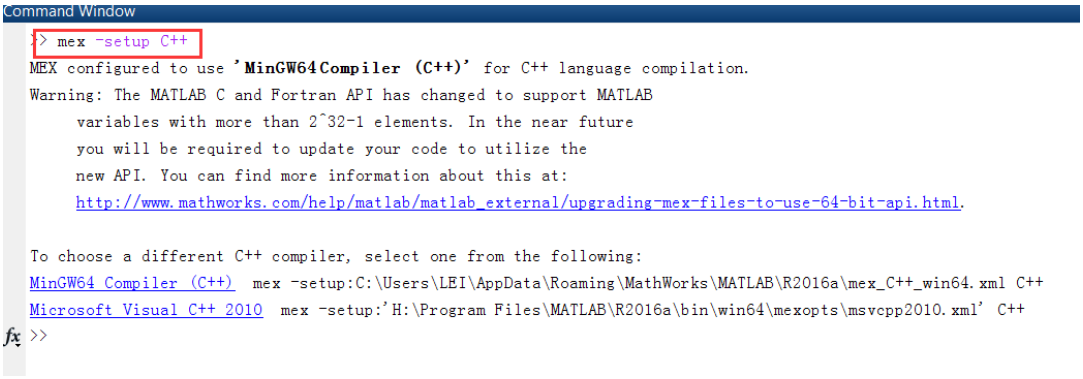


Note: If the user uses MinGW-GCC as the Stateflow complier, proceed to the next step.

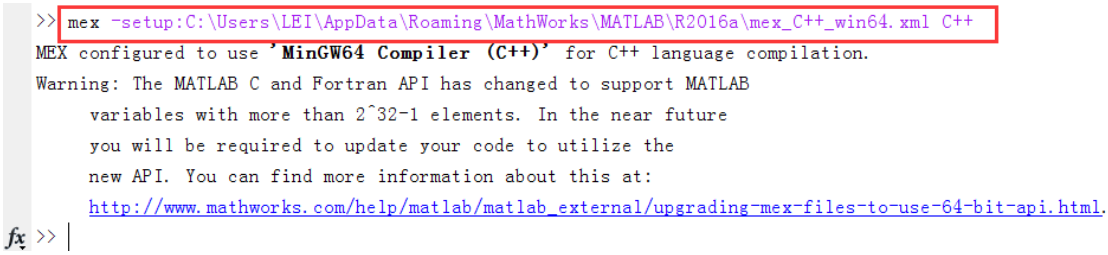
5. Add environment variables



6. Restart or open MATLAB.
7. Enter “mex -setup C++”



8. Choose MinGW64 Complier



2.4 C++ Compiler Installation

MATLAB 32-bit system comes with a ‘LCC’ compiler which supports Stateflow automatic code generation. MATLAB 64-bit system does not provide compiler. To use Stateflow coder, it is necessary to install a third-party C++ Compiler that supports MATLAB 64-Bit version.

2.4.1 Installation of Compiler for MATLAB 32-Bit

1. Type ‘mex -setup’ at MATLAB Command Window.

2. Type ‘y’ at Command Window.

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

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

3. Type ‘1’ at Command Window.

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2010b\sys\lcc

[0] None

Compiler: 1
```

4. Type ‘y’ at Command Window.

```
Compiler: Lcc-win32 C 2.4.1
Location: C:\PROGRA~1\MATLAB\R2010b\sys\lcc

Are these correct [y]/n? y
```

5. When the following information is displayed, the installation is successful.

```
Trying to update options file: C:\Documents and Settings\Administrator\Application Data\MathWorks\MATLAB\R2010b\mexopts.bat
From template:              C:\PROGRA~1\MATLAB\R2010b\bin\win32\mexopts\lccopts.bat

Done . . .
```

2.4.2 Compiler Selection for MATLAB 64-Bit

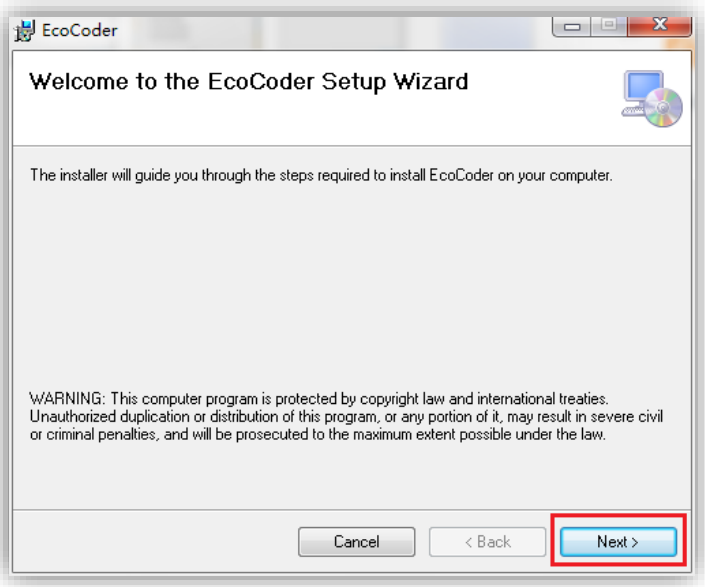
1. Go to the official website of MathWorks

   https://www.mathworks.com/support/sysreq/previous_releases.html
2. Click '*Details*' under '*Supported Compilers*' of MATLAB version on customer PC.
3. For Windows 64-bit system, refer to the page titled: '*Windows 64bit'*.
4. After finishing the compiler installation, follow the steps in previous section to configure compiler for MATLAB.

2.5 EcoCoder Installation

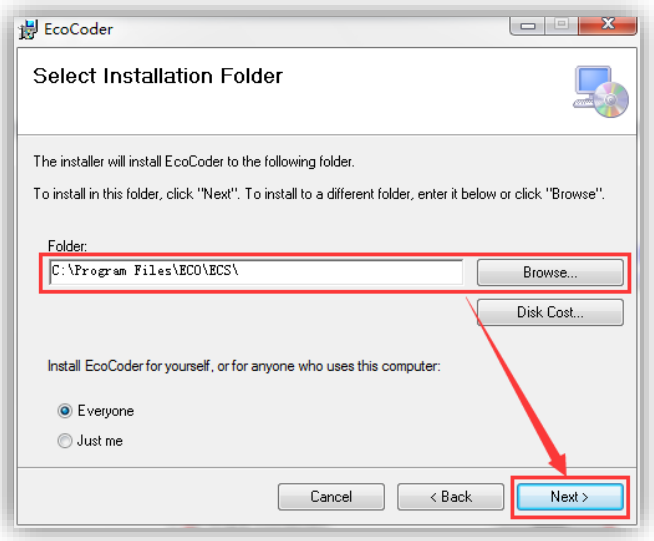
Note: Please keep MATLAB closed during the entire installation and licensing process.

1. Double-click 'EcoCoder 56xx Vx.x.x Setup.msi', click '*Next*' at the following screen.

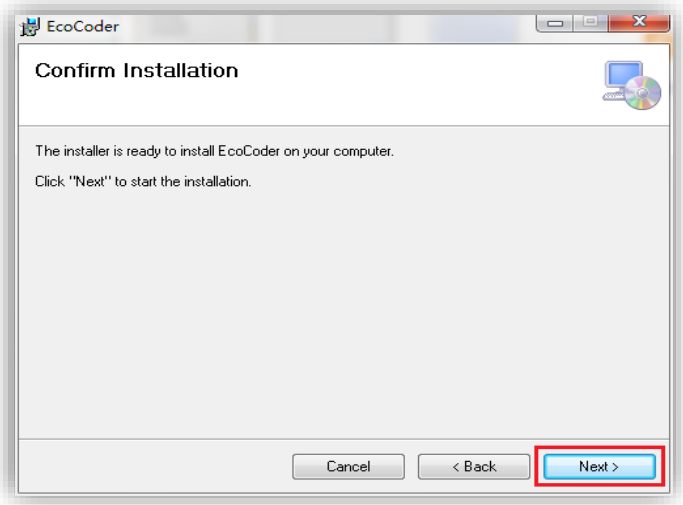


2. Choose installation path, click '*Next*'.

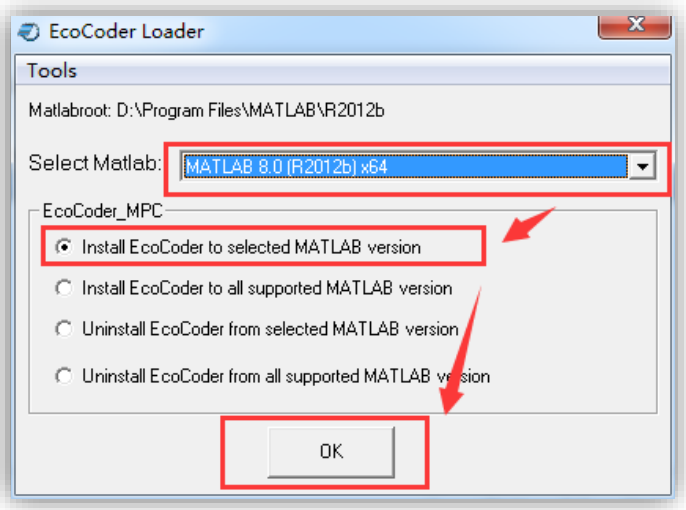
Note: it is recommended to install EcoCoder under the system drive.



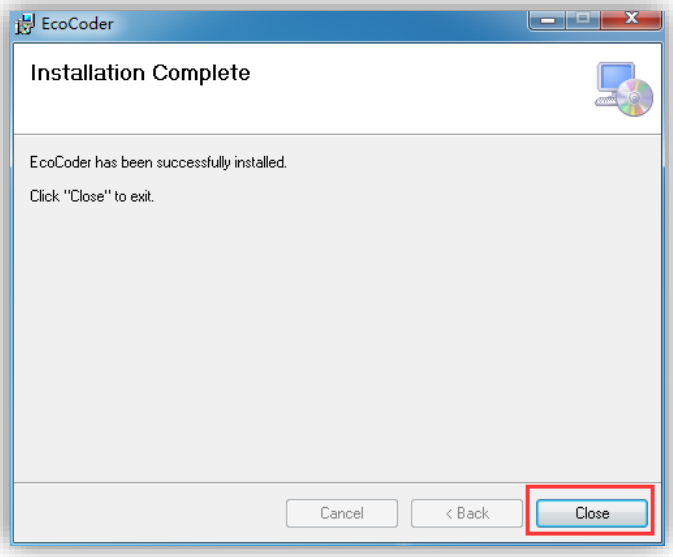
3. Click '*Next*'.



4. Choose the version of MATLAB you want to install EcoCoder to, then select '*Install EcoCoder to selected MATLAB version*', click *'OK'*. You can also install EcoCoder for all MATLAB versions on computer.

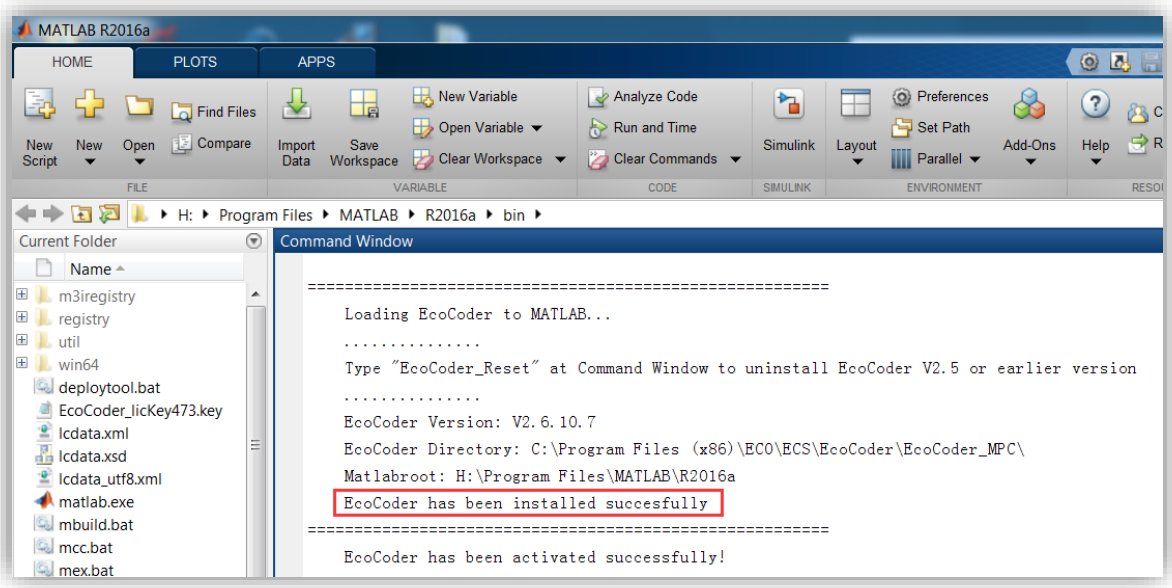


5. Click '*Close*'.



6. After successfully install it, the icon '*EcoCoder Loader*' will appear on the desktop. EcoCoder Loader will be used to generate the license file and activate EcoCoder.

7. If you run MATLAB then, it will prompt message '*EcoCoder has been installed successfully'* as shown in following red box. It indicates that EcoCoder is successfully installed to MATLAB.



2.6 Link S32DS_Power_Win32 to EcoCoder

Please refer to this video for setting up the S32DS IDE:

https://youtu.be/CiChf1_JZcw_

Link for copy and paste:

https://www.nxp.com/support/developer-resources/run-time-software/s32-design-studio-ide/s32-design-studio-ide-for-power-architecture-based-mcus:S32DS-PA?tab=Design_Tools_Tab

NXP provides it for free, but you need to register your own account.

http://tdm-gcc.tdragon.net/download

And this is the step where you can copy and paste the path:

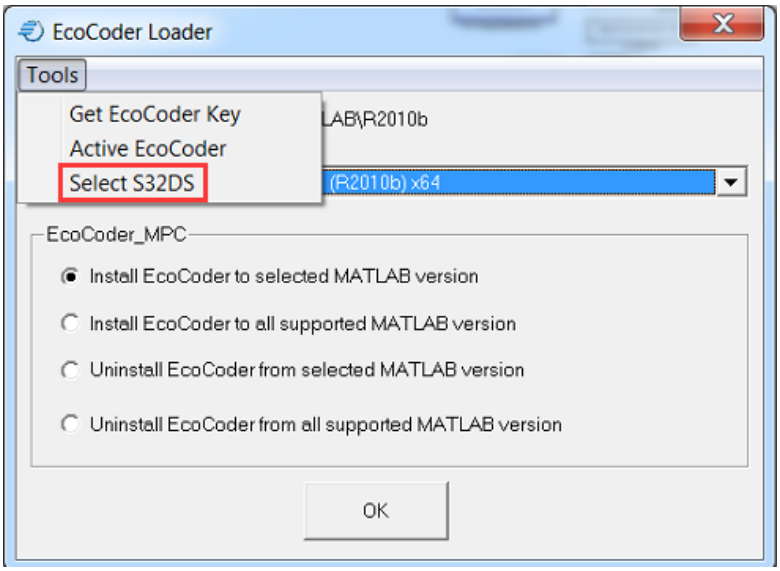
Add two directories where powerpc-eabivle-gcc.exe and make.exe are located to the environment variables. Split them by semicolon. Add at the end of the Variable value:

C:\NXP\S32DS_Power_v2017.R1\utils\msys32\usr\bin;C:\NXP\S32DS_Power_v2017.R1\Cross_Tools\powerpc-eabivle-4_9\bin

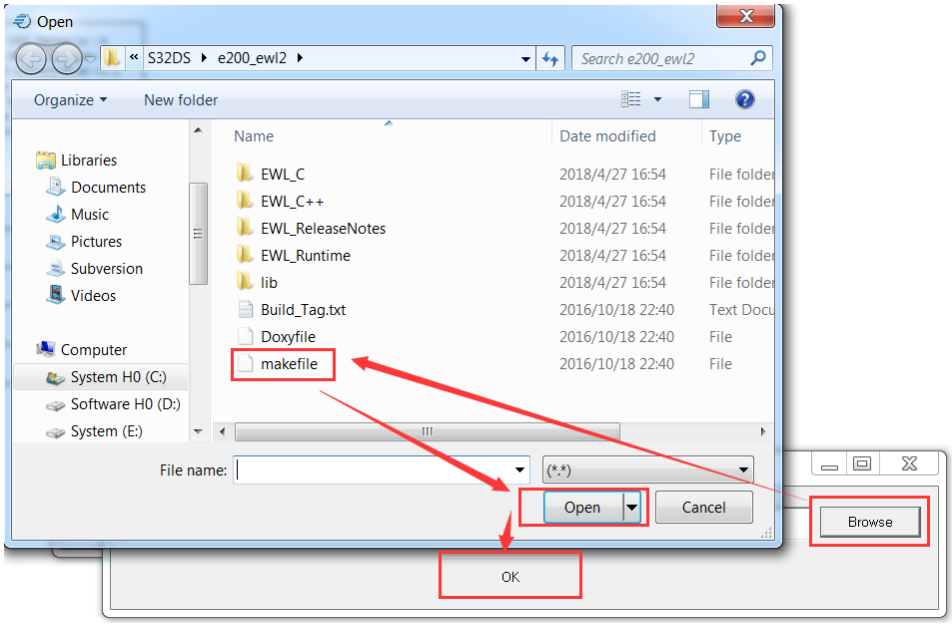
In case you're not comfortable with video, here are the text steps:

If you use any MPC5744 based unit, you need to install the S32DS as a compiler, such as *S32DS_Power_Win32_v2017.R1_b171019.exe*, which can be downloaded from the NXP official website for free. After installing EcoCoder and S32DS_Power_Win32, you also need to use EcoCoder to select the path of *makefile* and add two directories where *powerpc-eabivle-gcc.exe* and *make.exe* are located to the environment variables.

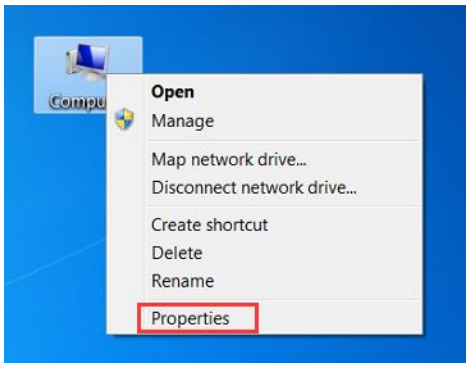
1. Open EcoCoder, choose Tools > Select S32DS



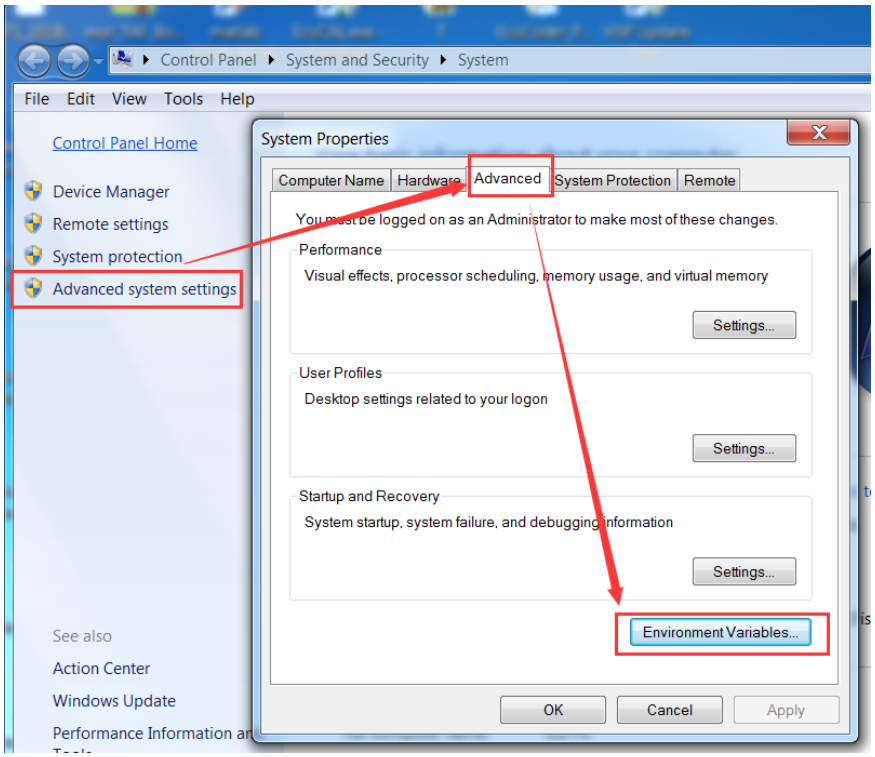
2. Click Browse, choose *makefile* under e200_ew12, for example, the full path is:



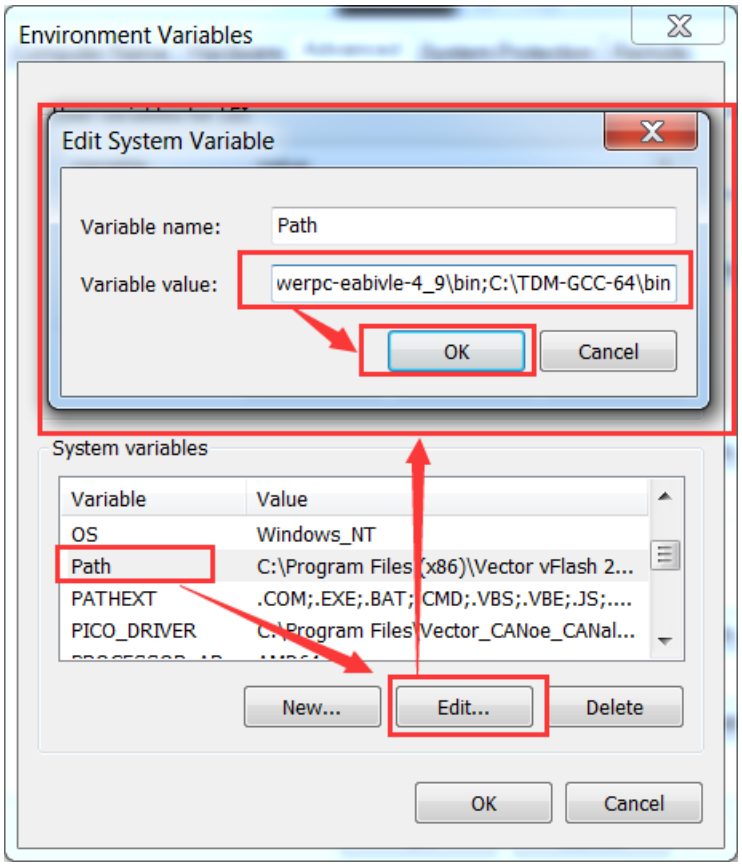
3. Right click on Computer, then click on properties



4. Choose Advanced system settings > Environment Variables



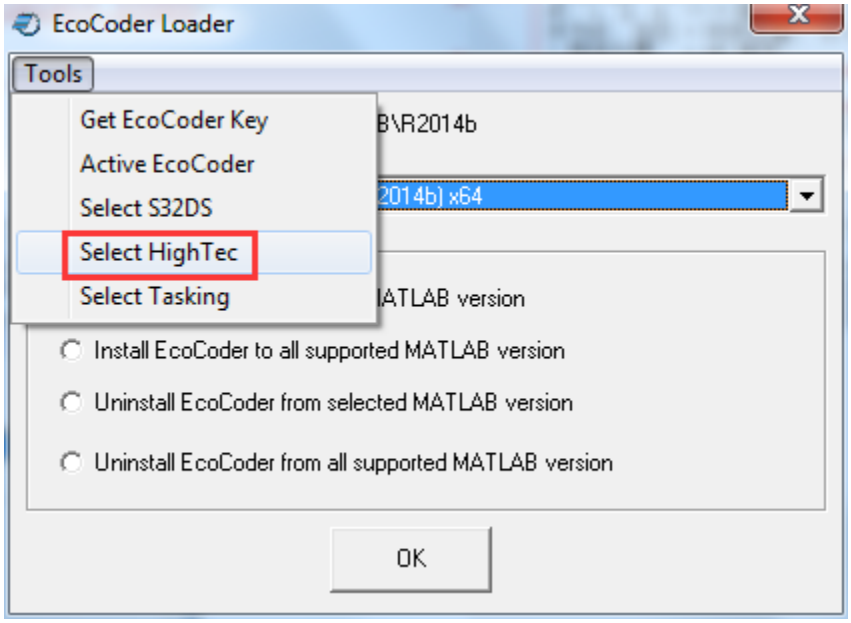
5. Add two directories where powerpc-eabivle-gcc.exe and make.exe are located to the environment variables. Split them by semicolon. Add at the end of the Variable value:
   C:\NXP\S32DS_Power_v2017.R1\utils\msys32\usr\bin;C:\NXP\S32DS_Power_v2017.R1\Cross_Tools\powerpc-eabivle-4_9\bin



2.7 Link HighTec TriCore Tool Chain to EcoCoder

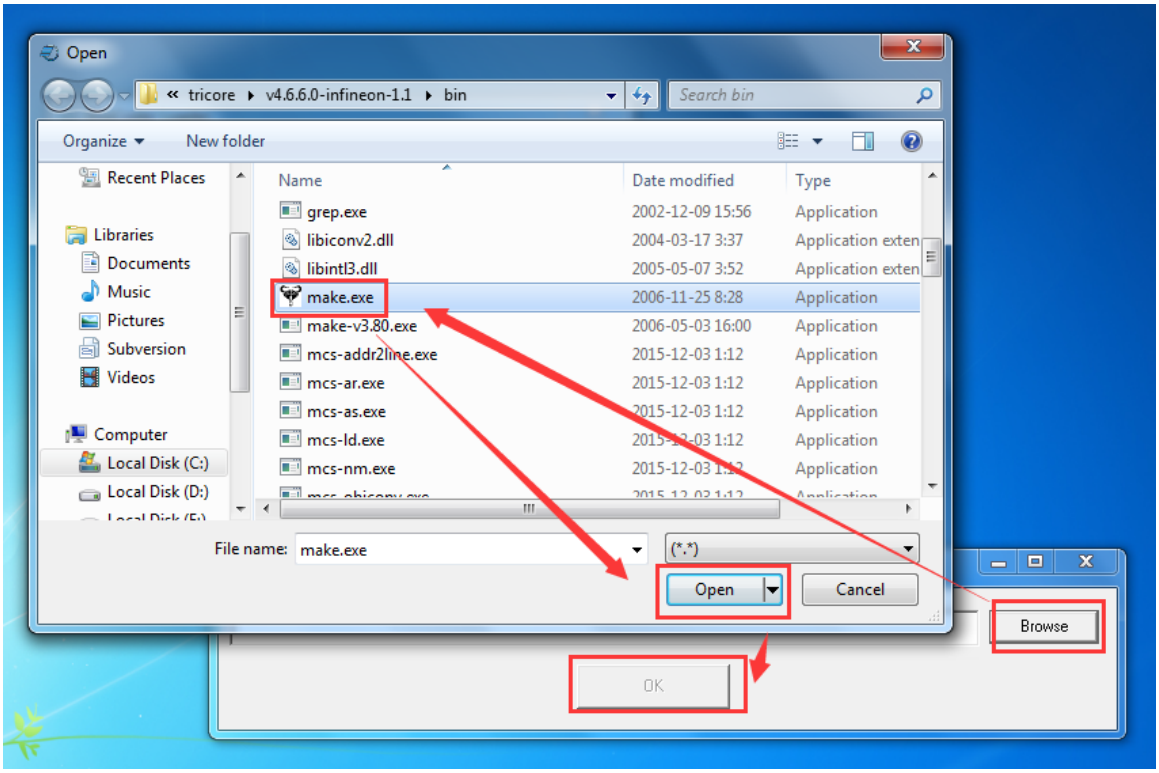
For EH2175A/EH2275A/ EAXVA03, the installation of HighTec TriCore tool chain is required. After the installation of HighTec TriCore Tool Chain, users need to specify the *makefile* directory in EcoCoder Loader.

Open EcoCoder Loader, click Tools, then click “Select HighTec”.



Then in the pop-up window, click “Browse”, locate the “make.exe” in the HIGHTEC installation path, in “bin” folder under the folder ‘toolchains’.

For example, the full path can be: “C:\HIGHTEC\toolchains\tricore\v4.9.1.0-infineon-1.1\bin\make.exe”



2.8 Link CS+ to EcoCoder

If target is GWRH850, CS+ should be installed as a compiler, for example, you can install *CSPlus_CC_Package_V70000.exe*. After installation, you need to add the directory where the file *CubeSuite+.exe* is located to the system environment path, then restart MATLAB. Regarding how to add environment variables, please refer to the operations in Link S32DS_Power_Win32 to EcoCoder.

2.9 Activate EcoCoder

There are two ways to activate EcoCoder and other Ecotron software.

Note: Please close MATLAB for the activation process.

1. Dongle

The hardware dongle released by Ecotron can activate software once it is plugged in PC.

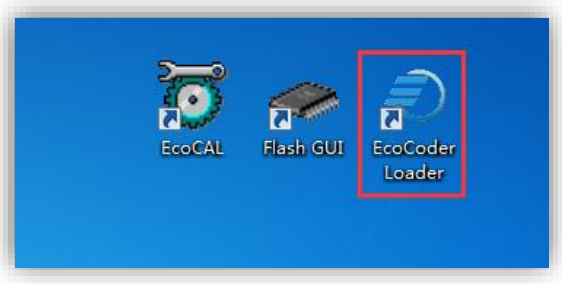
2. '*.dat'* file

'*.dat*' file is linked to Windows SN, meaning the *'.dat'* file is bound to a specific PC and not allowed to be transferred to another PC. New *'.dat'* file must be issued if customer shift to new PC.

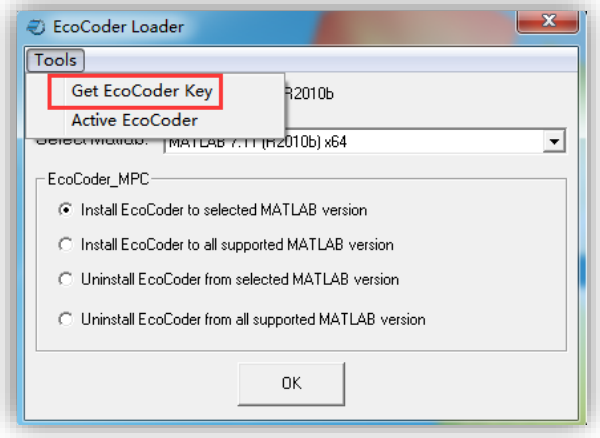
Note: all Ecotron software would 'remember' license once it is activated even after it is upgraded to new version. It is mandatory to use Windows Add/Remove programs to uninstall all previously installed versions of EcoCoder. For safety concern, please install new version to same folder as previous EcoCoder.

2.9.1 Get Key File

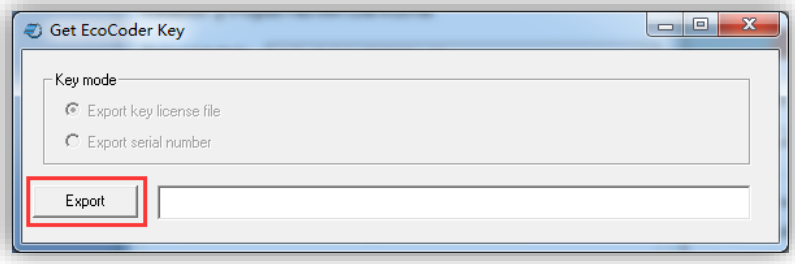
1. Double-click “EcoCoder Loader” on the desktop.

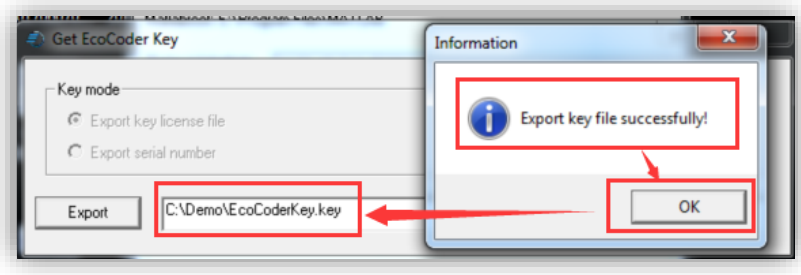


2. Select Tools → Get EcoCoder Key.

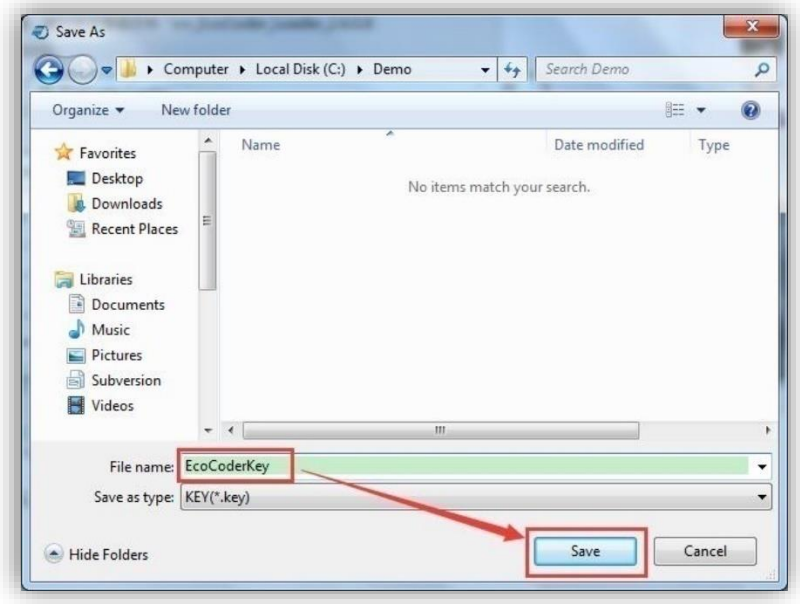


3. Click ‘Export’.





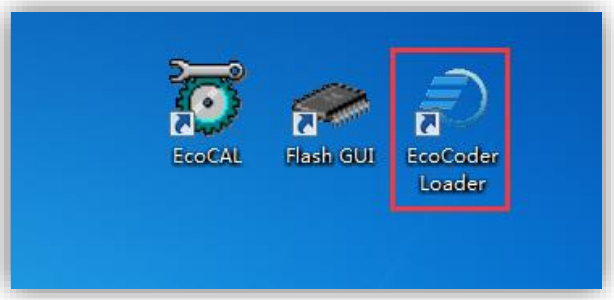
4. Save the key file.



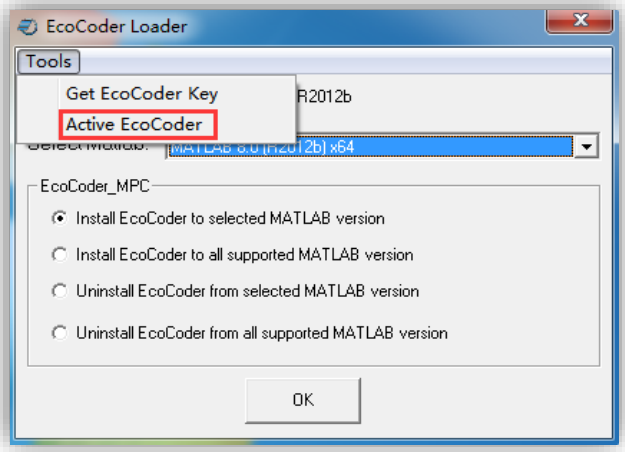
5. Please send the key file to *Support@ecotron.ai* for license file.

2.9.2 Activate EcoCoder by License (.dat) File

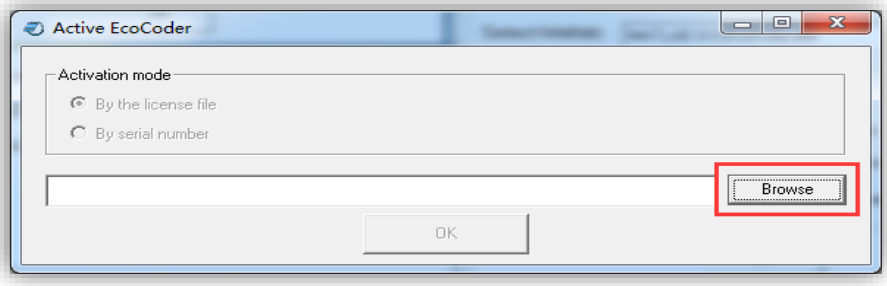
1. Double-click '*EcoCoder Loader*' at the Desktop.



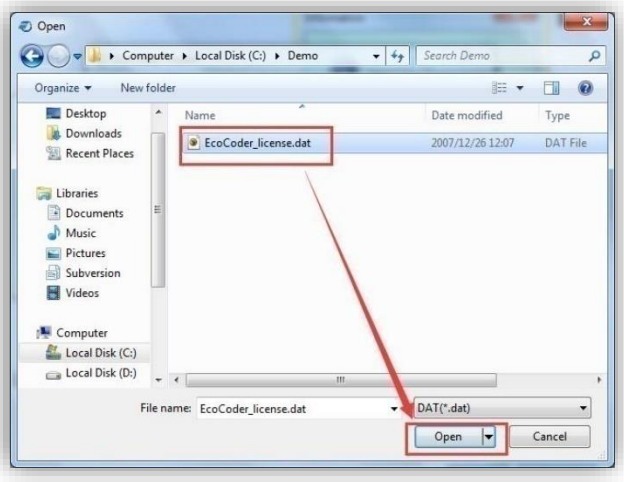
2. Select Tools->Activate EcoCoder.



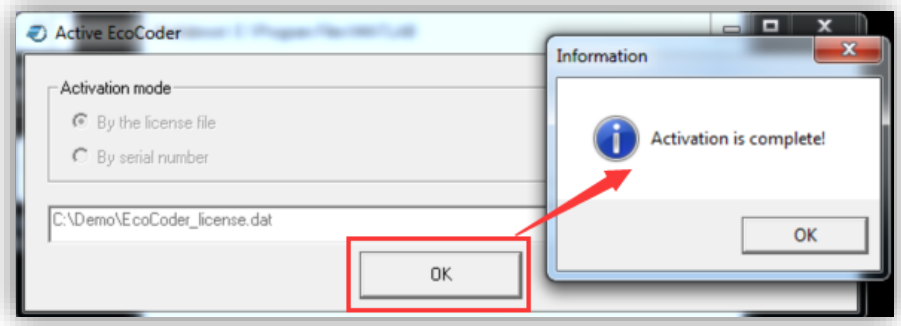
3. Click 'Browse'.



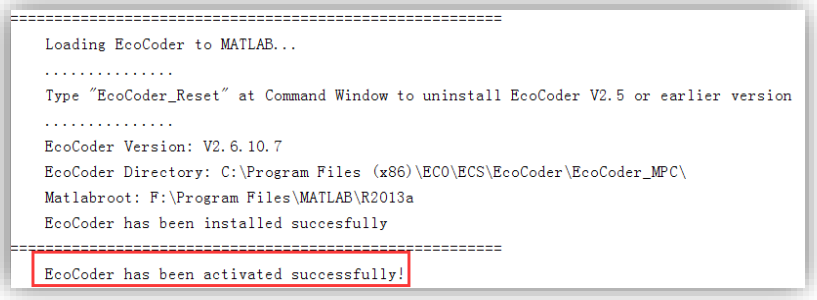
5. Open license file, for example, select '*EcoCoder_license.dat*', then select 'Open'.



6. Click 'OK'. The activation is successful if the pop-up window is displayed as following.



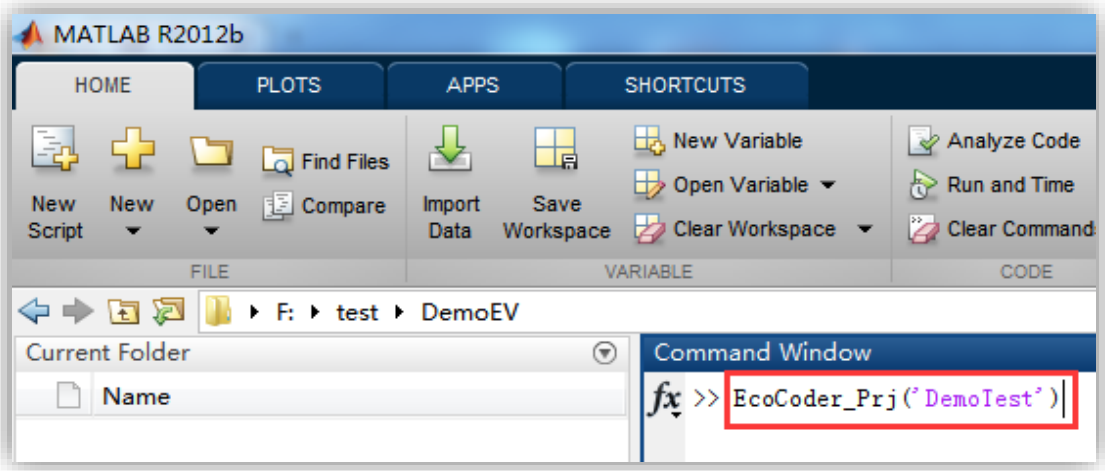
7. If following message shows up in MATLAB command window, EcoCoder should be ready to use.



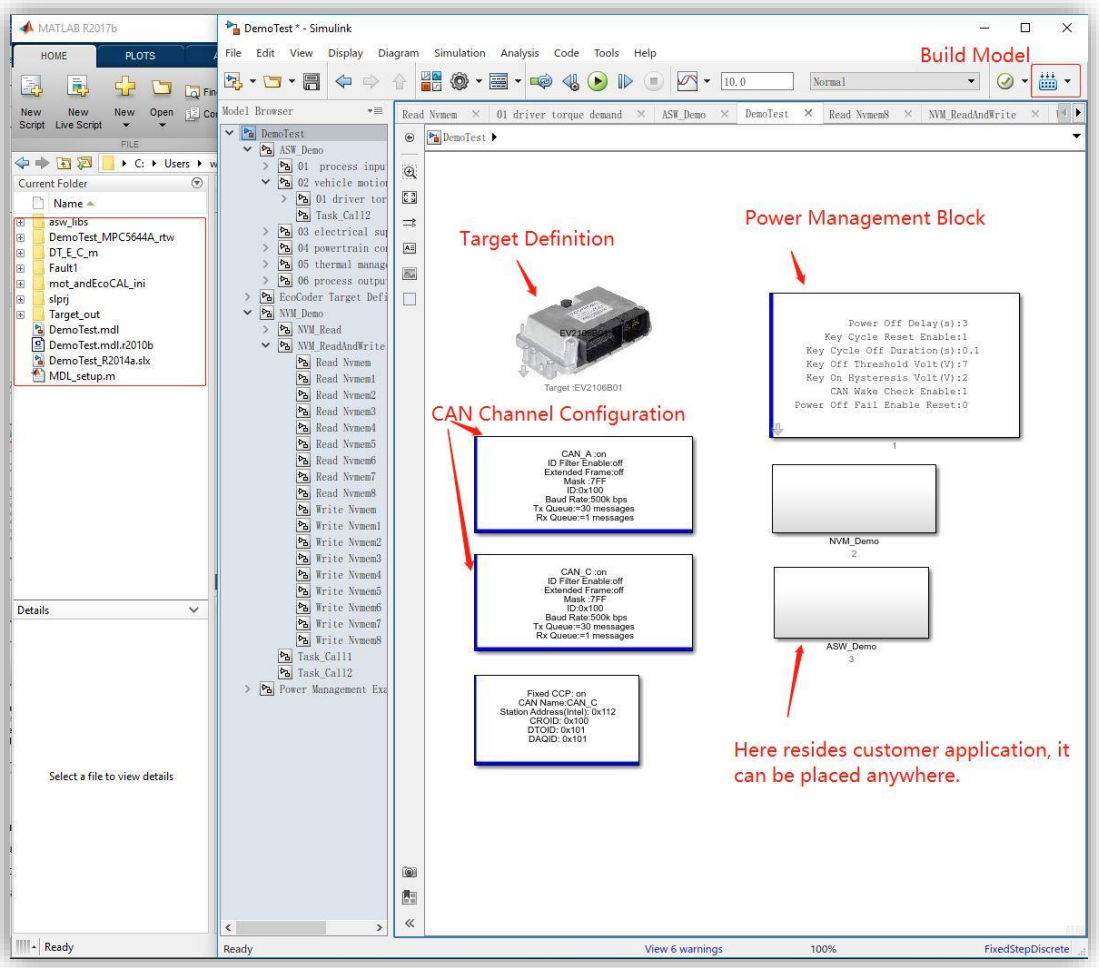
Chapter 3 Quick Start on Application Software

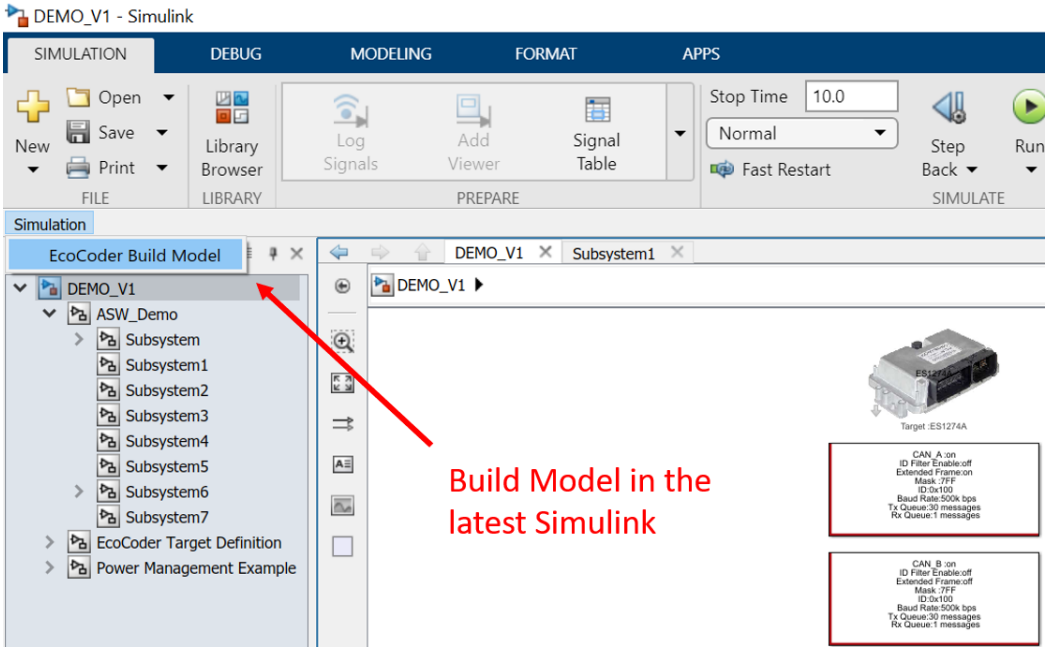
The purpose of this chapter is to give users a quick start to use EcoCoder for control system development. If you don't have any Simulink model yet, and want to have something to start with, or if you want to port your existing Simulink models into EcoCoder platform, this is a quick start. Because EcoCoder will provide an outline (a basic EV control model) for you, to fill in your existing model.

1. Change path to desired folder other than MATLAB installation directory.
2. Type the command '*EcoCoder_Prj ('DemoTest')*' in Command Window.



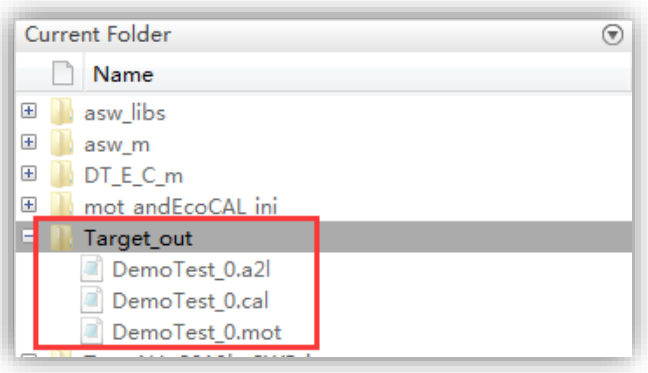
3. A model '*DemoTest.mdl*', and a '*.m*' file as shown in the following figure will be generated. In Simulink, by using shortcut key 'Ctrl + B' or click 'Build Model' button in Simulink task bar, *'.a2l'*, *'.mot', '.cal'* file would be generated.
   (Note: for the latest version of EcoCoder, if users choose CCP type as 'configurable', EcoCoder will only generate *'.a2l'* and *'mot'/'hex'.*)


Build Model
Target Definition
Power Management Block
CAN Channel Configuration
Here resides customer application, it can be placed anywhere.


DEMO_V1 - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
APPS
EcoCoder Build Model
Build Model in the latest Simulink

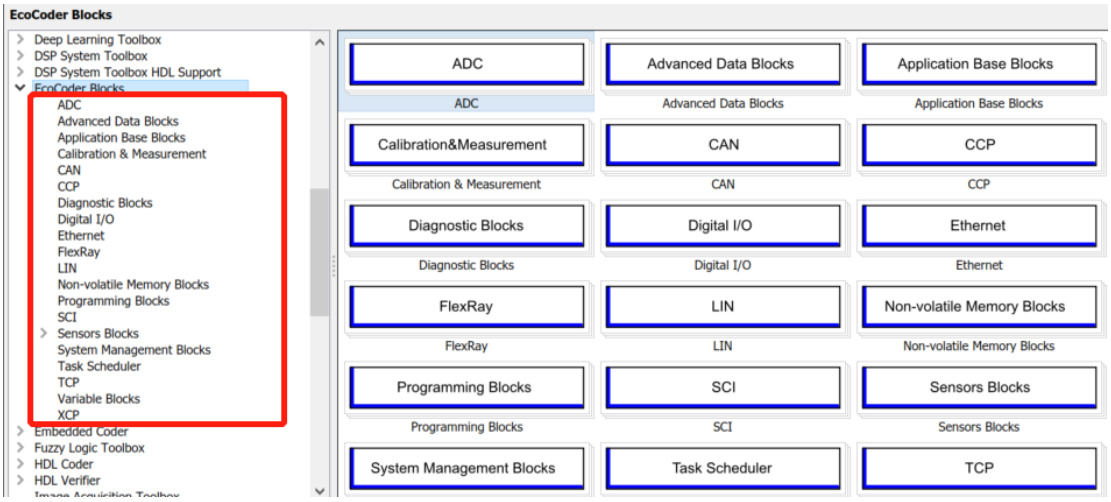
If the ‘CodeWarrior’ window pops up, do not manually intervene, it will automatically compile and close when the process is finished.

As shown in below picture, you can find the generated *‘.mot’*, *‘.cal’*, and ‘*.a2l*’ files.



Chapter 4 EcoCoder Library

The EcoCoder library is an add-on library in Simulink. EcoCoder library mainly provides interfaces for application software to handle I/Os, VCU power, communication and calibration / measurement setup, etc.

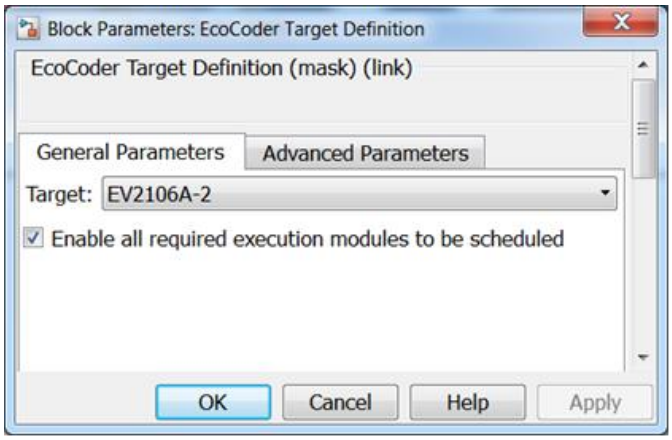


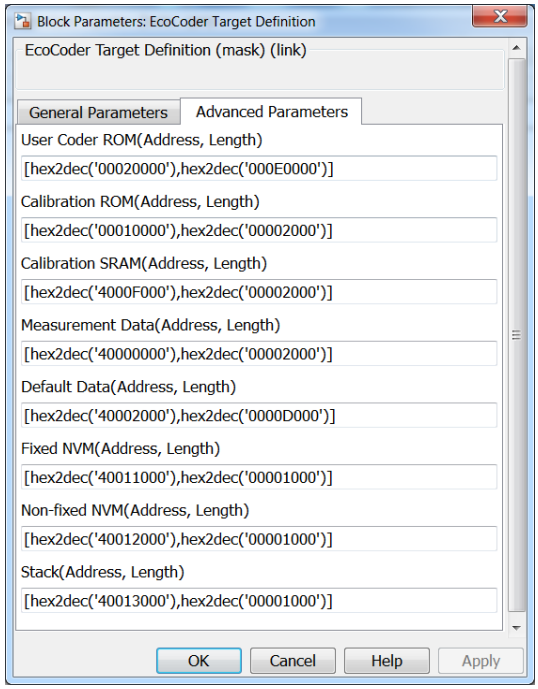
4.1 EcoCoder Target Definition

Folder: EcoCoder Blocks

Description:

EcoCoder Target Setting
Target :EV2106A-2





Under the 'General Parameters' tab, this block defines the specific model of Ecotron VCU that you are using.

Place this block in application model, usually at the top level, to select the VCU model for users' application. The 'Advanced Parameters' tab enables the user to work with part of the ROM and RAM memory addresses. If you would like to do the adjustment regarding all the addresses, please contact Ecotron Tech Support, otherwise, please keep it as default.

Block Parameters:

Parameter Field	Value	Comments/Description
Target	Drop-down list	Pick target VCU
Enable all required execution modules to be scheduled	Check box	If enabled: All subsystems that are not assigned to tasking triggers would be assigned to *L1ms*

		trigger*
Advanced Parameters	Memory addresses (RAM, ROM)	Contact our tech support for adjustments

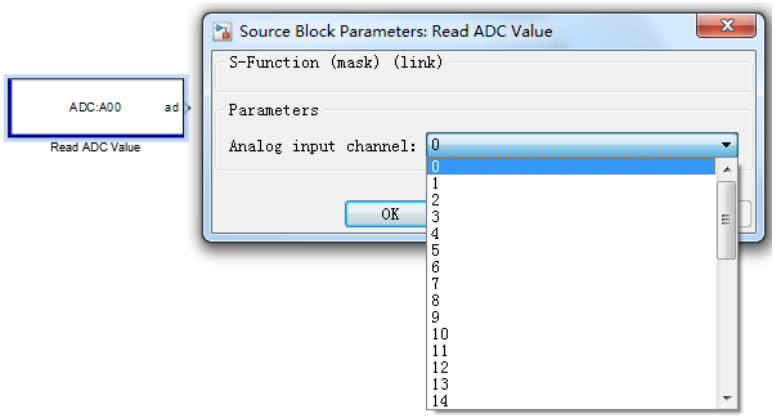
* Please refer to Task Scheduler for knowledge of '*Tasking*'.

4.2 ADC Analog Inputs

4.2.1 Read ADC Value

Folder: EcoCoder Blocks/ADC

Description:



In most cases, there are voltage dividing and shifting circuits on the target VCU hardware, they map the physical voltages being measured into the range that the microcontroller chip(s) can read, usually 0 to 5V. The resolution at which this pre-processed voltage by dividing circuits can be measured depends on the controller chip, usually 10 or 12 bits (1023 or 4095 maximum value, respectively). A reading of 0 represents the minimum voltage specified for these external circuits and a maximum value (1023 or 4095) represents the highest specified voltage.

This EcoCoder block outputs values of the A/D converter channel connected to corresponding physical pin. The output value of this block is the output of AD converter chip (10- or 12-bits binary value).

EcoCoder has predefined input voltage range and resolution of each channel, please refer

to datasheet of the specific VCU.

See examples below:

Channel	ADC Predefined Resolution (bits)	RAW ADC	Raw ADC (binary)
1	10	500	000111110100
2	12	500	00000111110100

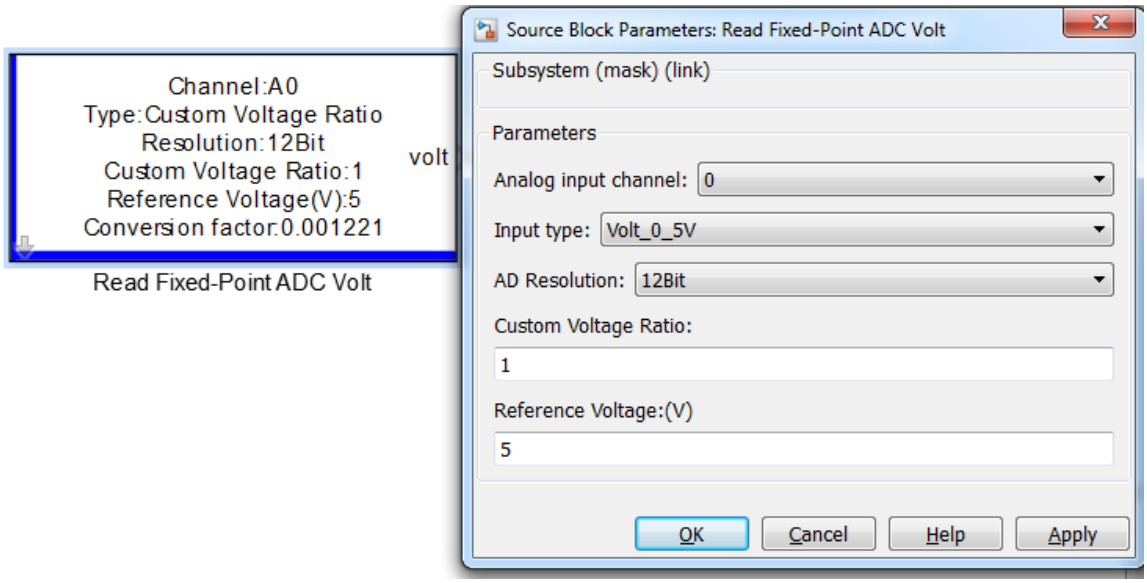
Block Parameters:

Parameter Field	Value	Comments/Description
Analog input channel	0 – n (determined by hardware resource of specific VCU)	Pick specific analog input channel

4.2.2 Read Fixed-Point ADC Volt

Folder: EcoCoder Blocks/ADC

Description:



This block enables user to read the physical voltage at the physical pin on the connector, and block output data type is fixed-point.

Block Parameters:

1. *Analog input channel*: Choose analog channel.

2. *Input type*: Channel type of the voltage input, 4 types are supported: '0-5V', '0-12V', '0-24V' and 'Custom Voltage Ratio'.

3. *AD Resolution*: Please refer to the datasheet of VCU for resolution selection.

4. *Custom Voltage Ratio*: This option is available for the fourth input type 'Custom Voltage Ratio' only. Previously, every voltage input type has fixed resistor divider, as a result, the user just needs to select voltage type. Different resistor divider is introduced in new hardware datasheet, which gives the introduction of new input type and this input option.

5. *Reference Voltage*: By default, it will be set as 5V. **Please do not change.**

Block Output:

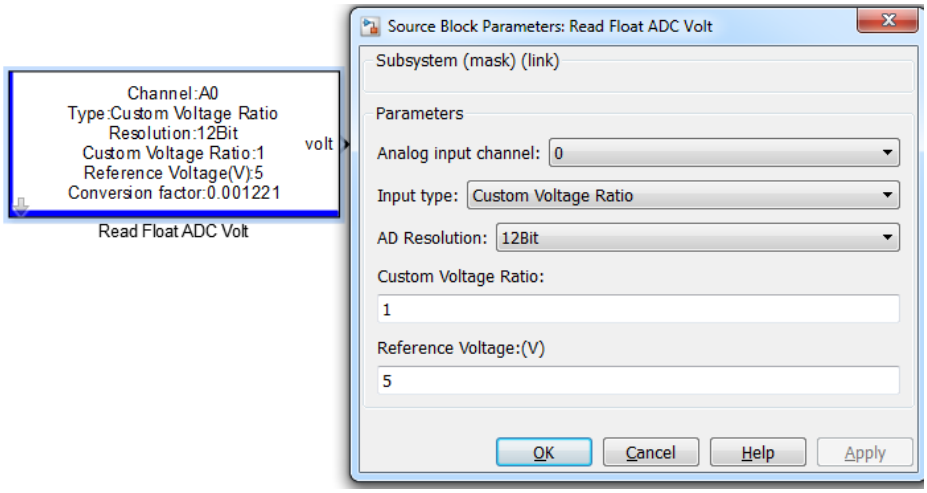
Volt: Physical value of input voltage of specified channel; unit: V; 'single' data type.

(*) For fixed point toolbox advantages, refer to

https://www.mathworks.com/help/simulink/fixed-point.html

(*) Every channel has its unique configuration defined in firmware, please refer to datasheet of VCU and select correct setting for the channel

4.2.3 Read Float ADC Volt



This block enables user to read the physical voltage at the physical pin on the connector, and block output data type is float-point.

Block Parameters:

1. *Analog input channel*: Choose analog channel.

2. *Input type*: Channel type of the voltage input, 4 types are supported: '0-5V', '0-12V', '0-24V' and 'Custom Voltage Ratio'.

3. *AD Resolution*: Please refer to the datasheet of VCU for resolution selection, since different VCUs have different AI channels setup.

4. *Custom Voltage Ratio*: This option is available for fourth input type 'Custom Voltage Ratio' only. Previously, every voltage input type has fixed resistor divider, as a result, the user just needs to select voltage type. Different resistor divider is introduced in new hardware, which explain the introduction of new input type and this input option.

5. *Reference Voltage*: By default, it will be set as 5V. Please do not change.

Block Output:

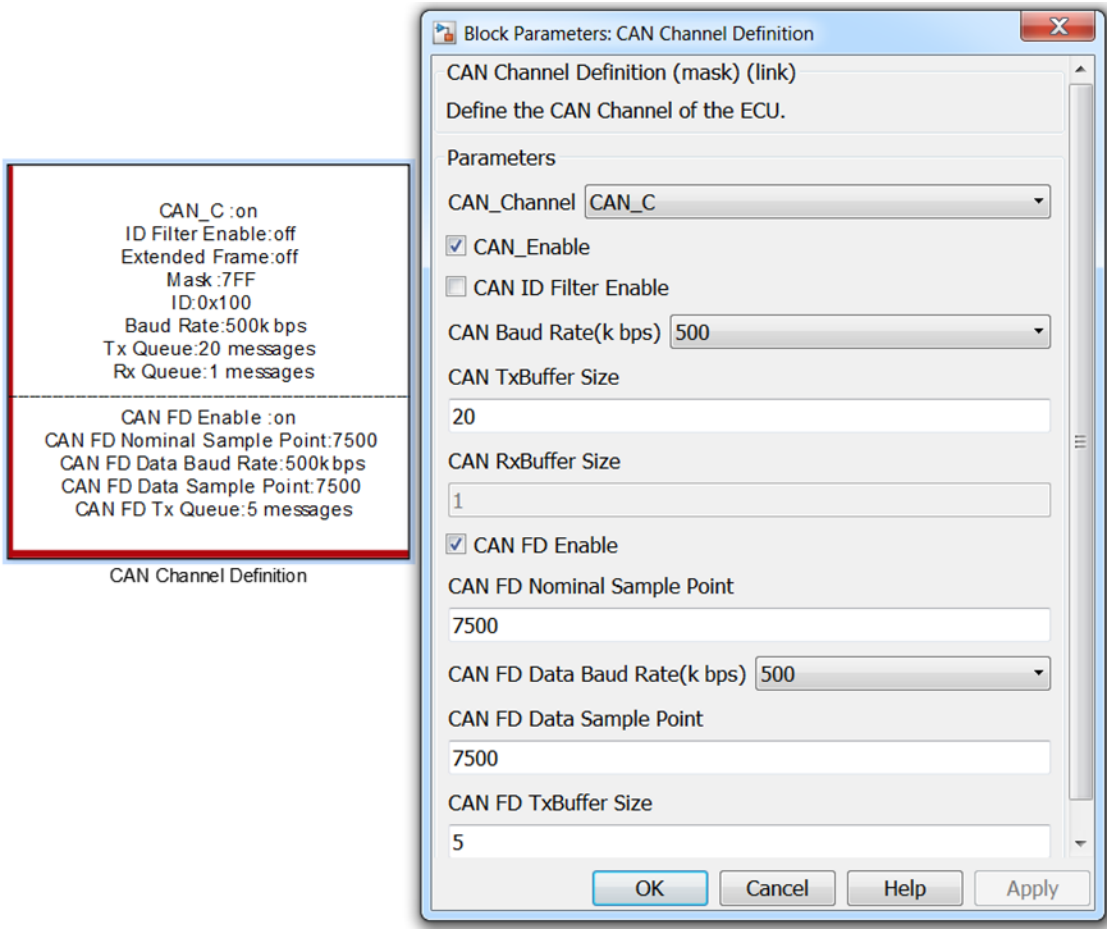
Volt: Physical value of input voltage of specified channel.

4.3 CAN Communication

Please Refer to **Chapter 5 *CAN theory of Ecotron*** before using EcoCoder CAN blocks. Chapter 5, combined with CAN bus communication protocol, will give the user preliminary knowledge of implementing CAN on Ecotron VCU.

4.3.1 CAN Channel Definition

This block is used to initialize the CAN bus configurations. Including CAN channel enable, CAN ID filtering, baud rate settings.



Parameter:

1) CAN Channel: CAN channel selection
2) CAN Enable: enabling the CAN channel
3) CAN ID Filter Enable: When filtering is enabled, the CAN bus is only receiving CAN messages with a specific ID.
4) CAN Extended: enable to receive extended frame only.
    a. Note: if the CAN ID Filter is not enabled, both extended and regular frame will be received.
5) CAN ID Mask (uint32 Hex): Set CAN ID mask
    b. Note: used CAN ID Filter (uint32 Hex) with this option
6) CAN Filter (uint32 Hex): Filters ID. It is a 16-bit variable; user needs to use this with CAN ID Filter (uint32 Hex)
7) CAN Baud Rate (kbps): CAN baud rate configuration
8) CAN TxBuffer Size: send buffer for CAN transmission, minimum value is 2
9) CAN RXBuffer Size: receive buffer for CAN transmission
10) CAN FD Enable: enables CAN FD
11) CAN FD Nominal Sample Point: number of sample points for CAN FD bus, 10,000 represents 100%
12) CAN FD Data Baud Rate (kbps): CAN FD data baud rate
13) CAN FD Data Sample Point: the number of CAN FD Data Sampling points, 10,000 represents 100%
14) CAN FD TxBuffer Size: CAN FD transmit buffer; minimum value is 2

Example:

If a bit in the CAN ID Mask is set to 0, means the CAN node does not care whether if this bit is 1 or 0 in a CAN ID.

1. Example 1
    a. MASK(4-bit): b1101
    b. FILTER(4-bit): b1111

c. The CAN Node is receiving:

i. b1111

ii. b1101

2. Example 2

a. MASK(4-bit): b1001

b. FILTER(4-bit): b1111

c. The CAN Node is receiving:

i. b1111

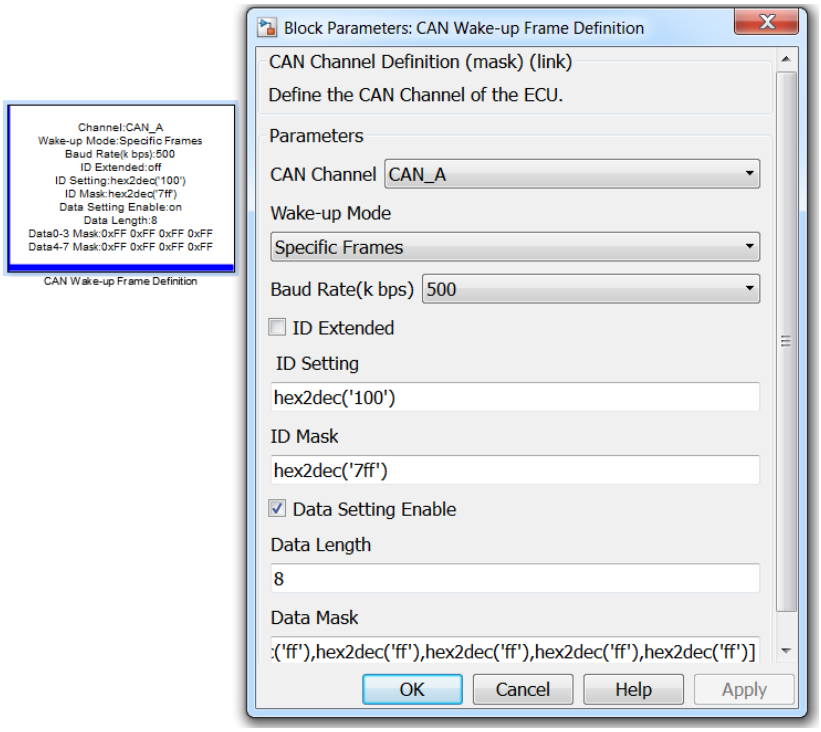
ii. b1101

iii. b1011

iv. b1001

If a bit in the CAN ID Mask is set to 1, means the CAN node does care whether if this bit is 1 or 0 in a CAN ID.

4.3.2 CAN Wake-up Frame Definition



This block is used to define the VCU wake-up CAN message, it is only supported by

several new VCU models. For more information, please consult Ecotron support.

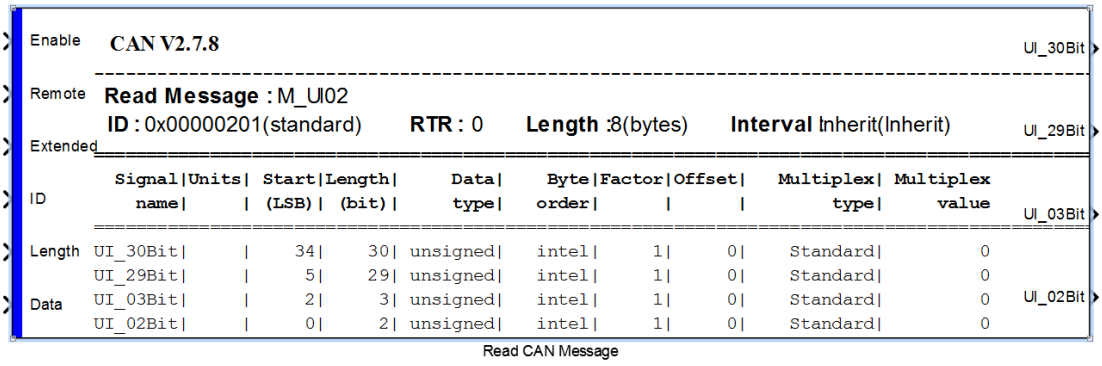
Parameters:

1) CAN Channel: Selecting CAN channel number for this function.
2) Wake-up Mode: Setting the wake-up mode, including Disable (disable CAN wake up function), All Frames (any frame on the specified bus can wake VCU up), and Specific Frames (User specify the frame that can wake up the VCU).
3) Baud Rate: CAN baud rate set up.
4) ID Extended: If checked, the specified VCU-waking-up message will have to use extended CAN ID. If not checked, the message has to use standard CAN ID.
5) ID Setting: Specify the ID here for the wake-up message.
6) ID Mask: The mask for VCU-waking-up message ID.
7) Data Setting Enable: If checked, not only a specific waking-up message ID is needed, but the user also needs to specify the data in the message. Only message with matching ID and data can wake up the VCU. Data can be specified in the following blank.
8) Data Length: Set the wake-up message data length. Only when the data length of the wake-up message matches this value, the message can wake up the VCU.
9) Data Mask: The mask for wake-up message data. The message data bitwise AND with this mask value, if one or more bit of the result of bitwise AND is (are) not 0, the message can wake up the VCU.

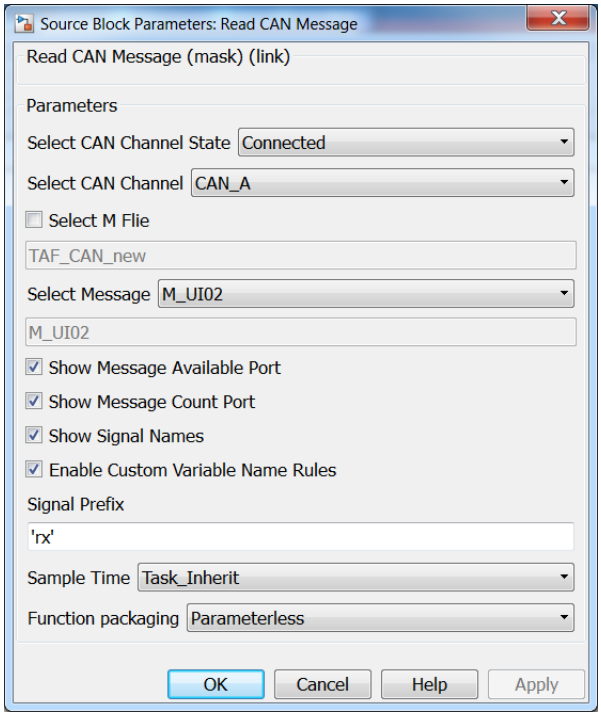
4.3.3 Read Fixed-Point CAN Message

Folder: EcoCoder Blocks/CAN

Description:



Read CAN Message



This block provides CAN messages receiving and unpacking functions. It requires a .m file of CAN message definition to help unpack CAN messages. The generation process (from .dbc file to .m file) is explained in Chapter 5.

Block Parameters

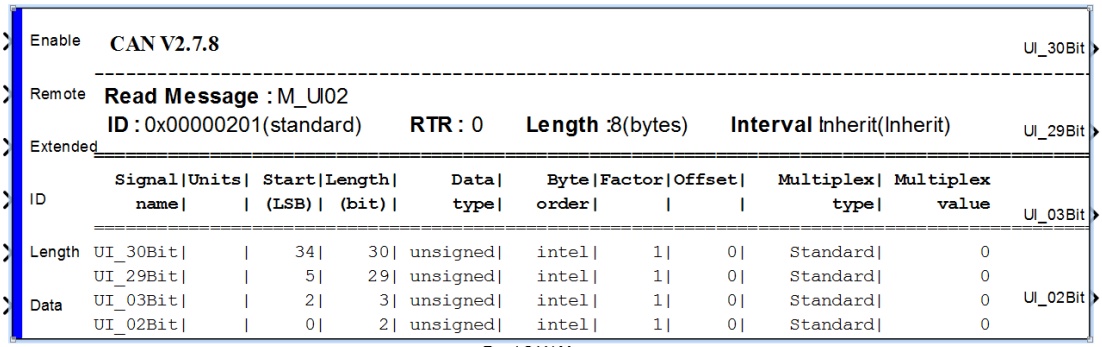
Parameter Field	Value	Comments/Description
Select CAN Channel	Drop-down list	The CAN channel has to be defined before applied.

Select M File	Check box	If checked: please enter the name of m file in the blank space under check box.
Select Message	Drop-down list	Specify CAN message to be received and processed by the block.
Show Message Available Port	Check box	If checked: the block will provide a signal flag to help tell the availability of this CAN message.
Show Message Count Port	Check box	Message counter, if checked: every time the message is received, the counter increments by 1.
Show Signal Name	Check box	If checked: the names of the signals will be displayed.
Define Signal	Check box	If checked: signals parsed out from the block will be cast as measurement variables. *'Show Signal Name'* must be checked before checking this item.
Signal prefix	Alpha-numeric text	Specify prefix to parsed out signal names, remember to use single quote.
Output Data Type: Inherit	Check box	If checked: the data type of the signal is inherited from input data type. If not checked: the signal type is automatically

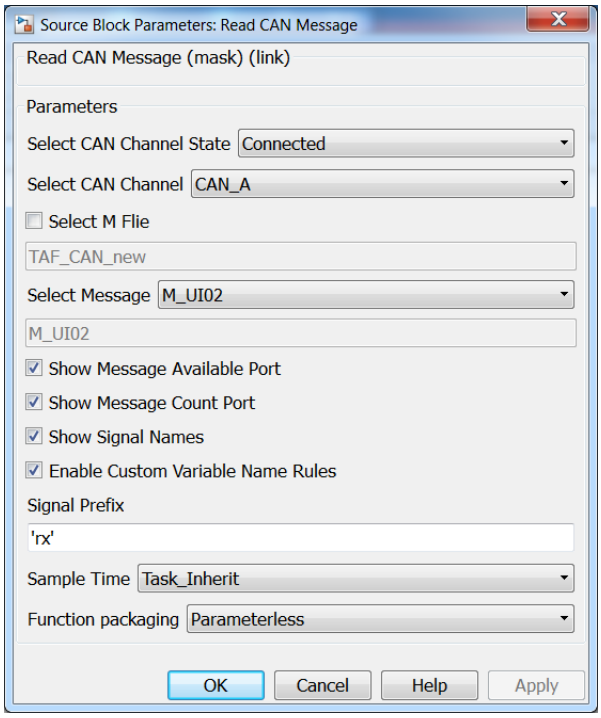
		defined using fixed point tool.
Sample time	Drop-down list	Please refer to section 5.2.5

4.3.4 Read CAN Message

This block is to read CAN messages. Decode CAN messages into signals. All the decoded signals are inherited backwards. This block does not support .dbc files, but it supports .m files converted from .dbc files.



Read CAN Message



Parameter:

1) Select CAN Channel State: Select CAN Channel state, including Connected, Disconnected and DisconnectedOnlyData options.
    a. Connected: output port is disabled, messages are directly sent to the channel
    b. Disconnected: messages sent from the output port
    c. DisconnectedOnlyData: the block is only used as a data converter
2) Select CAN Channel: CAN channel selection
3) Select M file: select the m file from DBC converting tool. Click OK then double click the block to select specific message.
4) Select Message: select specific message
5) Show Message Available Port: check to show the message availability port, output 1 when read a valid message.
6) Show Message Count Port: check to show message counter.
7) Show Signal Names: Enable to show the signal name
8) Enable Custom Variable Name Rules: enable custom variable name rules.
    a. Rules: change the letter after the second underscore from lowercase to upper case, remove the second underscore in the variable name.
9) Signal prefix: show signal prefix on the signal name
10) Sample time: Simulation sample time

Input:

When the channel state is “Disconnected”, the input ports are listed as below. There is no port available when the channel state is “Connected”.

1) Enable: Enable CAN message decoding
2) Remote: this flag is set to 1 when sending a remote frame
3) Extended: this flag is set to 0 when sending an extended frame
4) ID: CAN message ID
5) Length: data length
6) Data: data in the message
7) Function Packaging:

a. Parameterless: use C functions without parameter. This option has high code execution efficiency.

b. Parameterized: use C functions with parameters. This option offers more stable compilation.

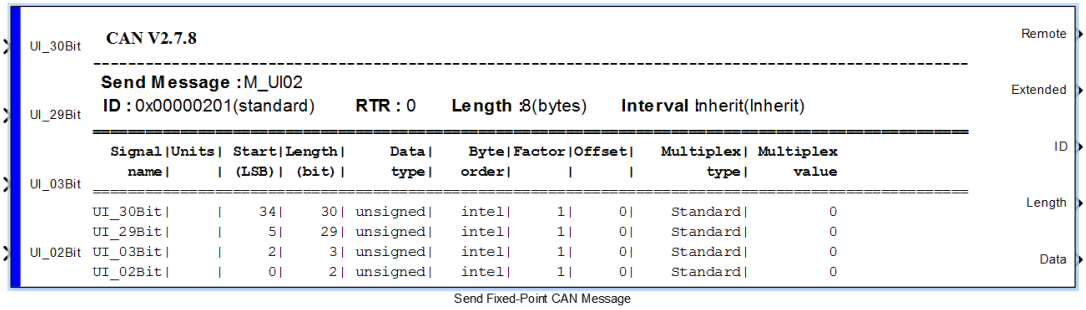
Output:

1) The signal needs to be transmitted. The unit should be the actual physical unit.

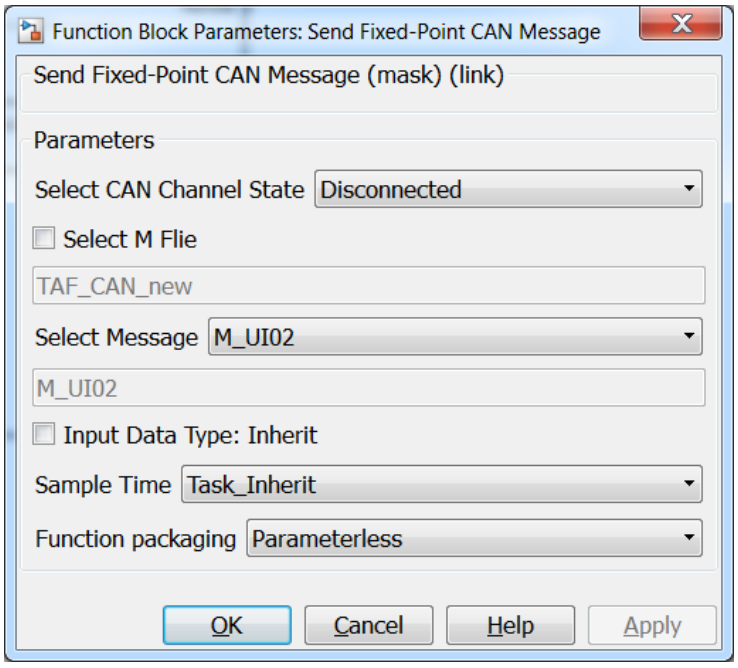
4.3.5 Send Fixed-Point CAN Message

Folder: EcoCoder Blocks/CAN

Description:



Send Fixed-Point CAN Message



Parameter Field	Value	Comments/Description
Select CAN Channel State	Drop-down list - Connected - Disconnected	Connected: Message will be sent out from the CAN channel selected under 'Select CAN Channel' Disconnected: User will have to assign the output port manually.
Select CAN Channel	Drop-down list	CAN channel selection
Select M File	Check box	If checked: please enter the name of m file in the blank space under check box.
Select Message	Drop-down list	Specify CAN message to be sent and processed by the block.
Input Data Type: Inherit	Check box	If checked: the data type of the signal is inherited from

		input data type. If not checked: the signal type is automatically defined using fixed point tool.
Sample time	Drop-down list	Please refer to section 5.2.5

Block Inputs:

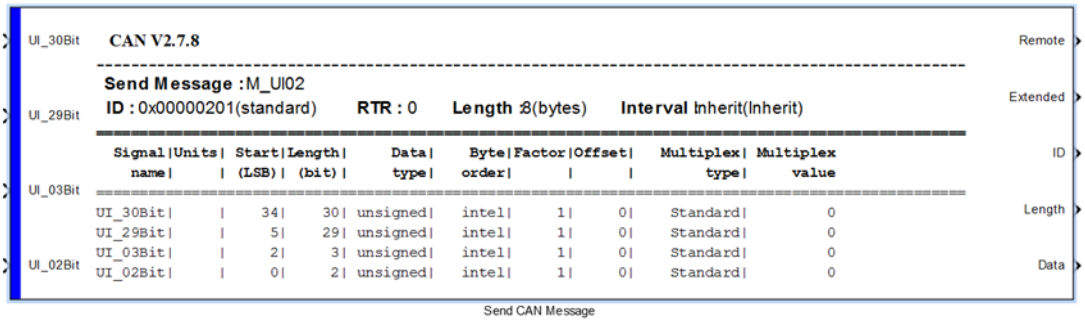
Signals to be sent out.

Block Outputs (if 'Disconnected' is selected under 'Select CAN Channel State'):

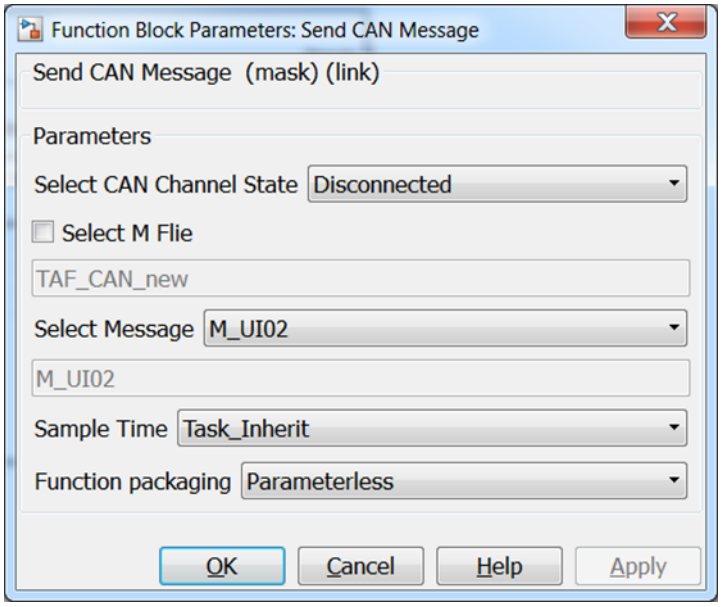
1. Remote: frame type- 1 is remote frame, 0 is standard frame
2. Extended: frame type- 1 is extended frame, 0 is standard frame.
3. ID: message ID.
4. Length: message data length.
5. Data: message data

4.3.6 Send CAN Message

This block is used for sending CAN messages. It takes the .m file converted from .dbc file and send CAN messages accordingly.



Send CAN Message



Parameter:

1) Select CAN Channel State: Select CAN Channel state, including Connected, Disconnected and DisconnectedOnlyData options.
    a. Connected: output port is disabled, messages are directly sent to the channel
    b. Disconnected: messages sent from the output port
    c. DisconnectedOnlyData: the block is only used as a data converter
2) Select CAN Channel: CAN channel selection
3) Select M file: select the m file from DBC converting tool. Click OK then double click the block to select specific message.
4) Select Message: select specific message
5) Sample time: select simulation sample time
6) Function Packaging:
    d. Parameterless: use C functions without parameter. This option has high code execution efficiency.
    e. Parameterized: use C functions with parameters. This option offers more stable compilation.

Input:

1) The signal needs to be transmitted. The unit should be the actual physical unit.

Output:

2) When channel state is at “Disconnected”:

   a. Remote: this flag is set to 1 when sending a remote frame

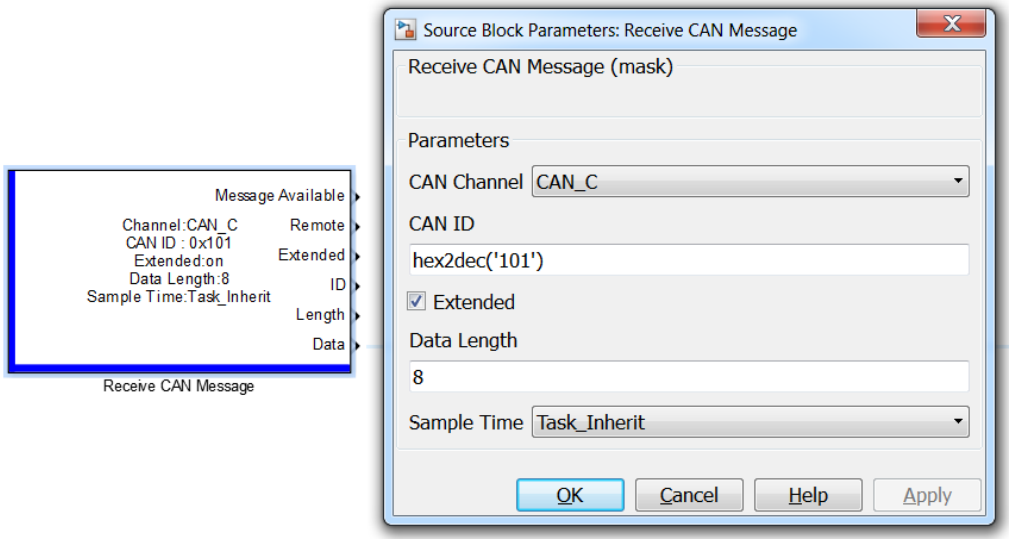
   b. Extended: this flag is set to 1 when sending an extended frame

   c. ID: CAN message ID

   d. Length: data length

   e. Data: data in the message

4.3.7 Receive CAN Message



Block Parameters:

1) CAN channel: CAN channel selection.

2) CAN ID: The ID of the message to be received.

3) Extended: Message type to be received. If checked: extended frame. Otherwise, standard frame.

4) Data Length: The data length of the to-be-received message.

5) Sample Time: Define the task scheduling time of this block being triggered.

Block Outputs:

1) Message Available: Flag for message availability, 1 stands for message valid and

available.

2) Remote: Flag for frame type, 1 stands for remote frame. 0 stands for data frame.

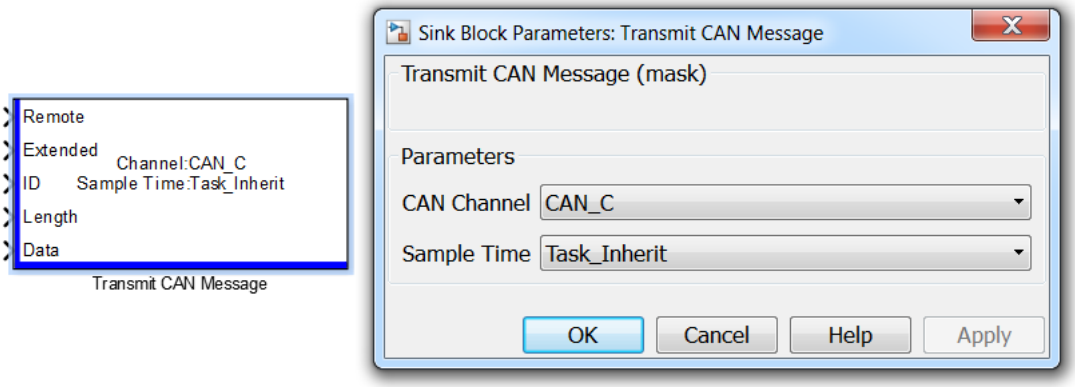
3) Extended: Flag for frame type, 1 stands for extended frame. 0 stands for standard frame.

4) ID: Message ID.

5) Length: Message data length.

6) Data: Message data.

4.3.8 Transmit CAN Message



Block Parameters:

1) CAN Channel: Channel selection

2) Sample Time: Define the task scheduling time of this block.

Block Inputs:

1) Remote: Flag for frame type, 1 stands for remote frame. 0 stands for data frame.

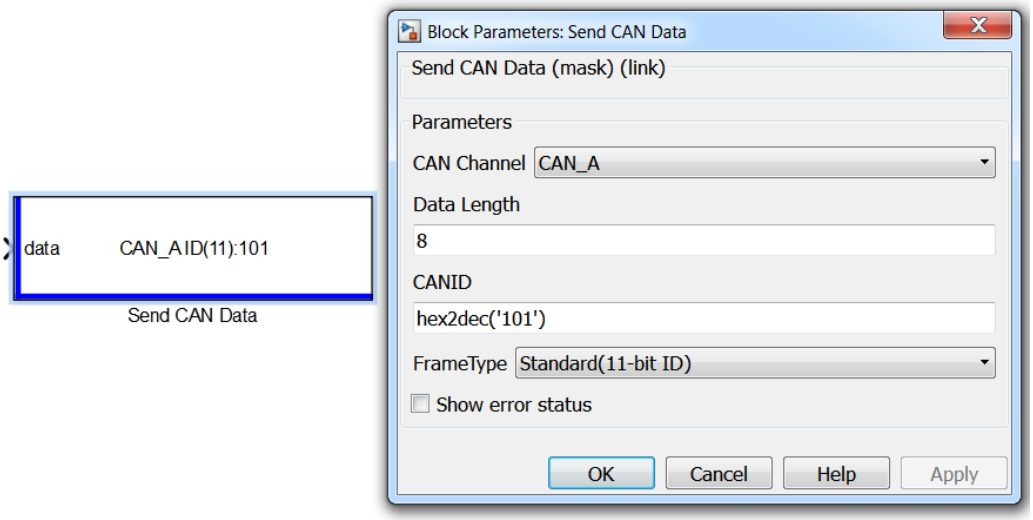
2) Extended: Flag for frame type, 1 stands for extended frame. 0 stands for standard frame.

3) ID: Message ID.

4) Length: Message data length.

5) Data: Message data.

4.3.9 Send CAN Data



Block Parameters:

1) CAN Channel: CAN channel selection.
2) Data Length: Message data length, in bytes.
3) CANID: The ID of the message to be sent. HEX value.
4) Frame Type: Drop-down list for frame type selection.

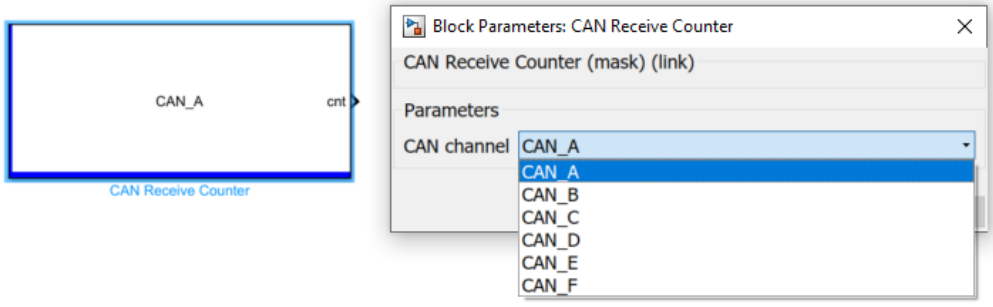
Block Inputs:

1) data: The message data to be sent.

4.3.10 CAN Receive Counter

Folder: EcoCoder Blocks/CAN

Description:



This module can be used to count the number of frames received by specified CAN channel.

Block Parameters

Parameter Field	Value	Comments/Description
CAN channel	Drop-down list	Specify the channel to be monitored

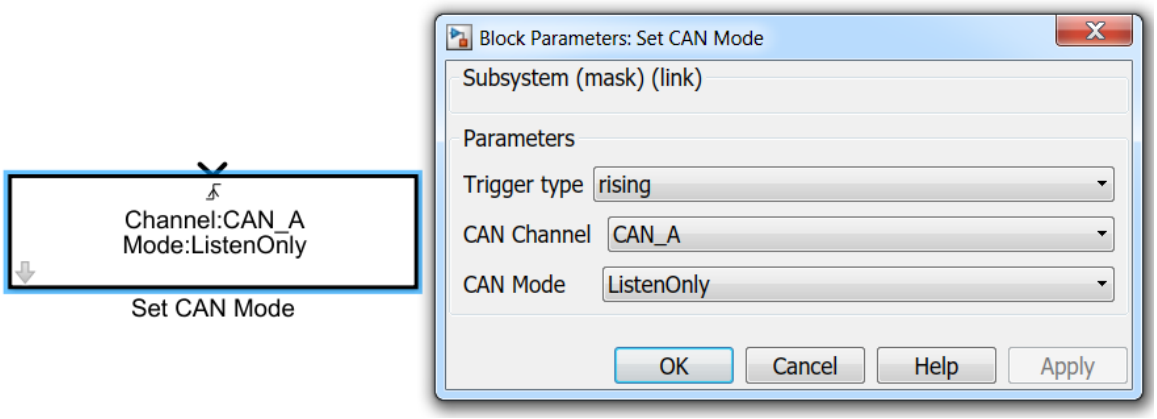
Block Output:

cnt: If the selected channel receives one frame, cnt value increments by 1.

4.3.11 Set CAN Mode

Folder: EcoCoder Blocks/CAN

Description:



This module can be used to switch CAN operating mode between '*listen only*' and '*normal*'.

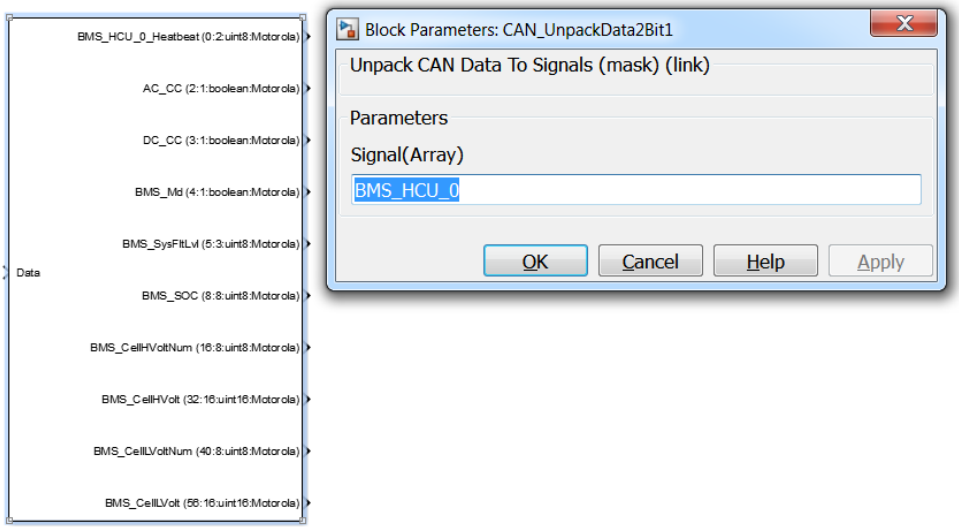
Block Parameters

Parameter Field	Value	Comments/Description
Trigger type	Drop-down list	Trigger type selection
CAN Channel	Drop-down list	Specify CAN channel to be controlled
CAN Mode	Drop-down list	Specify the CAN mode to be triggered by the block

Block Input:

Trigger signal: the signal input to trigger the execution of this block.

4.3.12 Unpack CAN Data to Signals



Block Parameter:

1) Signal (Array): the signal definition matrix of CAN frame.

Block Inputs:

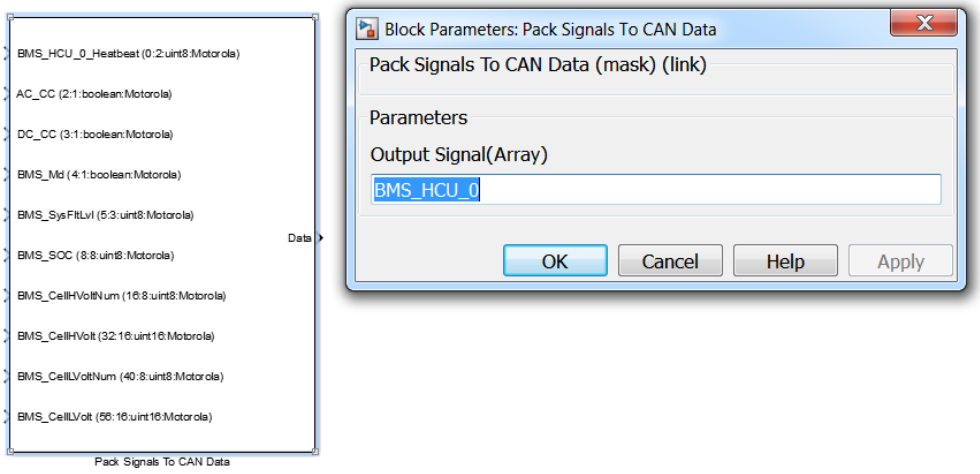
1) Data: the message data to be unpacked.

Block Outputs:

1) Unpacked signals from the CAN data.

4.3.13 Pack Signals to CAN Data

Pack CAN signals to CAN message, usually used together with *Send CAN Data* block.



Block Parameters:

1) Out Signal (Array): The definition array of the signals to be packed.

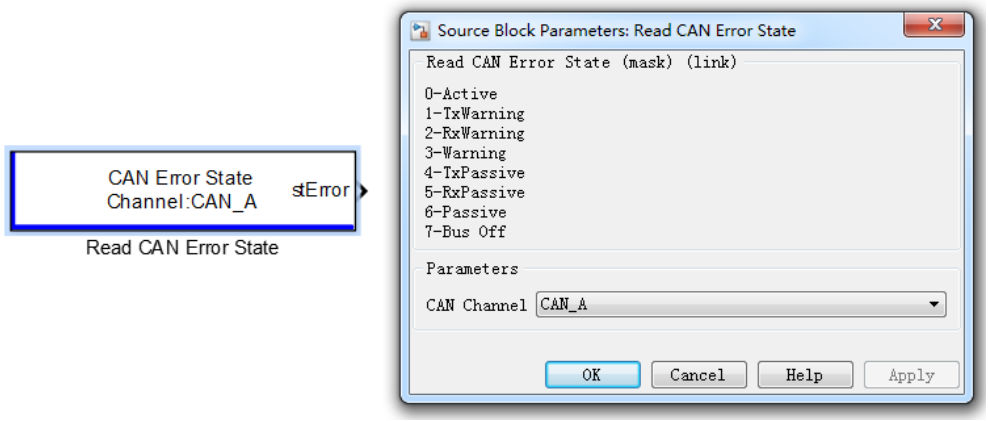
Block Inputs:

1) Signals to be packed, values are in HEX.

Block Output:

1) Data: the packed CAN message data.

4.3.14 Read CAN Error State

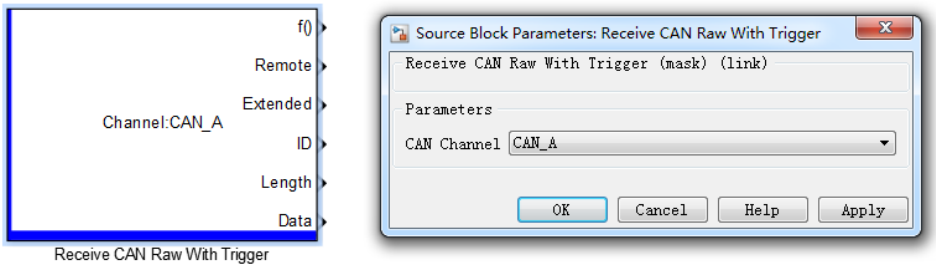


This block diagnoses errors happen in CAN channels.

Block Parameters:

1) CAN Channel: Channel selection

4.3.15 Receive CAN Raw with Trigger



Block Parameter:

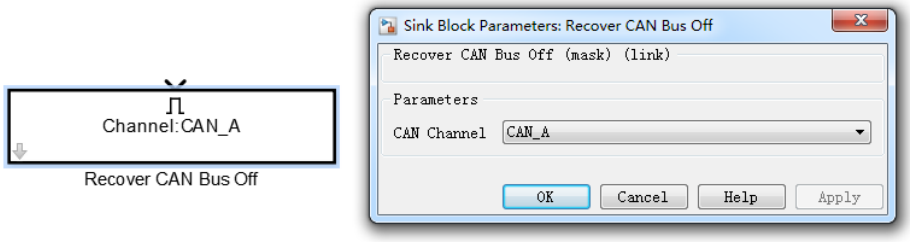
1) CAN Channel: channel selection

Block Output:

1) f(): trigger event
2) Remote: set to 1 to use remote frame, set to 0 to use data frame
3) Extended: set to 1 to use extended frame, set to 0 to use standard frame
4) ID: message ID
5) Length: data length
6) Data: data in the message

4.3.16 Recover CAN Bus Off

This block recovers CAN Bus off errors.



Block Parameters:

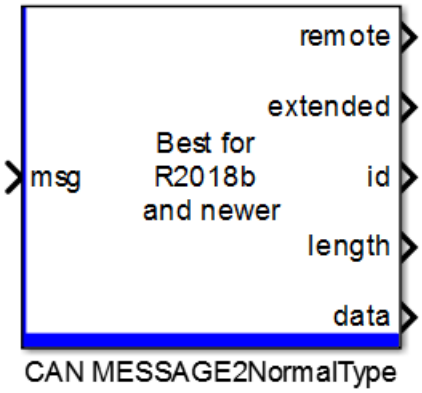
1) CAN Channel

Block Input

1) Trigger signal: Bus off recovered if the signal returns 1.

4.3.17 CAN MESSAGE2NormalType

Along with Transmit CAN Message block, this block can pack and send message with Simulink Embedded Coder by execute the following steps: Embedded Coder->Embedded Targets->CAN Pack. CAN Pack block is recommended to use in MATLAB R2018b (or higher version) since the CAN Pack works more efficiently in the later versions.



Block Input：

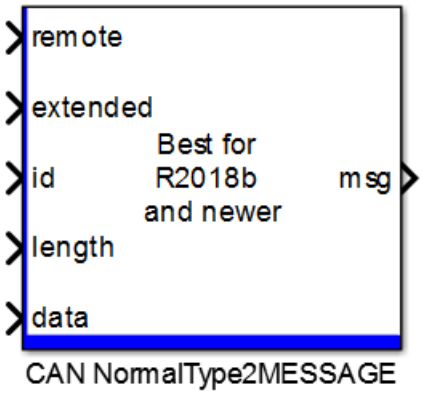
1) msg：Simulink default CAN_MESSAGE type.

Block Output：

1) remote：Remote frame:1 Data frame: 0
2) extended：Extended frame:1 Standard frame: 0
3) id
4) length
5) data

4.3.18 CAN NormalType2NormalType

Along with Receive CAN Message block (or Receive CAN Raw With Trigger), this block can unpack and receive message with Simulink Embedded Coder by execute the following steps: Embedded Coder->Embedded Targets-> CAN Unpack. CAN UnPack block is recommended to use in MATLAB R2018b (or higher version) since the CAN Pack works more efficiently in the later versions.



Block Input：

1) remote：Remote frame:1 Data frame: 0。
2) extended：Extended frame:1 Standard frame: 0
3) id：
4) length

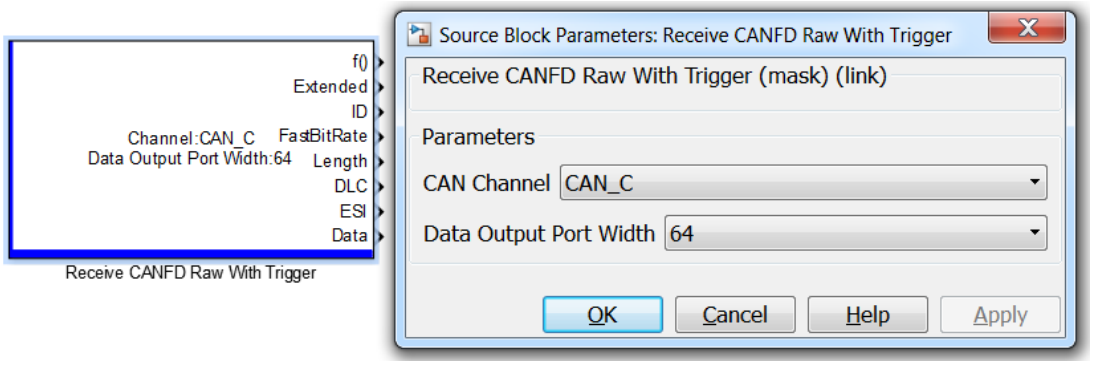
5) data

Block Output：

1) msg：Simulink default CAN_MESSAGE type.

4.3.19 Receive CANFD Raw With Trigger

This block receives CANFD with Trigger



Block Parameter：

1) CAN Channel

2) Data Output Port Width

Block Output：

1) f(): trigger event

2) Extended：Extended frame:1 Standard frame: 0

3) ID: message ID

4) FastBitRate: Changeable rate, Changeable: 1, Unchangeable: 0

5) Length: the length of the date

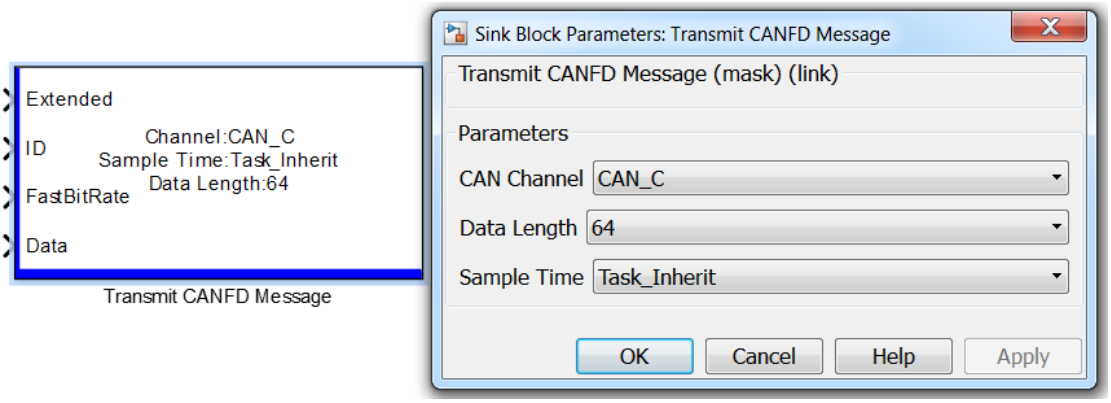
6) DLC：Data Length Code.

7) ESI：Error state, Passive Error:1, Active Error: 0

8) Data：Data Output Port Width function can be used to set up the port width.

4.3.20 Transmit CANFD Message

This block can be used to send CAN FD data. Along with Send CAN Message block, it can send certain signal value when the channel is disconnected or DisconnectedOnlyData.



Block Parameter:

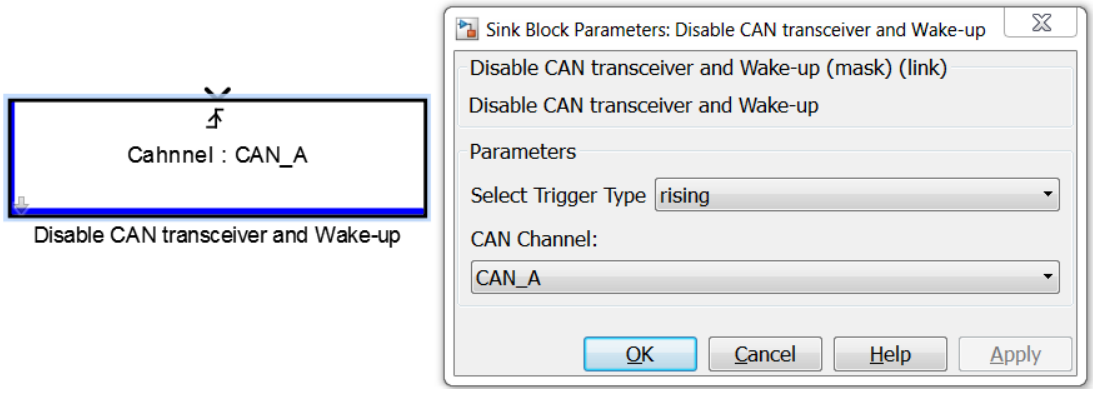
1) CAN Channel
2) Data Length
3) Sample Time

Block Input:

1) Extended：Extended frame:1 Standard frame: 0
2) ID
3) FastBitRate：Changeable rate, Changeable: 1, Unchangeable: 0
4) Data: Data Output Port Width function can be used to set up the port width.

4.3.21 Disable CAN transceiver and Wake-up

This block can turn off CAN Wake-up in when powered off



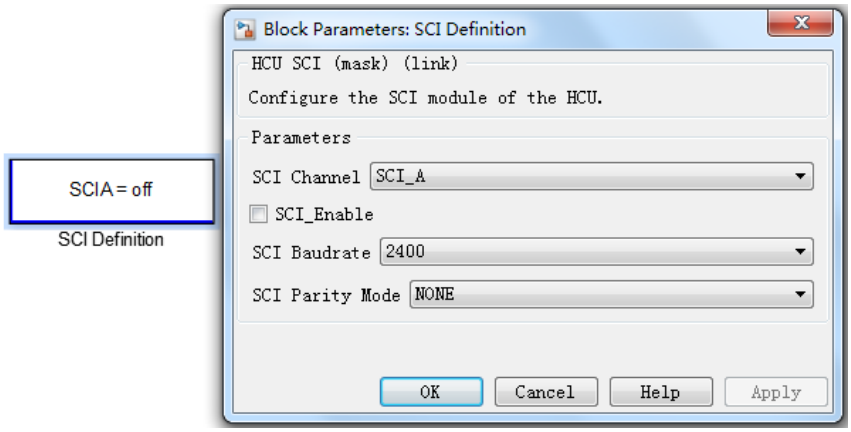
Block Parameter:

1) Select Trigger Type
2) CAN Channel

4.4 Serial Communication Interface (SCI) Block

The SCI mode includes SCI_RxData and SCI_TxData. Currently, only SCI_A channel is supported.

4.4.1 SCI Definition



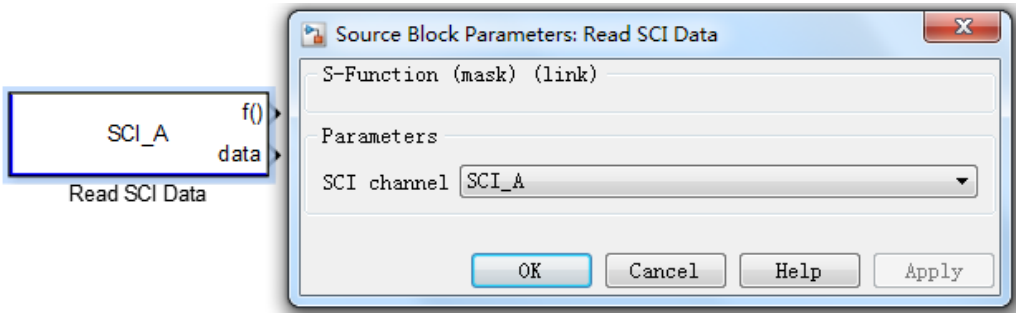
Block Parameters:

3. SCI Channel: Communication channel selection.
4. SCI_Enable: Enable selected channel.
5. SCI Baud rate: Channel baud rate setup.

6. SCI Parity Mode: Parity check mode setup.

4.4.2 Read SCI Data

This block enables the VCU to read data from specific SCI port.



Block Parameter:

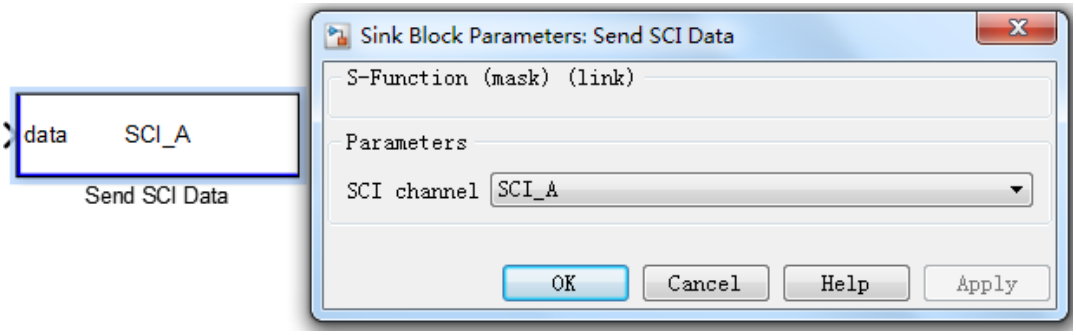
SCI_Channel: SCI communication channel selection.

Block Outputs:

1. f (): Flag for receiving data. If data received, the flag will be 1. This signal can be used as a trigger signal.
2. Data: Output received 8-bit data.

4.4.3 Send SCI Data

This block will send SCI data to assigned channel.



Block Parameter:

SCI_Channel: SCI channel selection.

Block Input:

Data: The 8-bit data to be sent out.

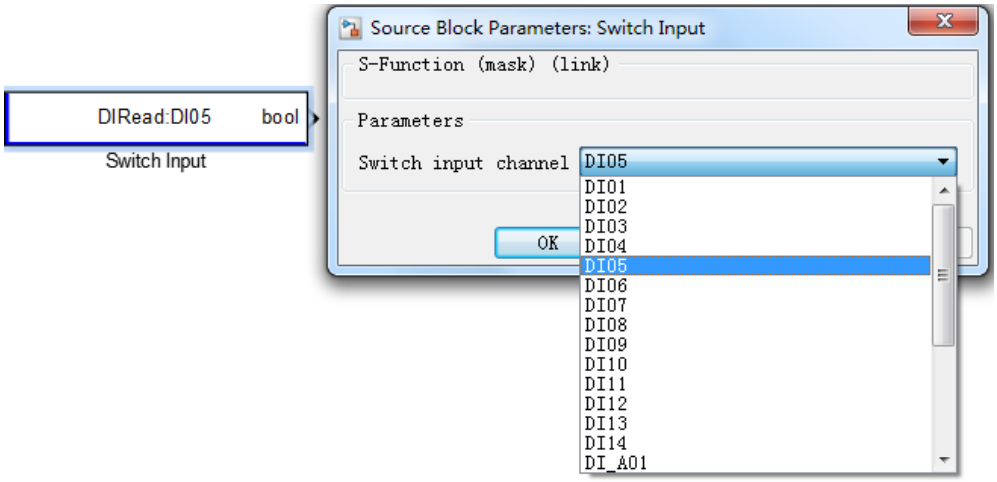
4.5 Digital I/O

These blocks are used for measuring digital input/output. Including Switch Input, Switch Output, PWM input and PWM output.

4.5.1 Switch Input

Folder: EcoCoder Blocks/Digital I/O

Description:



This block reads the physical voltage level of digital input channels and output Boolean variable to application layer.

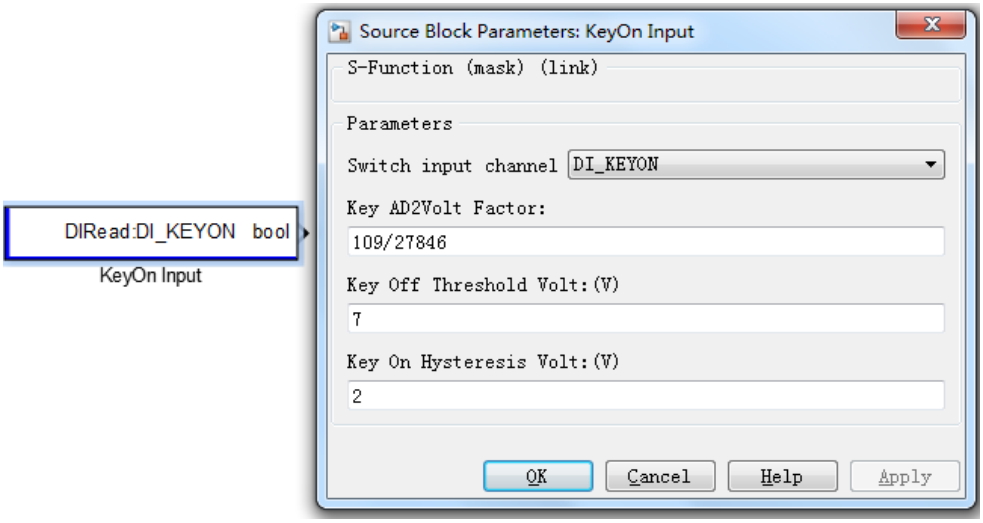
Block Parameters:

Parameter Field	Value	Comments/Description
Switch input channel	Drop-down list	Digital input channel selection

4.5.2 KeyOn Input

Folder: EcoCoder Blocks/Digital I/O

Description:



KeyOn signal is recommended to be used for powering up and shutting down the VCU.

For different VCUs, KeyOn signal inputs are different (refer to the VCU datasheet to confirm the KeyOn signal input type) - if KeyOn signal is digital input, leave the configuration as default; If KeyOn signal is read through analog input channel, user will have to configure factor according to voltage divider parameter - for this parameter, please refer to VCU datasheet.

Block Parameters

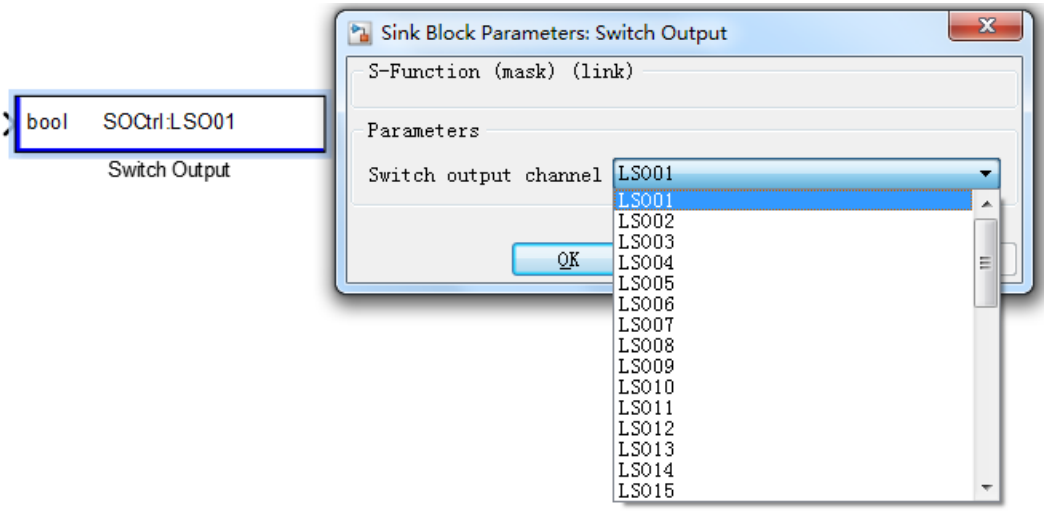
Parameter Field	Value	Comments/Description
Switch input channel	Drop-down list	Only one channel selectable for KeyOn
Key AD2Volt Factor	Numeric	The voltage factor for KeyOn voltage detection (only valid when KeyOn is read from AI)
Key Off Threshold Volt	Numeric	If the input voltage is lower than this value, output is '0'.

KeyOn Hysteresis Volt	Numeric	The hysteresis value between upper and lower threshold. If the interpreted voltage is larger than the sum of 'Key Off Threshold Volt' value and this value, output is '1'.

4.5.3 Switch Output

Folder: EcoCoder Blocks/Digital I/O

Description:

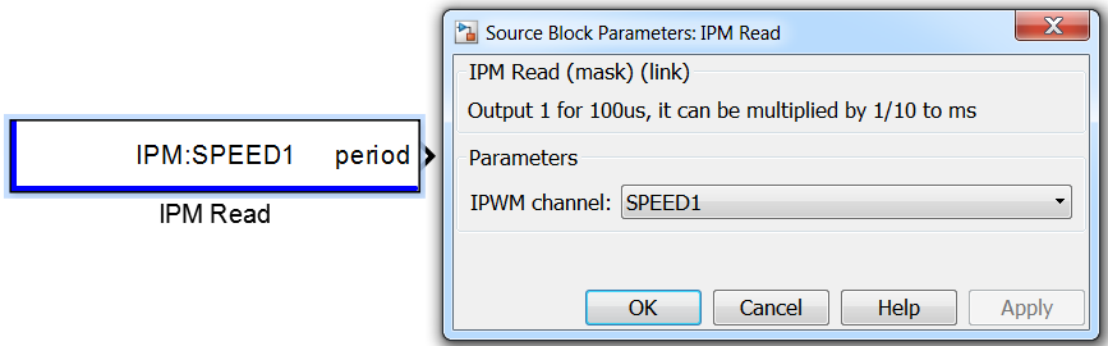


Block Parameters

Parameter Field	Value	Comments/Description
Switch output channel	Drop-down list	Select switch channels to be controlled
Input	Numeric (bool)	0 or 1, switch control value

4.5.4 IPM Read

This block is used for monitoring frequency input from specific channel. Output is period of the signal, can be used for calculating the frequency.



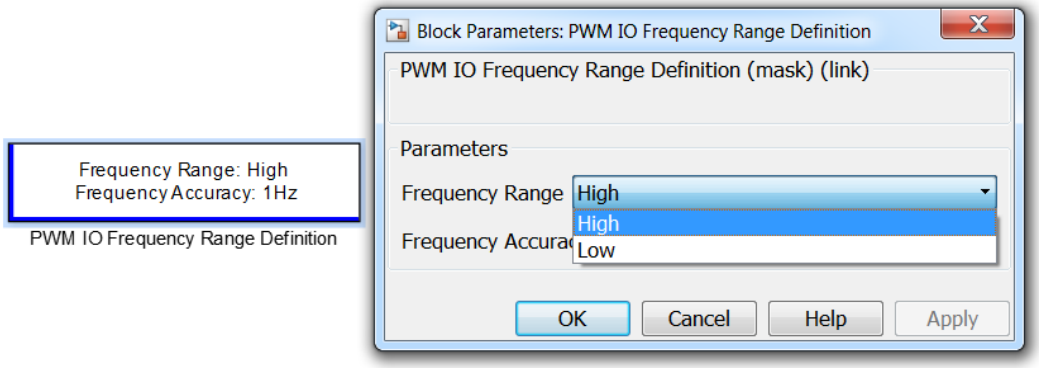
Parameter:

1) IPM Channel: digital channel selection

Output:

2) Period: period of specific channel, 1 means 0.1ms.

4.5.5 PWM IO Frequency Range Definition



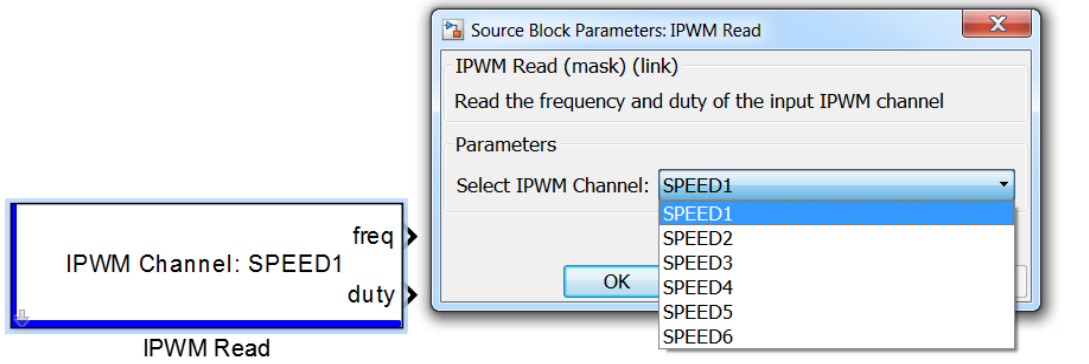
This block defines the frequency range and accuracy of PWM IOs. If it is not implemented in the model, the accuracy will be default value, 1 Hz.

Block Parameters:

1) Frequency Range: Frequency range selection, changes in this option will alter the frequency range and accuracy of all the frequency related block in the model.

2) Frequency Accuracy: Accuracy adjustment. Value in the box means the unit frequency for outputs/inputs of frequency related blocks. For example, if the Frequency Accuracy is 0.01 Hz, it means that when frequency related block outputs/inputs value is 5, the actual physical frequency value is 5x0.01Hz = 0.05 Hz.

4.5.6 IPWM Read



Block Parameter:

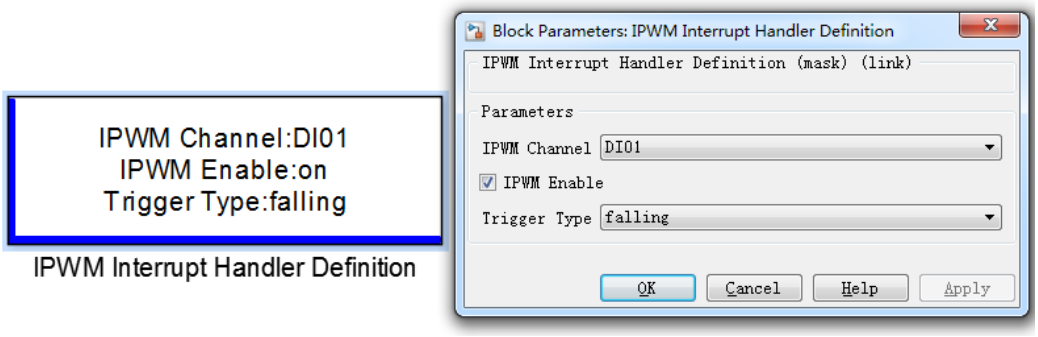
1) Select IPWM Channel: PWM inputs channel selection

Block Outputs:

1) freq: the input PWM frequency of the signal. The unit is related to PWM IO Frequency Range Definition
2) duty: the input PWM signal duty cycle. 0-10000 in this block is 0-100%

4.5.7 IPWM Interrupt Handler Definition

This module is used to initialize the configuration of the switch input channel using interrupt mode to measure frequency and duty cycle.

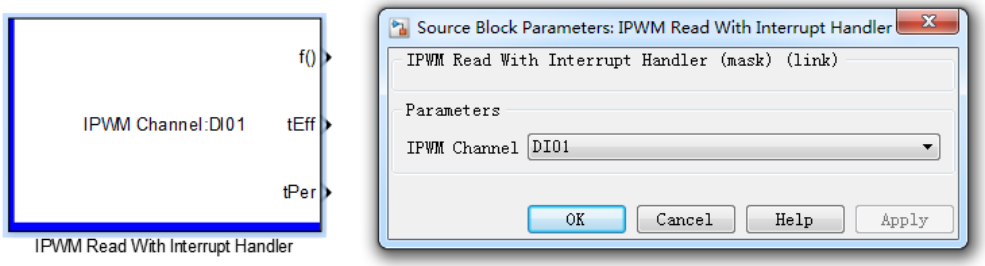


Block Parameter:

1) IPWM Channel
2) IPWM Enable
3) Trigger Type: Including rising edge and falling edge.

4.5.8 IPWM Read with Interrupt Handler

This module is used to measure the frequency and duty cycle of the switch input channel using interrupt mode.



Block Parameter:

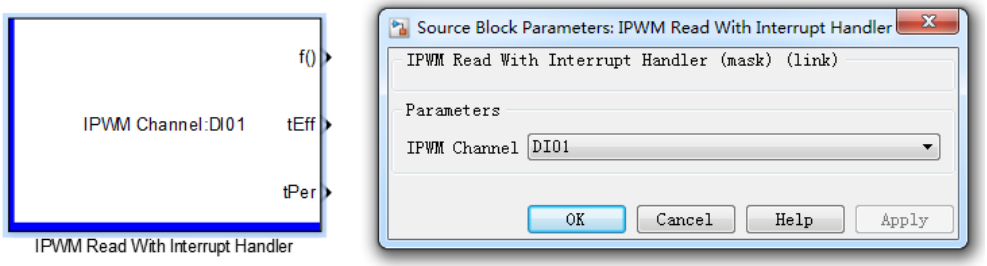
1) IPWM Channel

Block Output:

1) f(): functional call
2) tEff: Specifies the effective level time of the channel signal, in (us).
3) tPer: Specify the period of the channel signal, in (us).

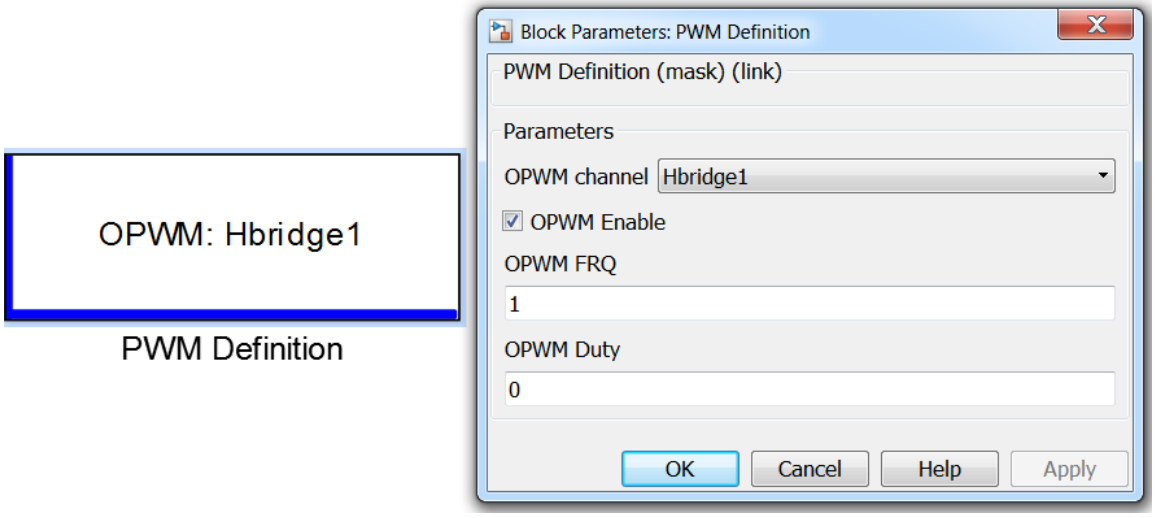
Note: When the trigger mode is rising edge trigger, the effective level time refers to

high level time; when the trigger mode is falling edge trigger, the effective level time refers to low level time.



4.5.9 PWM Definition

This module is used to initialize the OPWM channel. All H-bridge, high-side, and low-side channels can be initialized and only have OPWM function after setting.



Block Parameter:

1) OPWM channel
2) OPWM Enable
3) OPWM FRQ: The channel initialization frequency, the unit is related to the PWM IO Frequency Range Definition setting.
4) OPWM Duty: The channel initialization duty cycle, where 0-10000 corresponds

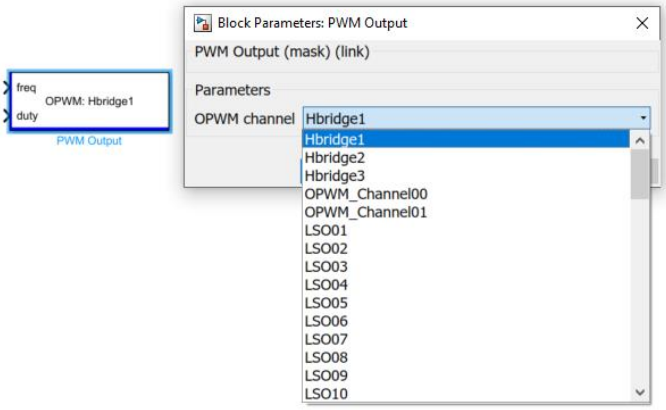
to 0-100%.

Note: The duty cycle refers to the percentage of the total on and off time when the channel is open.

4.5.10 PWM Output

Folder: EcoCoder Blocks/Digital I/O

Description:



This block configures PWM outputs.

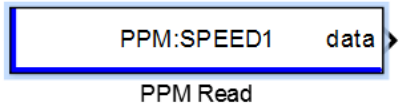
Block Parameters:

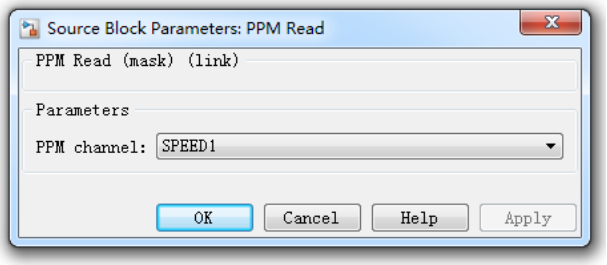
Parameter Field	Value	Comments/Description
OPWM channel	Drop-down list	Specify the PWM output channel
freq (input)	Numeric	Theoretical range is 1-2000000 Hz. Recommended range for perfect square wave output is 15 – 2000 Hz. For the input value unit, refer to *PWM IO Frequency Range Definition.*

duty (input)	Numeric	Control the duty cycle of the selected channel output, value 0-10000 corresponds to 0-100%.

4.5.11 PPM Read

This module is used to read PPM signal.





Block Parameter:

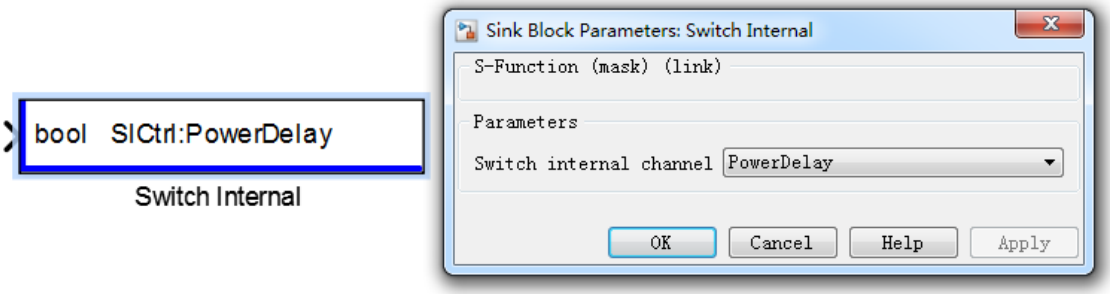
1) PPM channel

Block Output:

1) data

4.5.12 Switch Internal

This module is a switch control inside the software. If it is 1, the system is powered on; if it is 0, you need to refer to the state of other wakeup sources to determine whether the system is powered on.



Block Parameter:

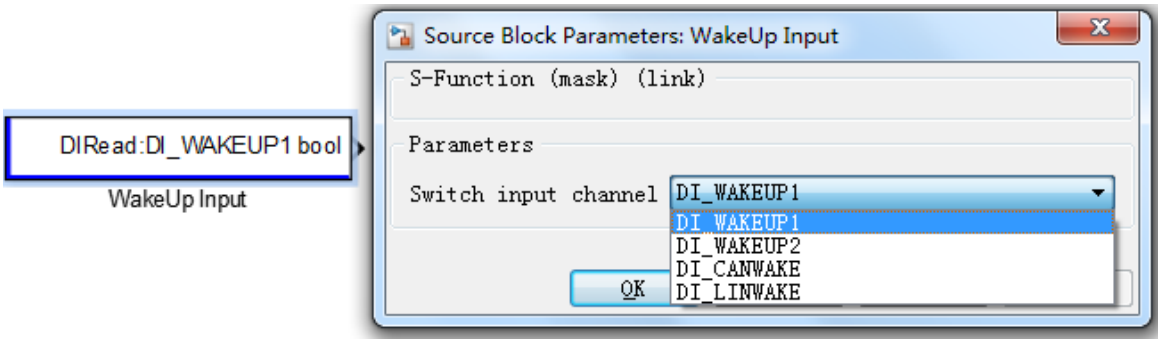
1) Switch internal channel

Note: Calling the shutdown module will clear this bit to 0.

4.5.13 WakeUp Input

Folder: EcoCoder Blocks/Digital I/O

Description:



This block can read wake-up signal status.

Block Parameters

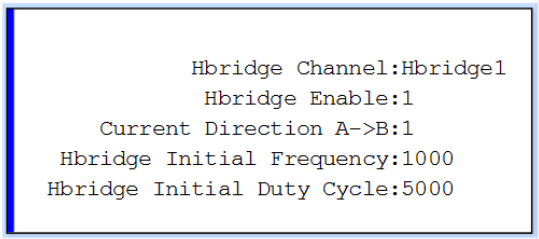
Parameter Field	Value	Comments/Description
Switch input channel	Drop-down list	Wakeup source selection

Output	Numeric (Boolean)	'1' is active

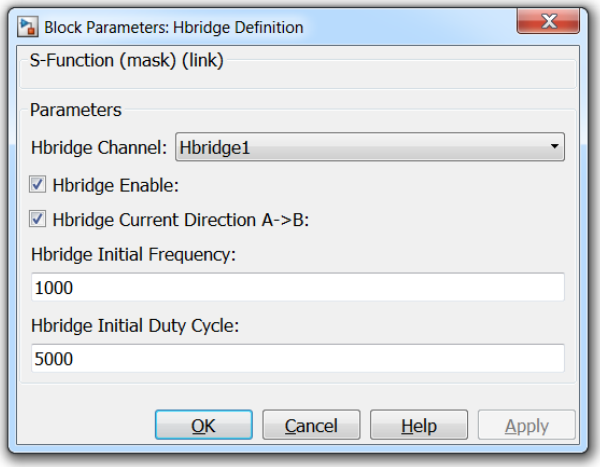
4.5.14 H-bridge Definition

Folder: EcoCoder Blocks/Digital I/O

Description:



Hbridge Definition



This block is used for setting up the VCU H-bridge(s).

Block Parameters

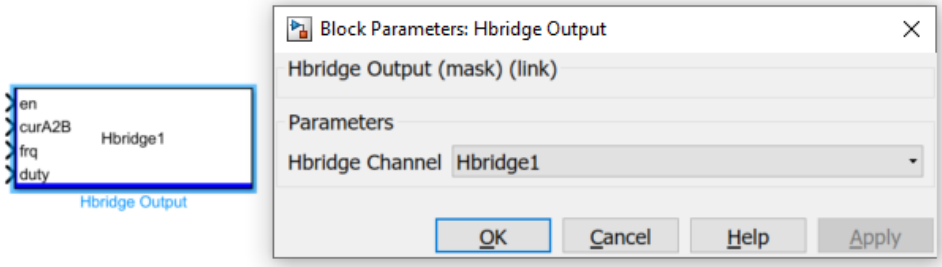
Parameter Field	Value	Comments/Description
H-bridge Channel	Drop-down list	Select H-bridge channel
H-bridge Enable	Check box	If checked: Enable H-bridge
H-bridge Current Direction A->B	Check box	If checked: The current direction is from A->B. If not checked: the current direction is B->A. (A and B are the two outputs of H-bridge, see the VCU data sheet for

		more information)
H-bridge Initial Frequency	Numeric (Hz)	The theoretical range is 1-2000000 Hz. Recommended range for perfect square wave output is 15 – 2000 Hz. Input value unit is configurable in the *PWM IO Frequency Range Definition* block.
H-bridge Initial Duty Cycle	Numeric	0-10000 corresponds to 0-100%.

4.5.15 H-bridge Output

Folder: EcoCoder Blocks/Digital I/O

Description:



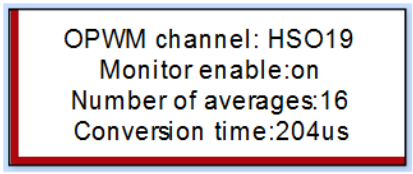
This block controls H-bridge output.

Block Parameters

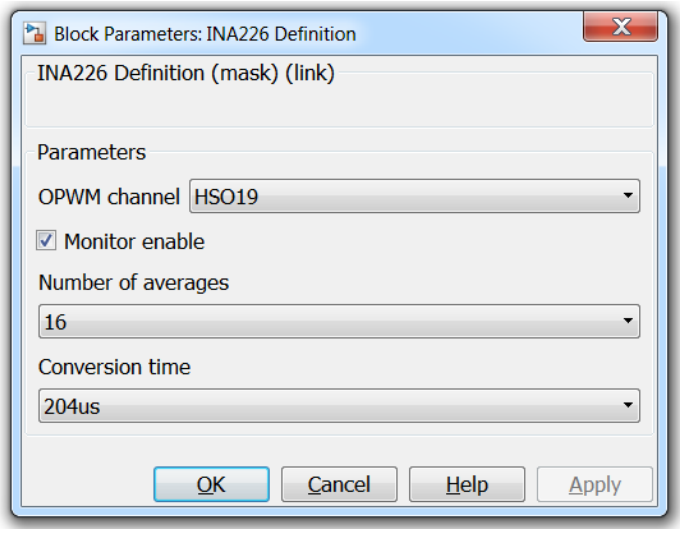
Parameter Field	Value	Comments/Description
H-bridge Channel	Drop-down list	Select H-bridge channel
en (input)	Numeric (bool)	'1' to enable H-bridge
curA2B (input)	Numeric (bool)	'1': current flows from A to B; '0': currents flow from B to A.

frq (input)	Numeric	The theoretical range is 1-2000000 Hz. Recommended range for perfect square wave output is 15 – 2000 Hz. Input value unit is configurable in the *PWM IO Frequency Range Definition* block.
duty (input)	Numeric	0-10000 corresponds to 0-100%.

4.5.16 INA226 Definition



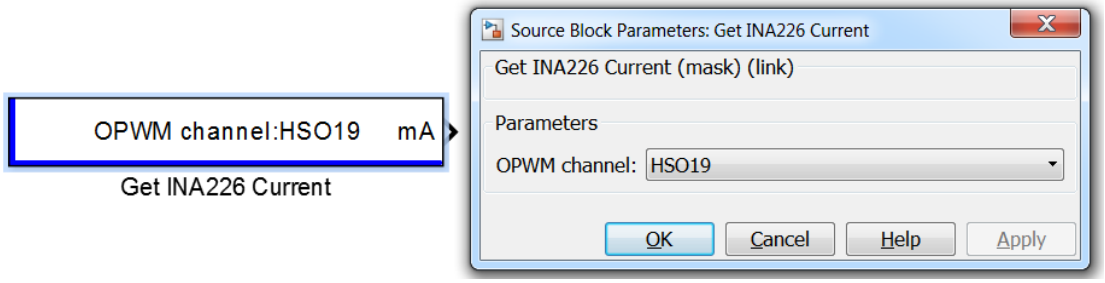
INA226 Definition



Block Parameter:

1) OPWM Channel
2) Monitor enable: Enable monitoring IN226
3) Number of averages
4) Conversion time：Bus and shunt voltage conversion time.

4.5.17 Get INA226 Current



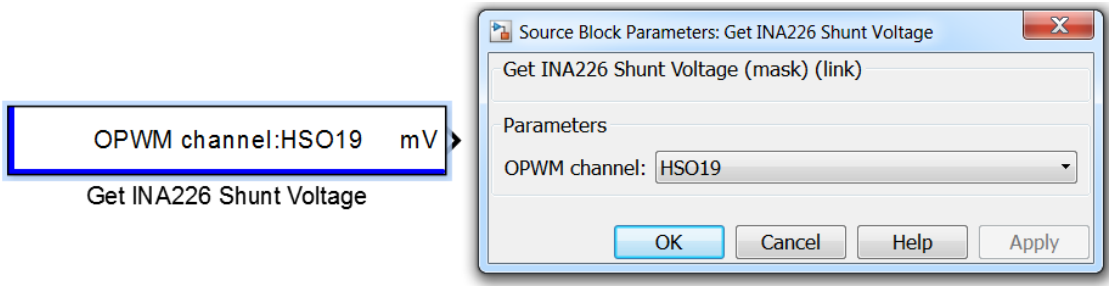
Block Parameter:

1) OPWM Channel

Block Output:

1) mA：Read the INA226 current, in mA.

4.5.18 Get INA226 Shunt Voltage



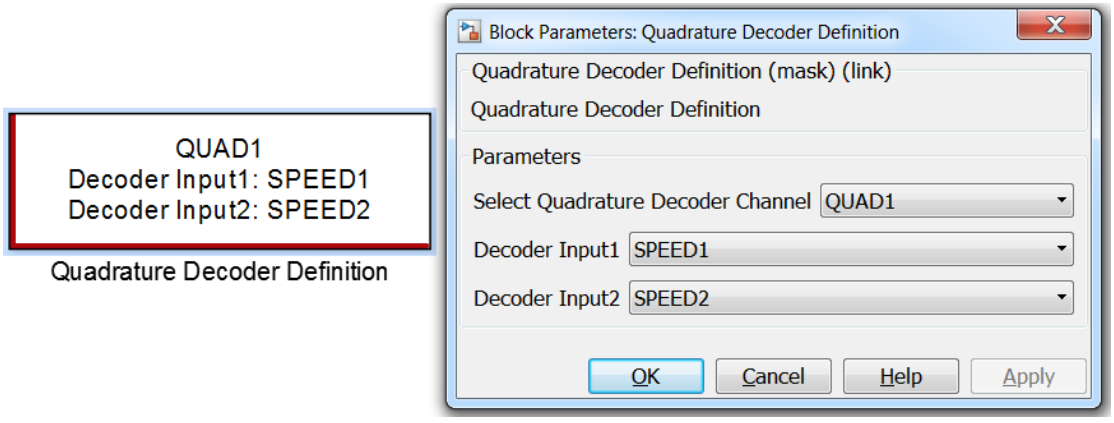
Block Parameter:

1) OPWM Channel

Block Output:

1) mV：Read the INA226 shunt voltage value，in mV.

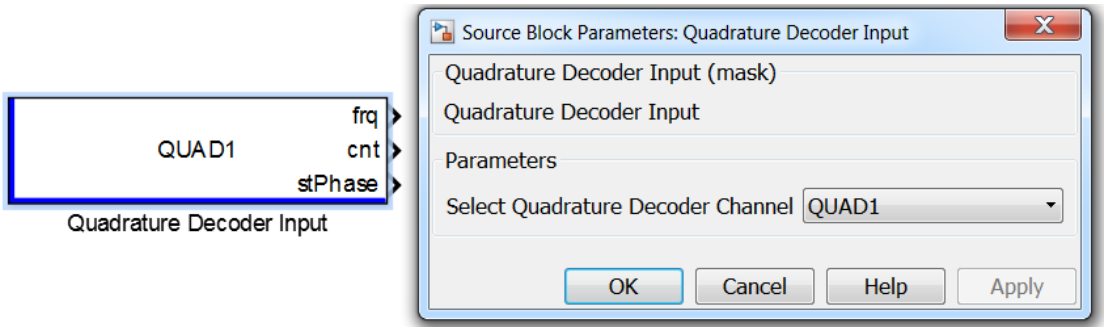
4.5.19 Quadrature Decoder Definition



Block Parameter

1) Select Quadrature Decoder Channel
2) Decoder Input1
3) Decoder Input2

4.5.20 Quadrature Decoder Input



Block Parameter

1) Select Quadrature Decoder Channel

Block Output:

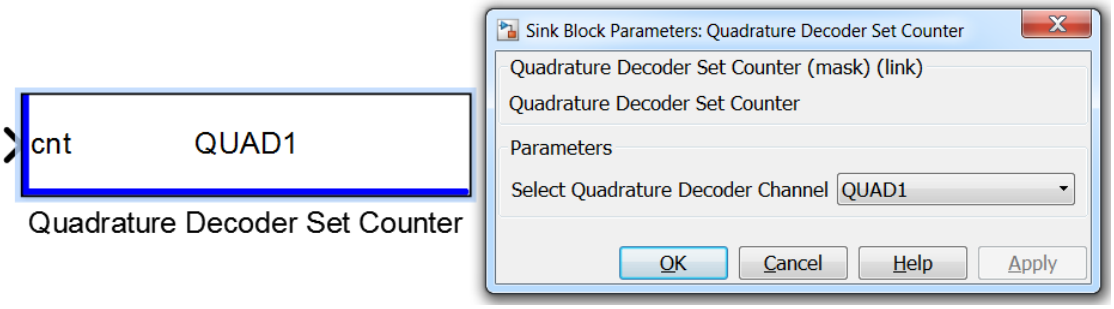
1) frq：Frequency in Hz

2) cnt：For a square wave counter, if the phase state stPhase is 1, it will increase by 1, and if it is -1, it will decrease by 1.

3) stPhase：Phase status, 1 means channel 1 leads channel 2 by 90°, -1 means channel 1 lags channel 2 by 90°

4.5.21 Quadrature Decoder Set Counter

This module can be used to set the initial value of quadrature decoding square wave count value, generally used to clear the count.



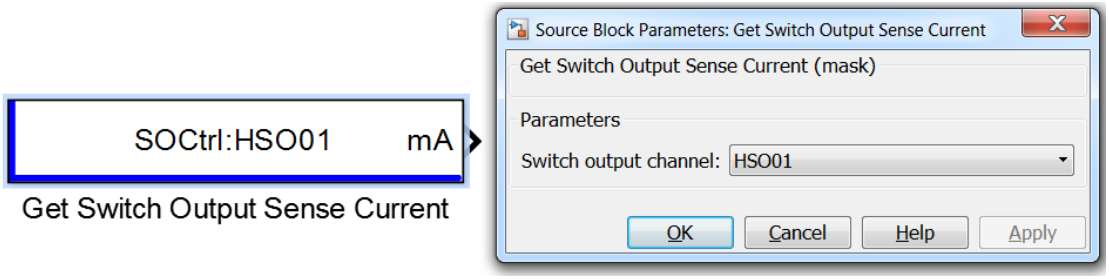
Block Parameter:

1) Select Quadrature Decoder Channel

Block Output:

1) cnt：Initial square wave count.

4.5.22 Get Switch Output Sense Current



Block Parameter

1) Switch output channel

Block Output:

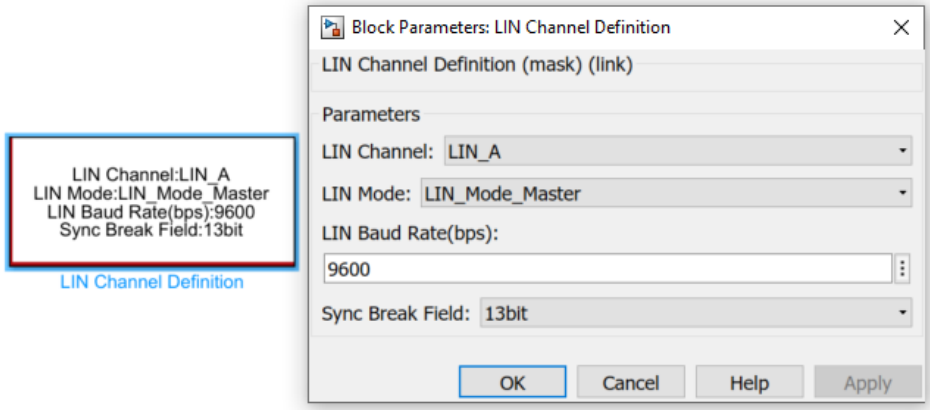
2) mA：Channel detection current value, the unit is mA.

4.6 LIN Communication

4.6.1 LIN Channel Definition

Folder: EcoCoder Blocks/LIN

Description:



This block provides configuration interface for LIN parameters.

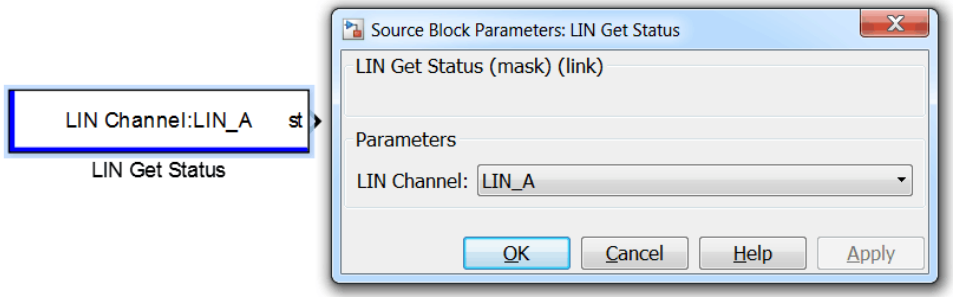
Block Parameters

Parameter Field	Value	Comments/Description
LIN Channel	Drop-down list	Please refer to datasheet to select supported LIN channel.
LIN Mode	Drop-down list	Select Lin mode. (Master or Slave)
LIN Baud Rate (bps):	Numeric	Input LIN baud rate.

4.6.2 LIN Get Status

Folder: EcoCoder Blocks/LIN

Description:



This module is used to get the status of LIN channel.

Block Parameters

Parameter Field	Value	Comments/Description
LIN Channel	Drop-down list	Please refer to datasheet to select supported LIN channel.

Block Output:

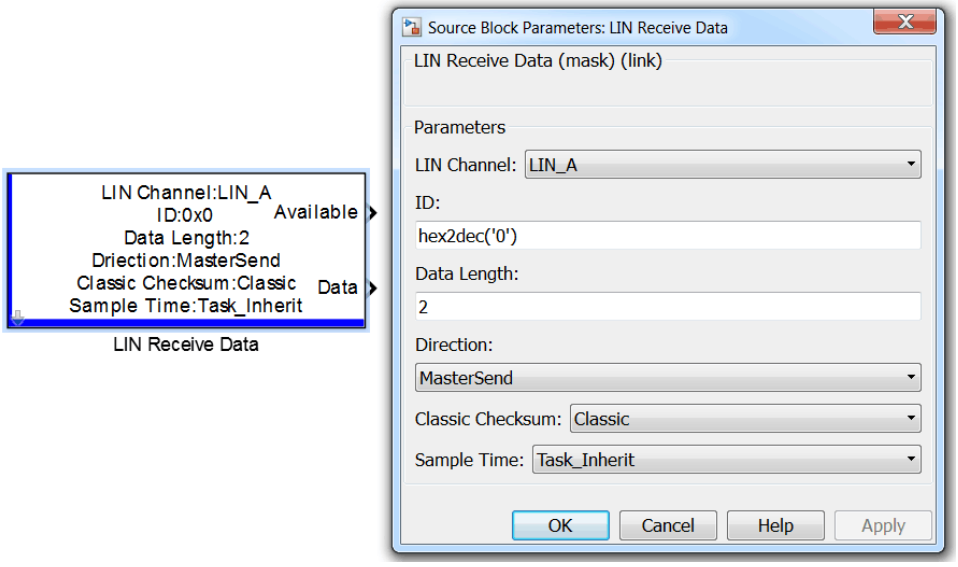
1. St: (status)

Value	Description
0	Normal
1	Error
6	Busy

4.6.3 LIN Receive Data

Folder: EcoCoder Blocks/LIN

Description:



This module is used to receive data from the LIN bus.

Block Parameters

Parameter Field	Value	Comments/Description
LIN Channel	Drop-down list	Please refer to datasheet to select supported LIN channel.
ID	Numeric	Data address to receive
Data Length	Numeric	Data length
Direction	Drop-down list	Select according to LIN mode
Classic Checksum	Drop-down list	Checksum category
Sample Time	Drop-down list	Task sample time

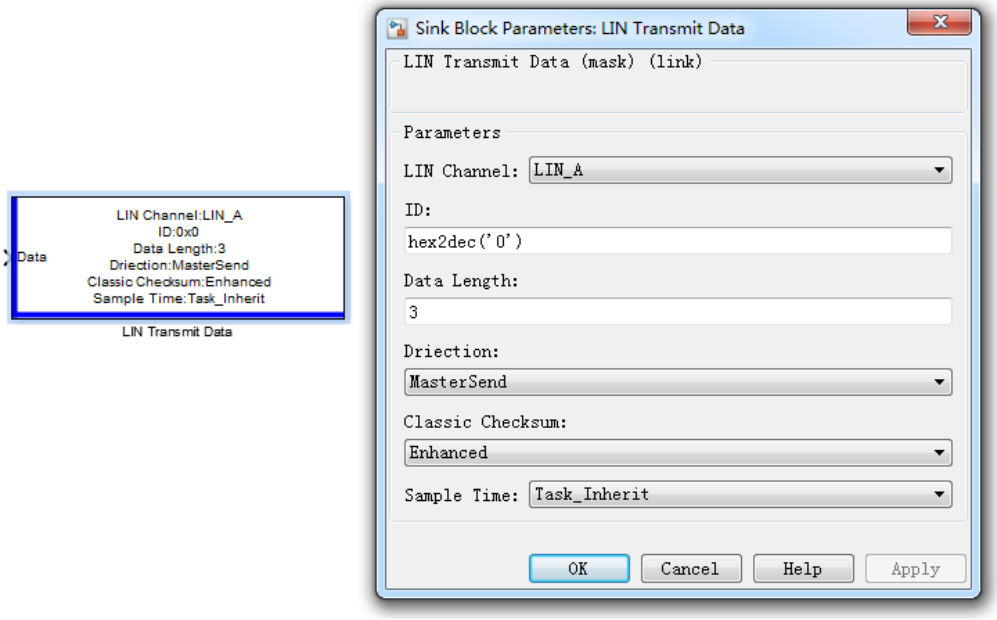
Block Outputs:

1) Available: output of 1 means data is valid, otherwise invalid.
2) Date: receiving data

4.6.4 LIN Transmit Data

Folder: EcoCoder Blocks/LIN

Description:



This module is used to send data to the LIN bus.

Block Parameters

Parameter Field	Value	Comments/Description
LIN Channel	Drop-down list	Please refer to datasheet to select supported LIN channel.
ID	Numeric	Data address to receive
Data Length	Numeric	Data length
Direction	Drop-down list	Select according to LIN

		mode
Classic Checksum	Drop-down list	Checksum category
Sample Time	Drop-down list	Task sample time

Block Input:

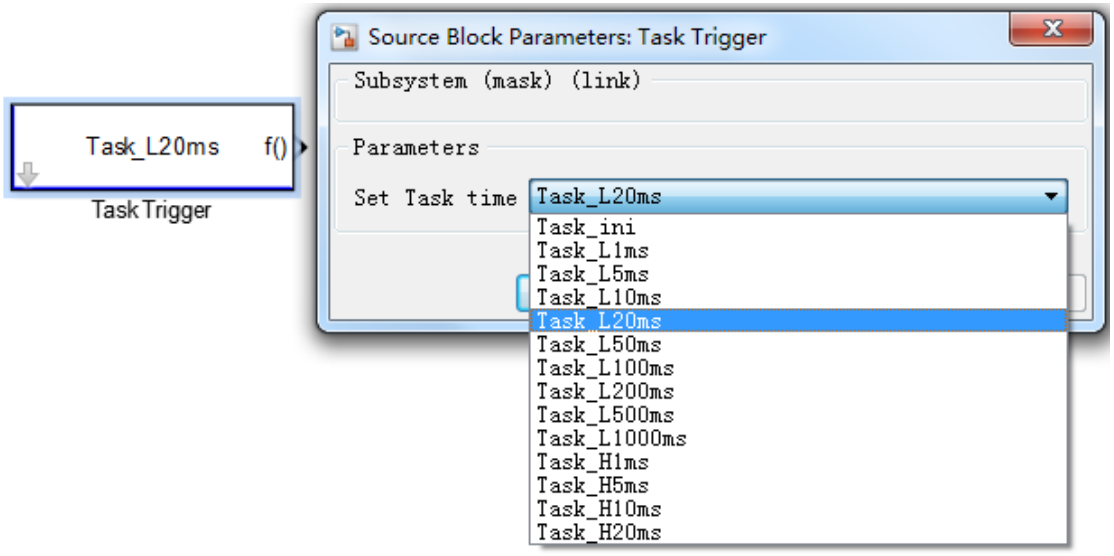
1) Date: sending data

4.7 Task Scheduler

4.7.1 Task Trigger

Folder: EcoCoder Blocks/Task Scheduler

Description:



This block is for task scheduling and prioritization.

Definition / initialization blocks need to be executed when the VCU power on, for variables initialization/parameter definitions.

All other blocks/subsystems should be triggered by this block, for task prioritization and scheduling.

Block Parameters

Parameter Field	Value	Comments/Description
Set Task Time	Drop-down list	Task type and execution period selection*

* **H** represents high priority. Tasks will be implemented by interruption. **L** represents low priority. Tasks will be implemented by software timer function call.

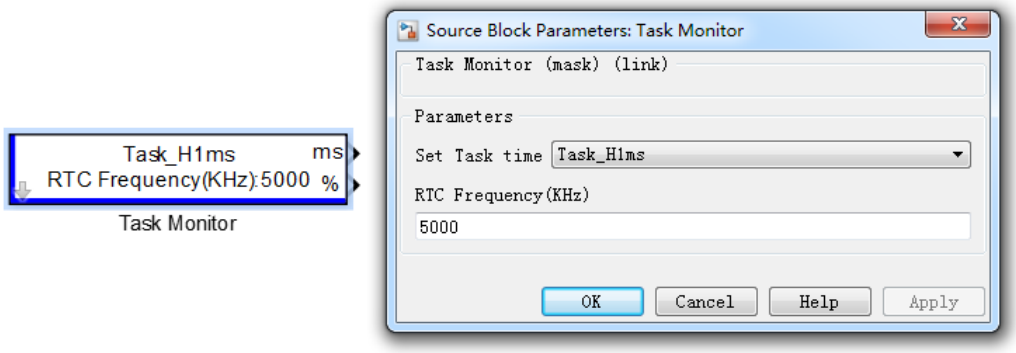
If two tasks are assigned to the same task type, then the user needs to specify priority of the two tasks to determine execution order. Please refer to the example in the following link for more information: *https://www.mathworks.com/help/simulink/examples/block-priority.html*

* For CAN bus applications, they are recommended to be set up in 10ms tasks.

4.7.2 Task Monitor

Folder: EcoCoder Blocks/Task Scheduler

Description:



This module is used to monitor task scheduling time.

Block Parameters

Parameter Field	Value	Comments/Description

Set Task Time	Drop-down list	Task type selection
RTC Frequency (KHz)	Numeric	NXP5606: 150,000 NXP5744: 5,000 TC275: 100,000 TC297: 200,000 TC397: 100,000

Block Output:

1. ms: The actual execution cycle of the task.

2. %: Load factor of chosen task type.

4.8 Non-Volatile Memory Blocks

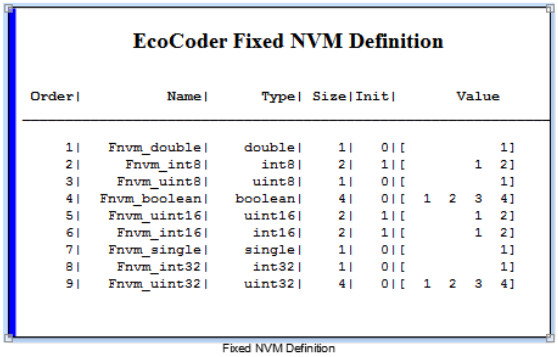
There are two types of Non-Volatile Memory. The first type is NVM, and the second type is Fixed NVM. Data stored in NVM will not be lost between power cycles. Data stored in Fixed NVM will not be lost after the VCU is programmed.

For more information and demo, please refer to Appendix A.

4.8.1 Fixed NVM Definition

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

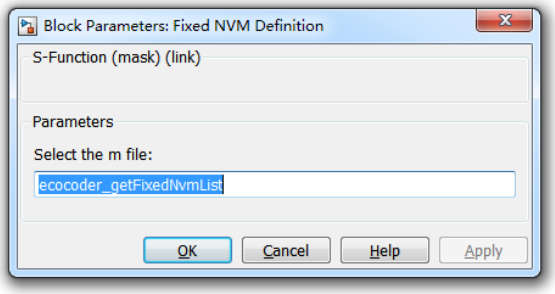
Description:



EcoCoder Fixed NVM Definition

Order	Name	Type	Size	Init	Value
1	Fnvm_double	double	1	0	[1]
2	Fnvm_int8	int8	2	1	[1 2]
3	Fnvm_uint8	uint8	1	0	[1]
4	Fnvm_boolean	boolean	4	0	[1 2 3 4]
5	Fnvm_uint16	uint16	2	1	[1 2]
6	Fnvm_int16	int16	2	1	[1 2]
7	Fnvm_single	single	1	0	[1]
8	Fnvm_int32	int32	1	0	[1]
9	Fnvm_uint32	uint32	4	0	[1 2 3 4]

Fixed NVM Definition



This module is used to define and initialize fixed NVM variables.

This Fixed NVM Definition block will only be executed once, during the first power-up process of VCU application software, every time after the .mot file being flashed into VCU.

Block Parameters

Parameter Field	Value	Comments/Description
Select the m file	.m file	Select the .m file defining NVM variables.

*The m file can be defined as the picture below:

```
function NVMList=ecocoder_getFixedNvmList()
NVMList={...
    struct('name', {'Fnvm_double'},'type', {'double'},'size', 1,'init', 0,'value',1), ...
    struct('name', {'Fnvm_int8'},'type', {'int8'},'size', 2,'init', 1,'value',[1 2]), ...
    struct('name', {'Fnvm_uint8'},'type', {'uint8'},'size', 1,'init', 0,'value',1), ...
    struct('name', {'Fnvm_boolean'},'type', {'boolean'},'size', 4,'init', 0,'value',[1 2 3 4]), ...
    struct('name', {'Fnvm_uint16'},'type', {'uint16'},'size', 2,'init', 1,'value',[1 2]), ...
    struct('name', {'Fnvm_int16'},'type', {'int16'},'size', 2,'init', 1,'value',[1 2]), ...
    struct('name', {'Fnvm_single'},'type', {'single'},'size', 1,'init', 0,'value',1), ...
    struct('name', {'Fnvm_int32'},'type', {'int32'},'size', 1,'init', 0,'value',1), ...
    struct('name', {'Fnvm_uint32'},'type', {'uint32'},'size', 4,'init', 0,'value',[1 2 3 4]), ...
};
end
```

Parameters in the .m file:

1) Order: order of the Fixed NVM variables
2) Name: variable name
3) Type: data type of the variable
4) Size: length of the array variable
5) Init: The 'init' in the .m file is the flag for NVM variable initialization.

   init = 1: The NVM variable value will be set to initial value defined in parameter 'Value' after .mot file being flashed into VCU.

   init = 0: The NVM variable value will be loaded from the VCU flash (value stored last time) after .mot file being flashed into VCU.
6) Value: initial value of the variable

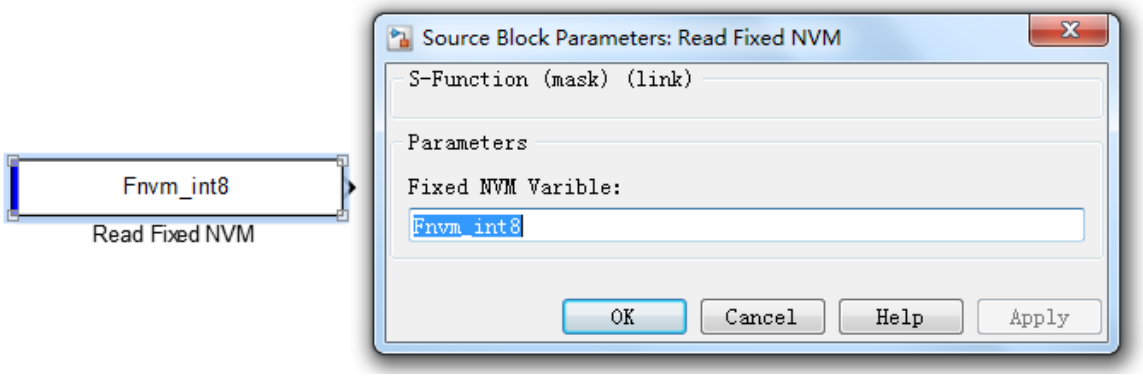
If the VCU that you are operating is a brand new VCU and will be flashed for the very first time, no matter what the 'init' value is, the NVM variables will be initialized from .mot file.

Note: The .m file needs to be added under MATLAB path.

4.8.2 Read Fixed NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This module is used for reading fixed NVM variables.

Block Parameters

Parameter Field	Value	Comments/Description

Fixed NVM Variable	Variable name	Specify the variable name to be read

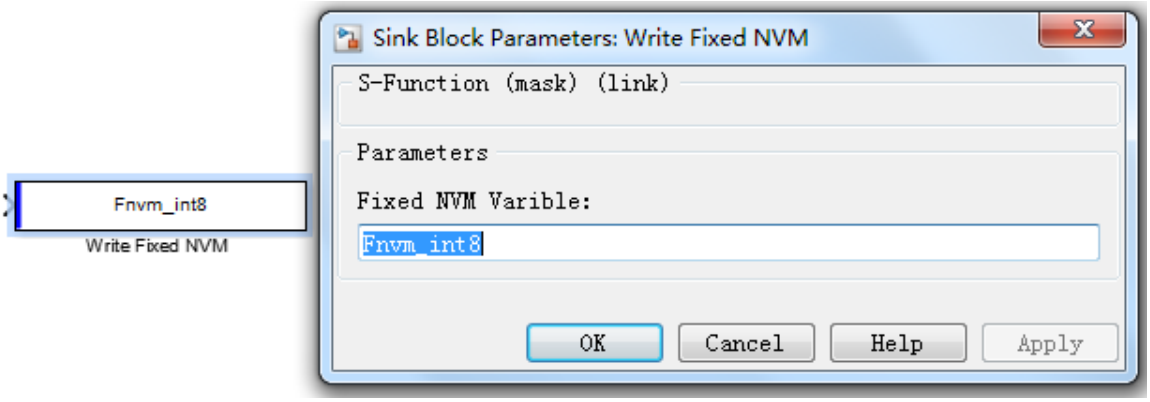
Block Output:

The value of corresponding variables.

4.8.3 Write Fixed NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This module is used for writing fixed NVM variables.

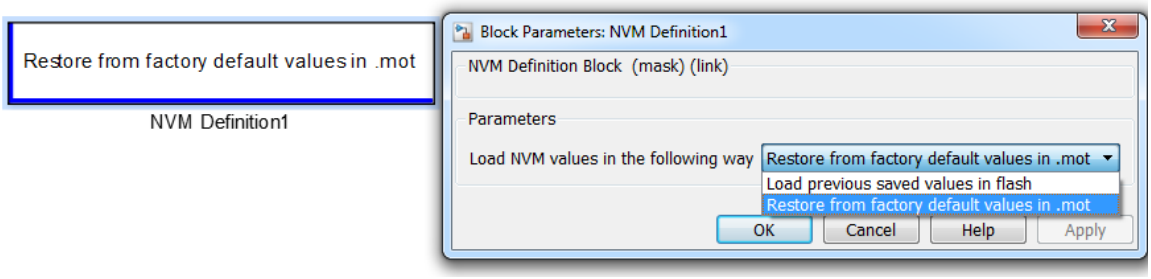
Block Parameters:

Parameter Field	Value	Comments/Description
Fixed NVM Variable	Variable name	Specify the variable to be written.

4.8.4 NVM Definition

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This block is used to initialize NVM variables and specify the NVM variable initialization method after every time the VCU being programmed by .mot file.

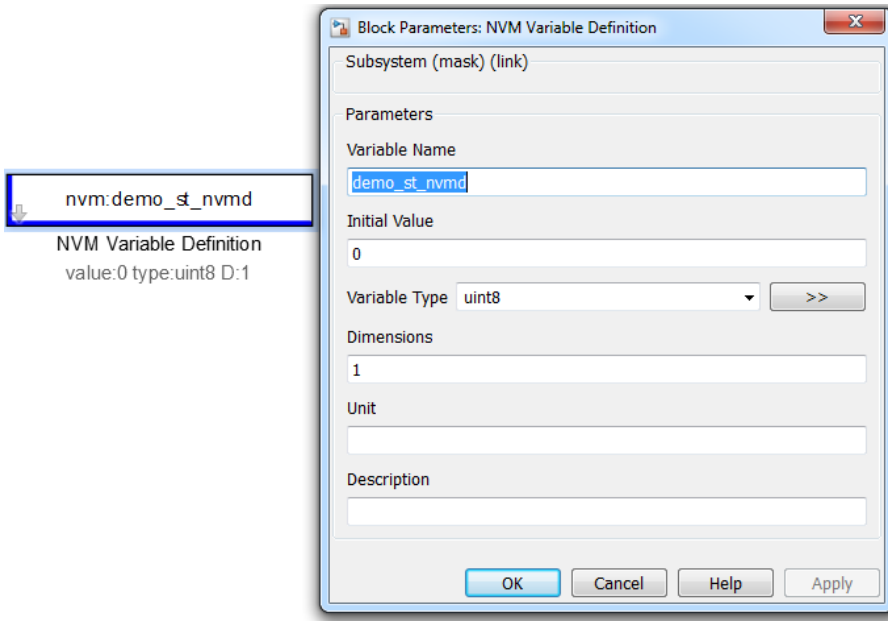
Block Parameters:

Parameter Field	Value	Comments/Description
Load NVM value in the following way	Drop-down list	**Load previous saved values in flash:** The corresponding NVM variable value would be initialized from the NVM memory area, instead of .mot file. **Restore from factory default values in .mot:** The corresponding NVM variable value would be initialized from .mot file.

4.8.5 NVM Variable Definition

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This block is used to define regular NVM variables.

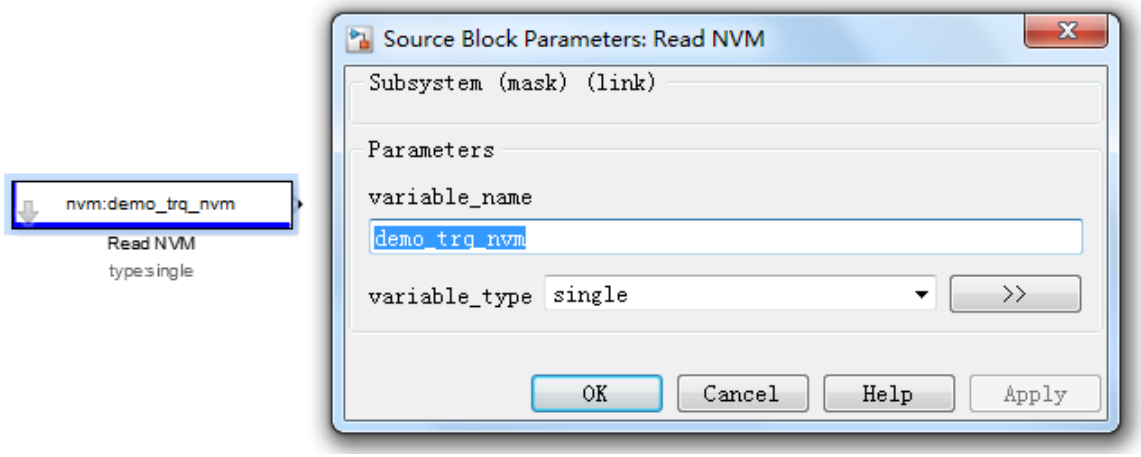
Block Parameters:

Parameter Field	Value	Comments/Description
Variable Name	Alpha-numeric text	Variable name
Initial Value	Numeric	Initial value of the to-be defined variable
Variable Type	Drop-down list	Select variable data type
Dimension	Numeric	Variable dimension
Unit	Alpha-numeric text	User-defined variable unit
Description	Alpha-numeric text	User-defined variable description

4.8.6 Read NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This module is used for reading regular NVM variables.

Block Parameters:

Parameter Field	Value	Comments/Description
Variable_name	Alpha-numeric text	Specify variable name
Variable_type	Drop-down list	Variable data type

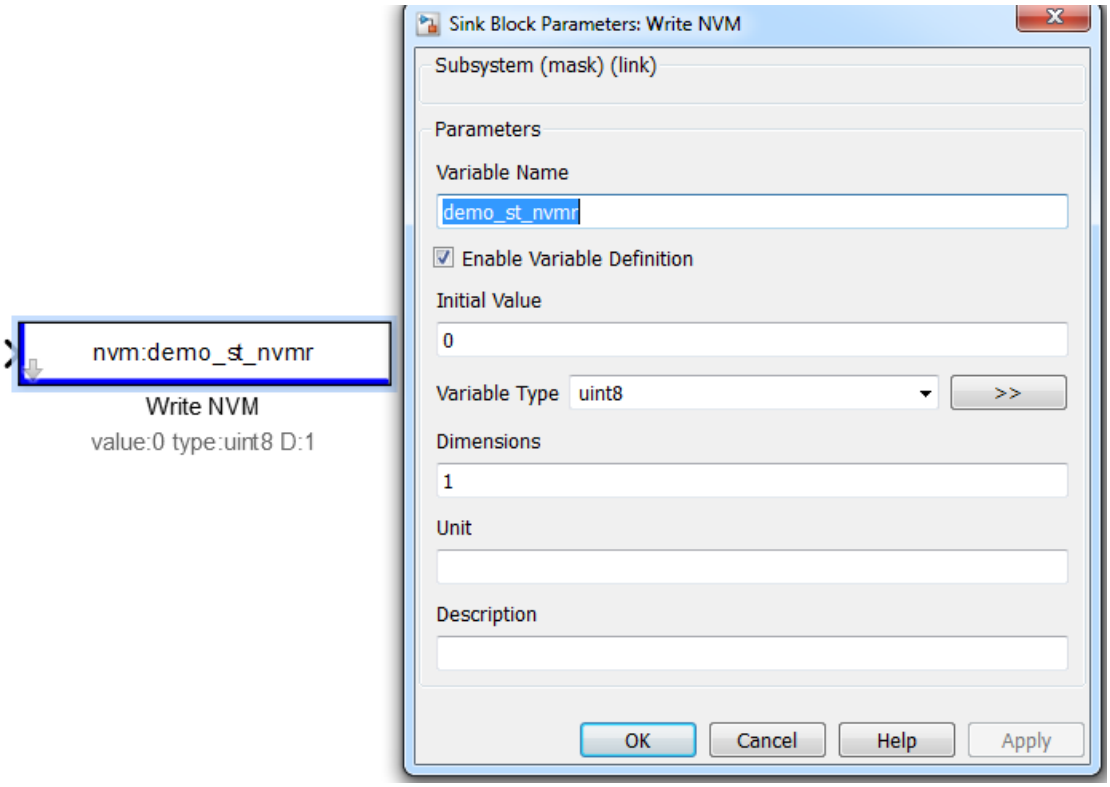
Block Output:

The NVM variable value

4.8.7 Write NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



This module is used for writing regular NVM variables into RAM. To save changed variables into VCU flash between power cycles, users need to use another block 'Store All NVM Data'.

Block Parameters:

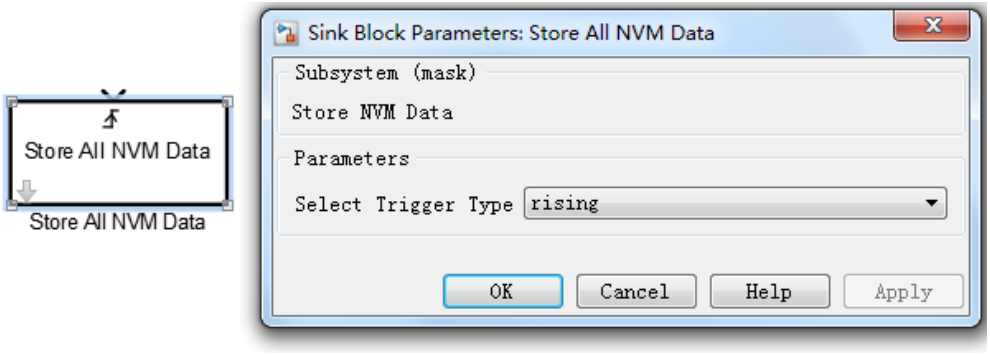
Parameter Field	Value	Comments/Description
Variable Name	Alpha-numeric text	NVM variable name
Enable Variable Definition	Check box	If checked: Define and write NVM variable If not checked: Only write NVM
Initial Value	Numeric	NVM variable initial value (for NVM variable definition)
Variable Type	Drop-down list	NVM variable type
Dimension	Numeric	Dimension of NVM variable

Unit	Alpha-numeric text	User-defined variable unit
Description	Alpha-numeric text	Memo

4.8.8 Store All NVM Data

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



When this module is triggered, all NVM variable data will be written from RAM to flash, so that the NVM data will be stored in the VCU.

It is recommended to call this block before VCU power-off. And please do not call this block too frequently. For example, if a 5ms task is assigned to this block, flash would quickly burn out because flash memory blocks have life span, and frequent programming/erasing of memory block will cause program/erase cycles running out.

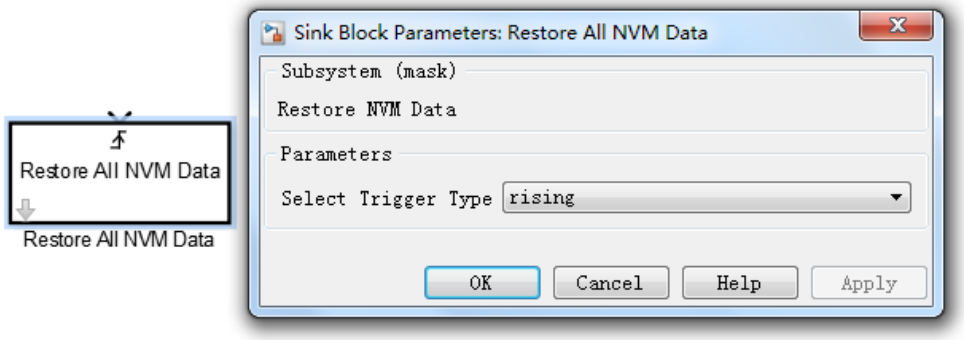
Block Parameters:

Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Select trigger type

4.8.9 Restore All NVM Data

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:



The module reads NVM data from ROM (flash) back to RAM.

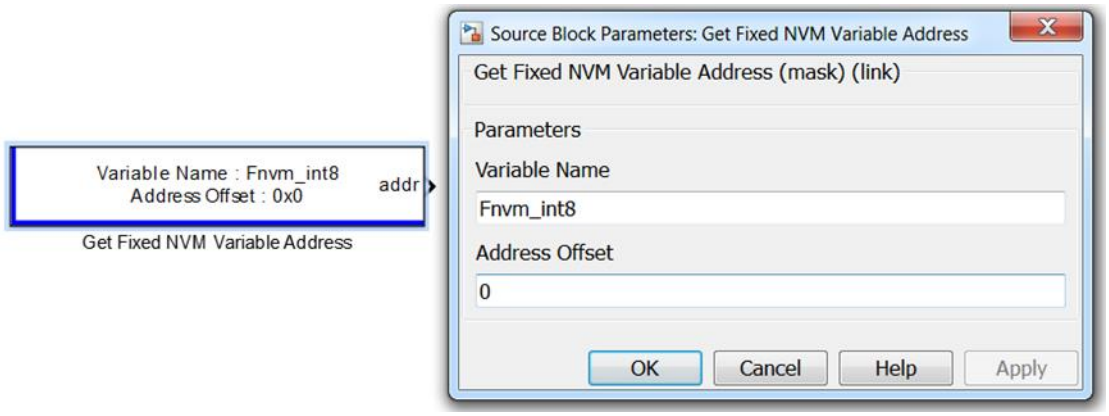
It is recommended to call this block when VCU powers on. This block can be triggered by 'Task_ini'.

Block Parameter:

Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Select trigger type

4.8.10 Get Fixed NVM Variable Address

Use this block to read the address mapped in RAM section for NVM. Use the offset parameter to read specific address in the flash. Do not use [0] for arrays for the output.



Parameter:

1) Variable Name: the name of the variable.
2) Address Offset: the offset of the address

Output:

1) addr: the address of the variable

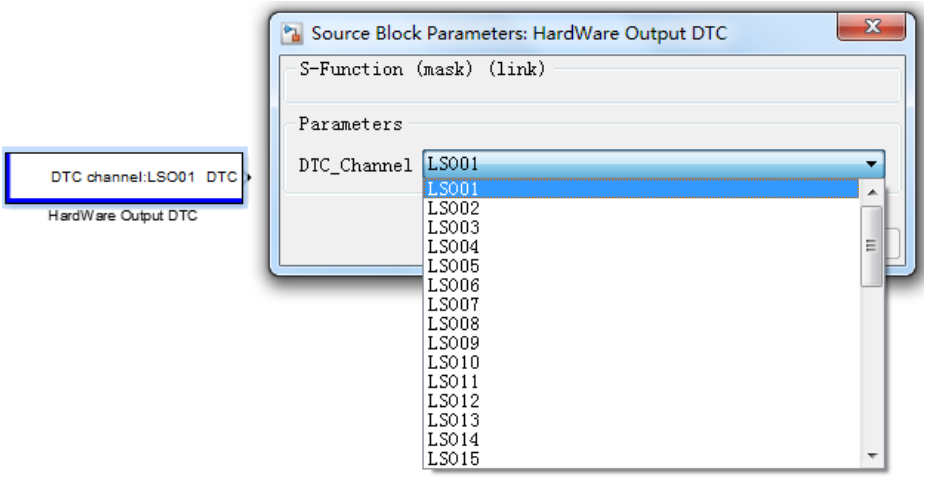
4.9 Diagnostic Blocks

Diagnostic blocks are designed to realize VCU diagnostic functions.

4.9.1 Hardware Output DTC

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:



This block can realize the hardware diagnosis of supported LSO, HSO and H-bridge.

Please refer to VCU datasheet for the channels that support diagnostic functions.

Block Parameters:

Parameter Field	Value	Comments/Description
DTC_Channel	Drop-down list	Select hardware channel

Block Output:

DTC： The diagnostic trouble code of the specified channel. An error code is 1 byte of data, each bit represents different error type. A bit is set to 1 when certain error is occurred.

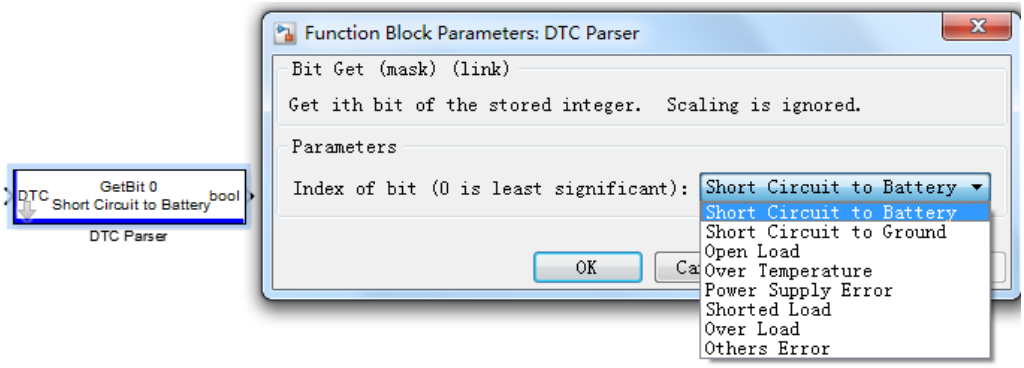
Bit Position	Error Type
Bit 0	Short to power supply
Bit 1	Short to ground
Bit 2	Open Circuit
Bit 3	Overheating

Bit 4	Power supply error
Bit 5	Short circuit at the load
Bit 6	Overload
Bit 7	Others

4.9.2 DTC Parser

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:



This block can help parse out specific fault of DTC.

Block Parameters:

Parameter Field	Value	Comments/Description
Index of bit	Drop-down list	Select the fault to be analyzed

Block Input:

DTC: The diagnostic trouble code.

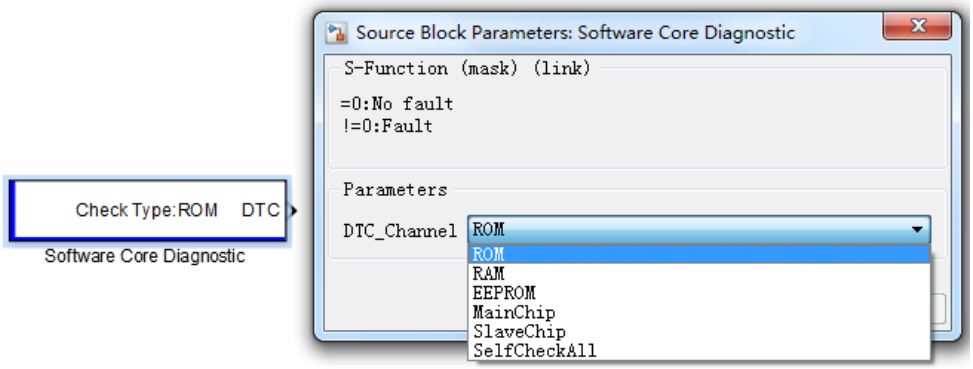
Block Output:

bool: If the output value is one, the specific fault selected in the Block Parameter happened; If output value is 0, the fault did not happen.

4.9.3 Software Core Diagnostic

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:



This block provides memory/chip fault check.

Block Parameters:

Parameter Field	Value	Comments/Description
DTC_Channel	Drop-down list	Select the memory or chip to be diagnosed.

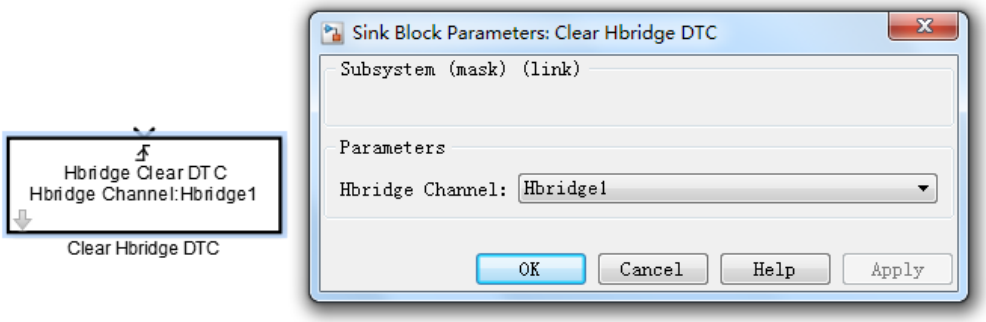
Output:

1) DTC: Output 1 represents error detected, output 0 represents error not detected

4.9.4 Clear H-bridge DTC

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:



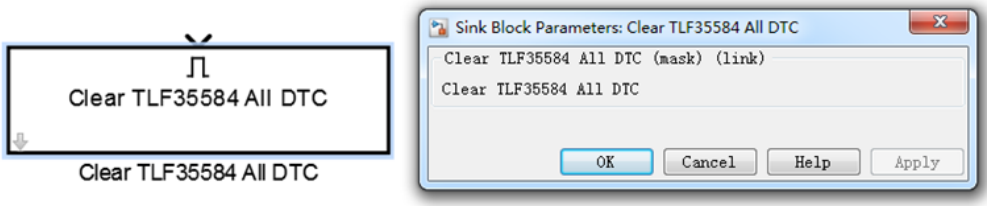
This block can clear the H-bridge channel faults, the trigger type to trigger this block is rising edge.

Block Parameters:

Parameter Field	Value	Comments/Description
H-bridge Channel	Drop-down list	Select the channel of H-bridge

4.9.5 Clear TLF35584 All DTC

This block is used for clearing all DTC code on TLF35584.



Input:

1) Triger signal: set to 1 to clear all DTC

4.9.6 Get TLF35584 Summary DTC

This block is used for reading the DTC summary from TLF35584.

Output:

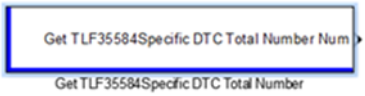
1) DTC: DTC summary from TLF35584.

Note:

1) For more information about specific error codes, please refer to TLF35584 manual.

4.9.7 Get TLF35584Specific DTC Total Number

This block is used to read the quantity of the DTC code from TLF35584.



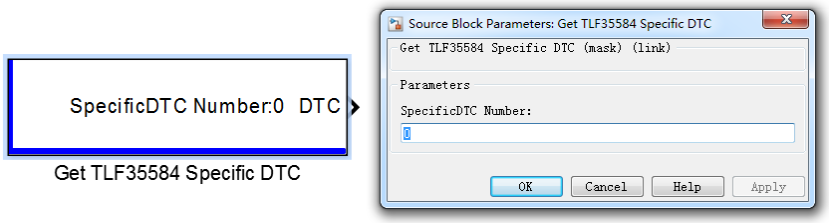
Output:

1) Num: the number of error code.

Note:

1) For more information about specific error codes, please refer to TLF35584 manual.

4.9.8 Get TLF35584 Specific DTC



Parameter:

1) SpecificDTC Number: Read error code from the specific index. To read multiple error codes, used multiple blocks to specify each error code index.

Note:

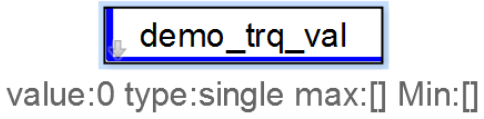
1) Since the number of error code produced by TLF35584 at any given time is unknown. It is recommended to initialize the array for DTC code based on the number of DTC codes to be read.
2) The output from this block is uint32 type.
3) For more information about specific error codes, please refer to TLF35584 manual.

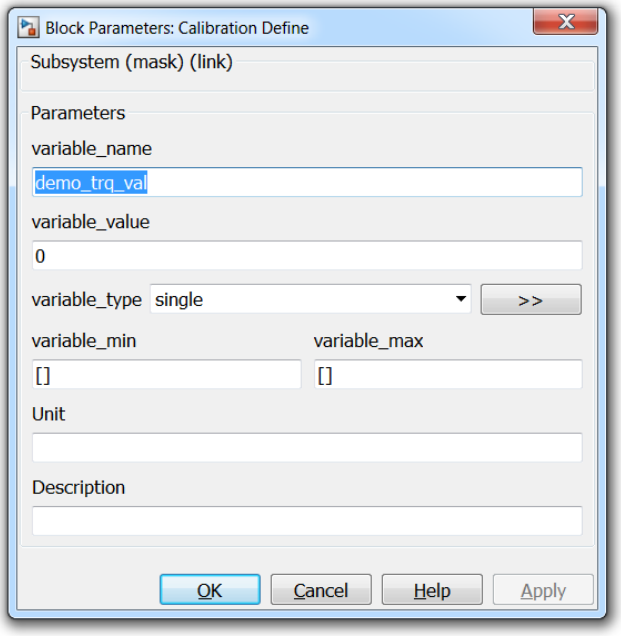
4.10 Calibration & Measurement

4.10.1 Calibration Definition

Folder: EcoCoder Blocks/Calibration & Measurement

Description:





This block can help define and initialize calibration variable.

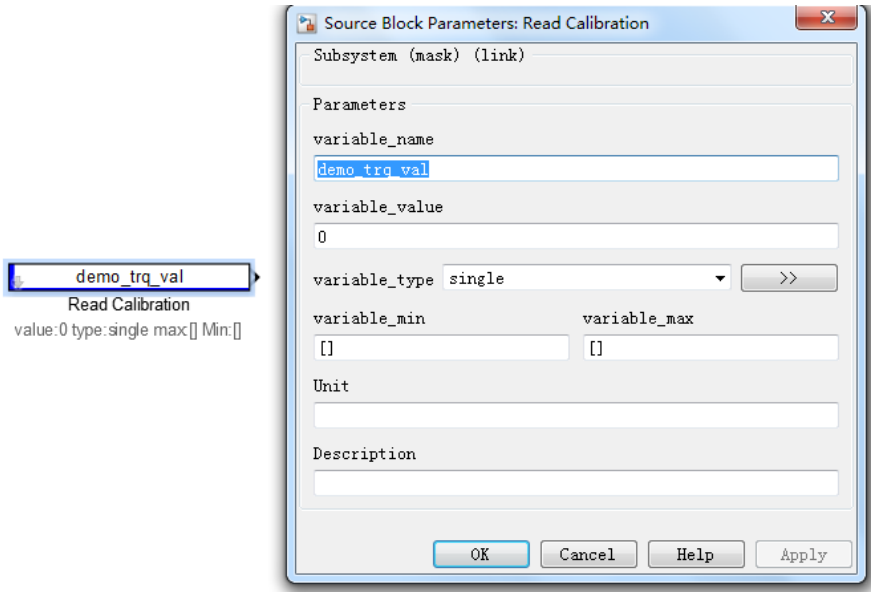
Block Parameters：

Parameter Field	Value	Comments/Description
variable_name	Alpha-numeric text	Calibration variable name
variable_value	Numeric	Calibration variable initial value
variable_type	Drop-down list	Calibration variable data type
variable_min	Numeric	Calibration variable lower limit
variable_max	Numeric	Calibration variable upper limit
Unit	Alpha-numeric text	User-defined calibration variable unit
Description	Alpha-numeric text	Memo

4.10.2 Read Calibration

Folder: EcoCoder Blocks/Calibration & Measurement

Description:



This block defines and reads calibration variables.

Block Parameters:

Parameter Field	Value	Comments/Description
variable_name	Alpha-numeric text	Calibration variable name

variable_value	Numeric	Calibration variable initial value
variable_type	Drop-down list	Calibration variable data type
variable_min	Numeric	Calibration variable lower limit
variable_max	Numeric	Calibration variable upper limit
Unit	Alpha-numeric text	User-defined calibration variable unit
Description	Alpha-numeric text	Memo

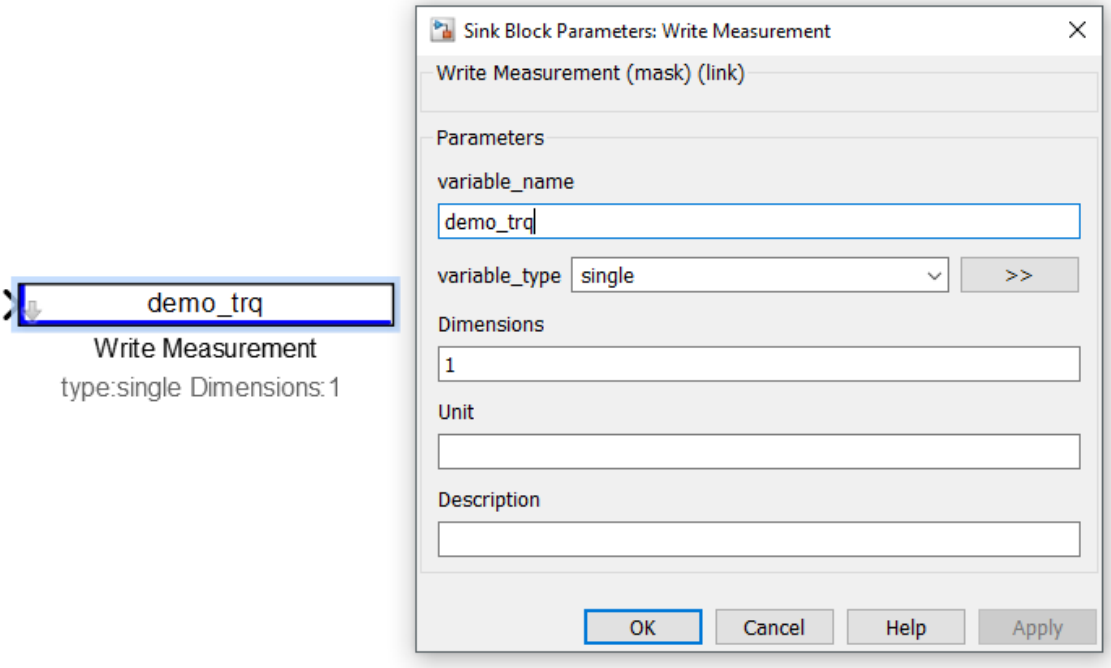
Block Output:

Calibration variable value.

4.10.3 Write Measurement

Folder: EcoCoder Blocks/Calibration & Measurement

Description



This block can help define measurement variables.

Block Parameters：

Parameter Field	Value	Comments/Description
variable_name	Alpha-numeric text	Measurement variable name
variable_type	Drop-down list	Variable data type
Dimensions	Numeric	The dimension of measurement variable
Unit	Alpha-numeric text	User-defined measurement variable unit
Description	Alpha-numeric text	Memo

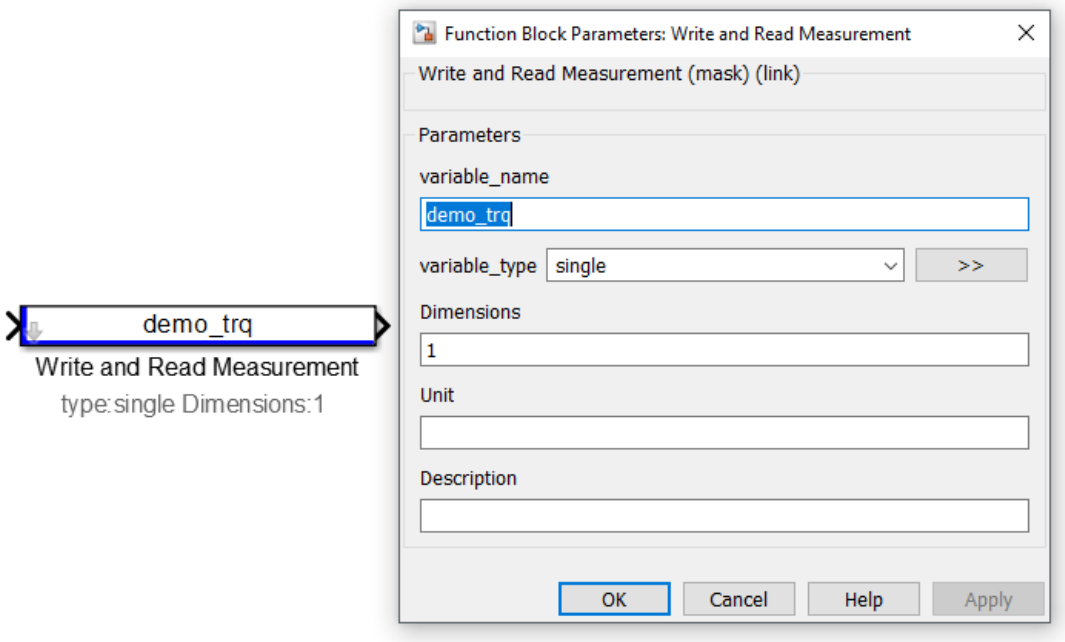
Block Input:

To-be-measured variable.

4.10.4 Write and Read Measurement

Folder: EcoCoder Blocks/Calibration & Measurement

Description:



This block is an inline block, it helps read measurement variables.

Parameter Field	Value	Comments/Description
variable_name	Alpha-numeric text	Measurement variable name
variable_type	Drop-down list	Variable data type
Dimensions	Numeric	The dimension of measurement variable
Unit	Alpha-numeric text	User-defined measurement variable unit
Description	Alpha-numeric text	Memo

Block Input:

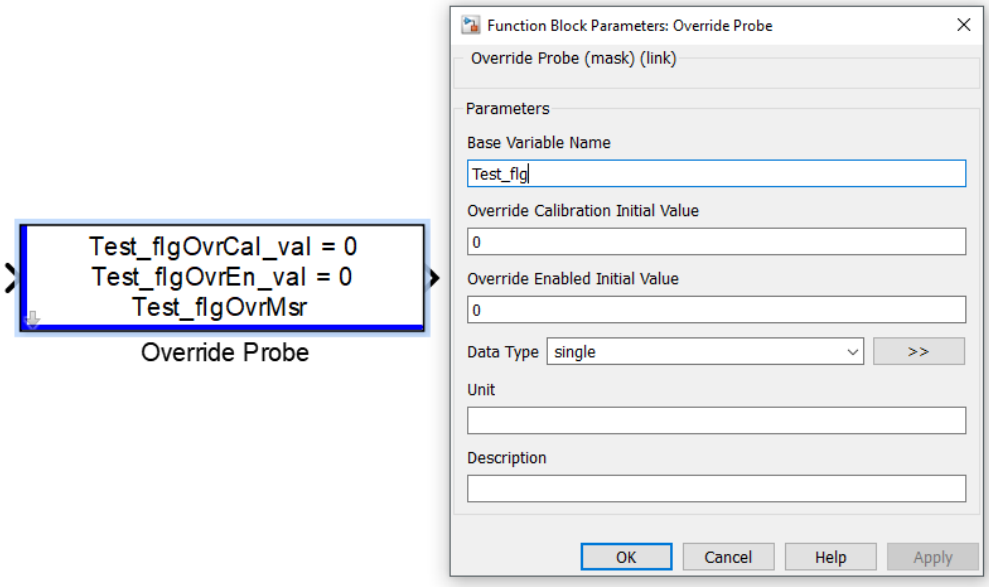
To-be-measured variable.

Block Output:

Same as input. (This block is used for variable measurements, it does not change variable values).

4.10.5 Override Probe

This block is used for overriding signal values for calibration.



In calibration software, 'Variable_nameOvrCal_val' is calibration variable, 'Variable_nameOvrMsr' is the measurement variable, 'Variable_nameOvrEn_val' is the control signal – when control signal is '1', the calibration variable will override the original signal that passes through the block, and the block output will be the calibration variable value. When the control signal is '0', this block will not override the passing-through signal, the measurement variable will have the same value as block input and block output would be the same as the block input.

Block Parameters:

1. Base Variable Name: user-defined name of the overriding variable.
2. Override Calibration Initial Value: initial value of the calibration variable.
3. Override Enable Initial Value： initial value of control signal.
4. Data Type: data type of calibration variable.
5. Unit: user-defined measurement variable unit
6. Description: the description of the variable.

Block Input:

Variable to be overridden.

Output:

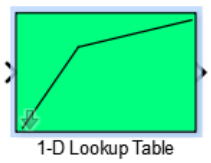
If control signal is 1, the output of the block would be the overriding calibration variable value;

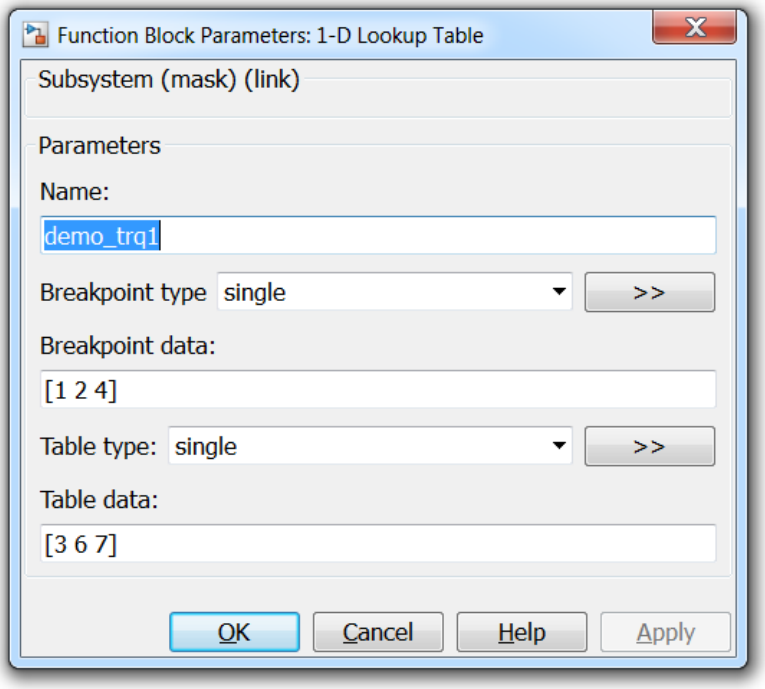
If the control signal is 0, the output of the block would be the same as the input. (No overriding)

4.10.6 1-D Lookup Table

Folder: EcoCoder Blocks/Calibration & Measurement

Description：





This block defines 1-D look-up table. 1-D look-up table supports online calibration.

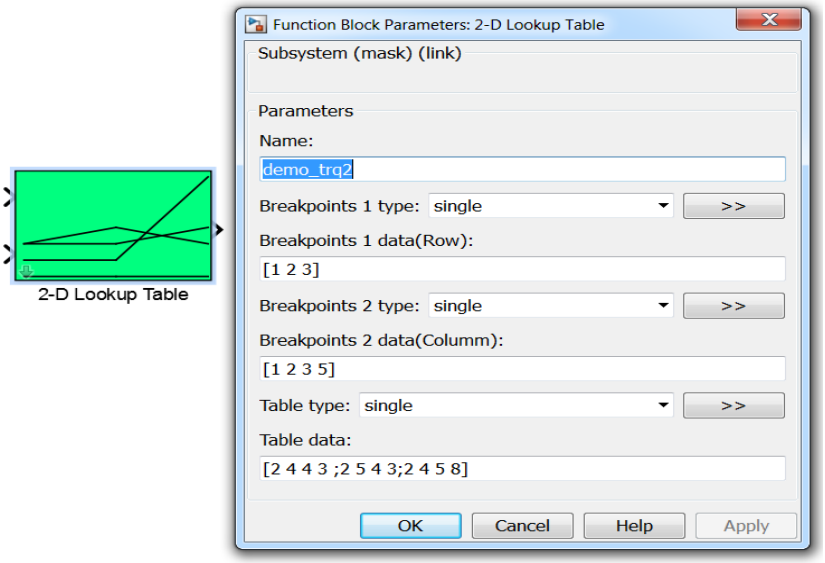
Block Parameters

Parameter Field	Value	Comments/Description
Name	Alpha-numeric text	Variable name
Breakpoint type	Drop-down list	Variable type
Breakpoint data	Numeric (Matrix)	Breakpoint data
Table type	Drop-down list	Table variable type
Table data	Numeric (Matrix)	Table data

4.10.7 2-D Lookup Table

Folder: EcoCoder Blocks/Calibration & Measurement

Description:



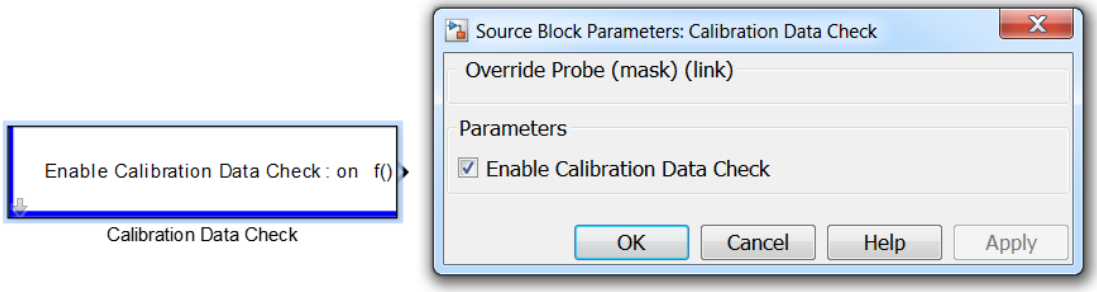
This block defines 2-D look-up table, 2-D look-up table supports online calibration.

Block Parameters:

Parameter Field	Value	Comments/Description
Name	Alpha-numeric text	2-D look-up table name
Breakpoints 1 type	Drop-down list	Breakpoints 1 variable data type
Breakpoints 1 data(Row)	Numeric (Matrix)	Breakpoints 1 variable data
Breakpoints 2 type	Drop-down list	Breakpoints 2 variable data type
Breakpoints 2 data(Column)	Numeric (Matrix)	Breakpoints 2 variable data
Table type	Drop-down list	Select table variable data type
Table data	Numeric (Matrix)	Initialize table data

4.10.8 Calibration Data Check

This module is used for checking the calibration data at the VCU power-on process. If there is any corrupted calibration data, the controller software will enter an infinite loop to avoid potential catastrophic results due to corrupted calibration data.



Block Parameters:

Enable Calibration Data Check: If checked: enable the function.

Output:

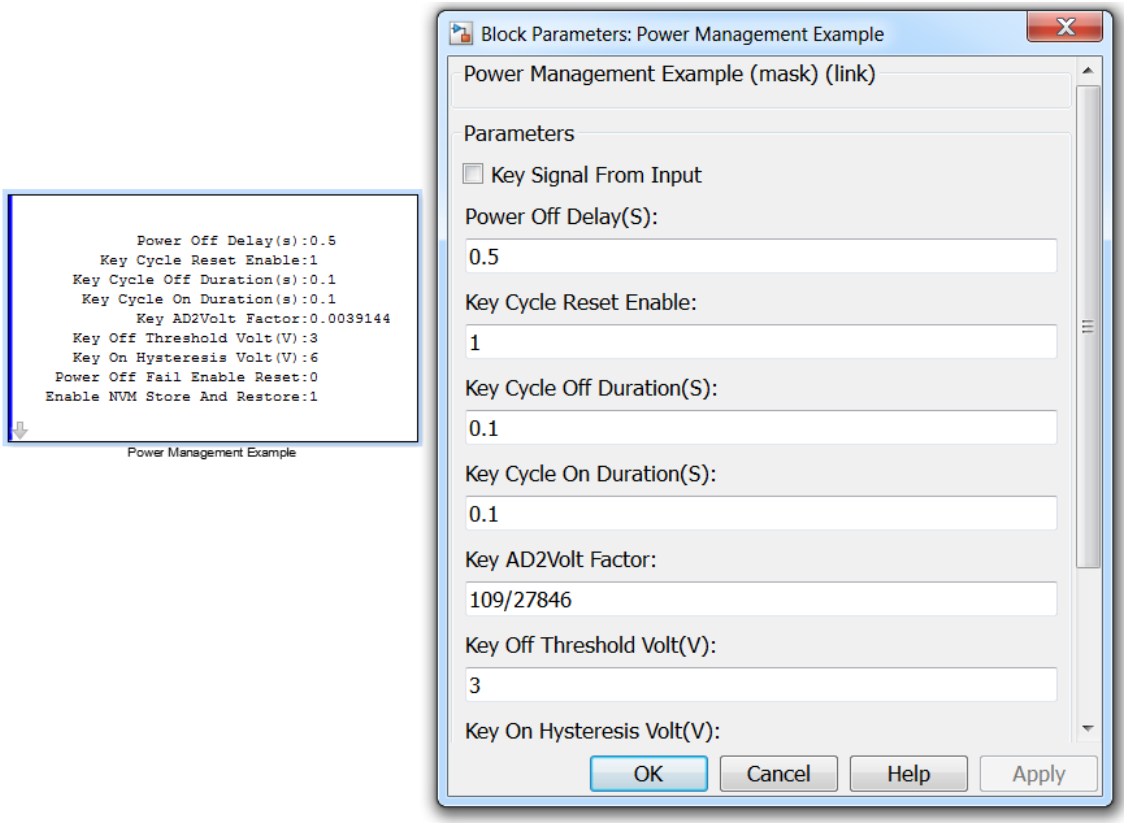
f() : Flag for checking calibration data. If there is a problem with the calibration data, the flag will be set to 1. This signal can be used as a trigger signal.

4.11 System Management Blocks

4.11.1 Power Management Example

Folder: EcoCoder Blocks/System Management Blocks

Description:



This block integrates power-off logic control and operations. It can be regarded as a reference/demo design of VCU power-off logic. Users are encouraged to understand the block first by looking down mask and then make necessary modification to the block for customized implementations.

Block Parameters

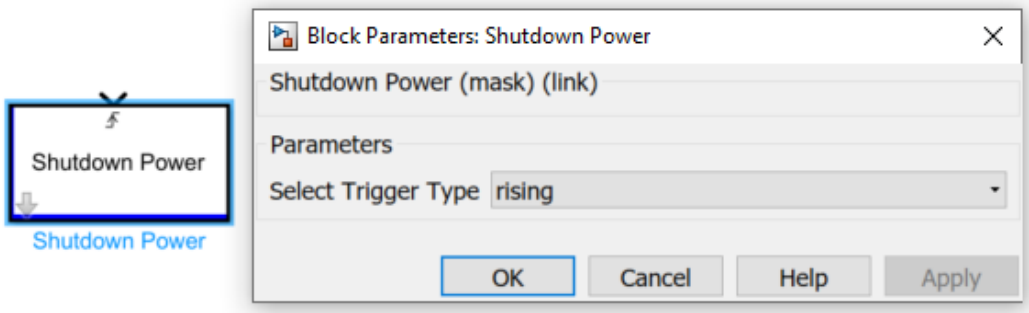
Parameter Field	Value	Comments/Description
Key Signal from Input	Check box	If checked, the key switch signal can be read from VCU input.
Power Off Delay(S)	Numeric	Power off delay time
Key Cycle Reset Enable	Numeric (Boolean)	If set to 0, VCU will not be reset if key switch turns back on before VCU power-off delay period ends. If set to 1, VCU will be reset if key switch turns back on before VCU power-off delay ends.
Key Cycle Off Duration(S)	Numeric	The duration (time threshold) after the key-off moment (KeyOn signal absent) to the time when VCU starts power-off process.
Key Cycle On Duration(S)	Numeric	VCU starts power-up process if KeyOn signal is detected for more than this time threshold.
Key AD2Volt Factor	Numeric	The factor to be multiplied that convert AD to voltage, see section 4.2.1 for details.
Key Off Threshold Volt(V)	Numeric	If the input KeyOn voltage is less than this value, KeyOn signal is interpreted as '0'.
Key On Hysteresis Volt(V)	Numeric	If the KeyOn input voltage is larger

		than the sum of 'Key Off Threshold Volt' and this hysteresis value, KeyOn signal is '1'.
Power Off Fail Enable Reset	Numeric (Boolean)	If this setting is '1', VCU would keep trying to power off at certain frequency when power-down process fails.
Enable NVM Store and Restore	Check box	If Checked: Enable NVM control option.
Set The Waiting Time(ms)	Numeric	Power-off delay time

4.11.2 Shutdown Power

Folder: EcoCoder Blocks/System Management Blocks

Description:



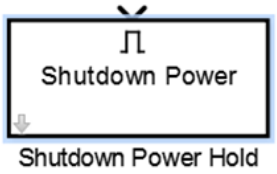
This block can be called to start VCU power-off process.

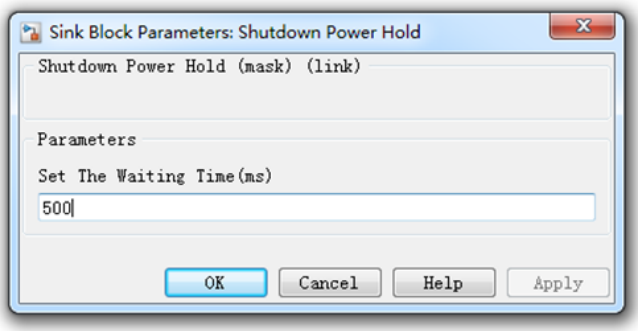
Block Parameters

Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Block trigger signal type selection.
Set The Waiting Time(ms)	Numeric	Set the time of power-off delay waiting time.

4.11.3 Shutdown Power Hold

This block is used to power off the controller.





Parameter:

1) Set the waiting time(ms): set the waiting time before powering off the controller.

Input:

1) Trigger signal: set 1 to power off the controller.

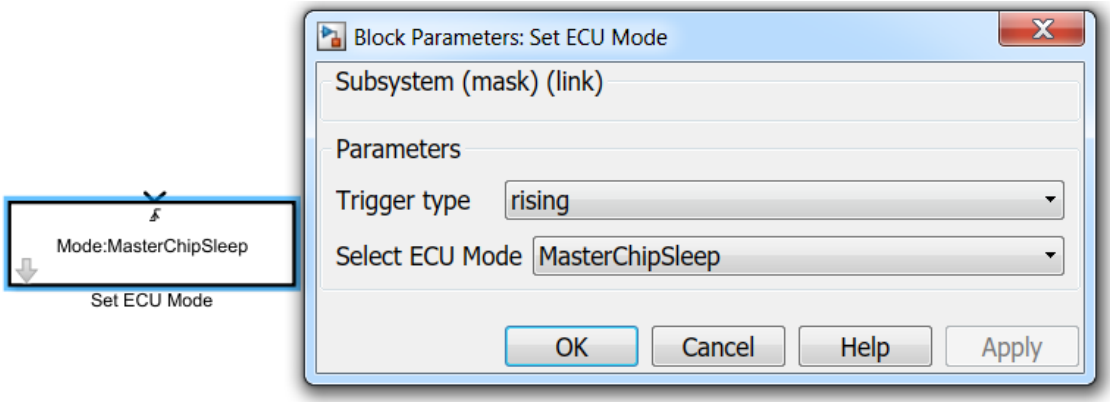
Note:

This block will be powered off when there is no other wake up signals' presence.

4.11.4 Set ECU Mode

Folder: EcoCoder Blocks/System Management Blocks

Description:



This module can set the working mode of ECU.

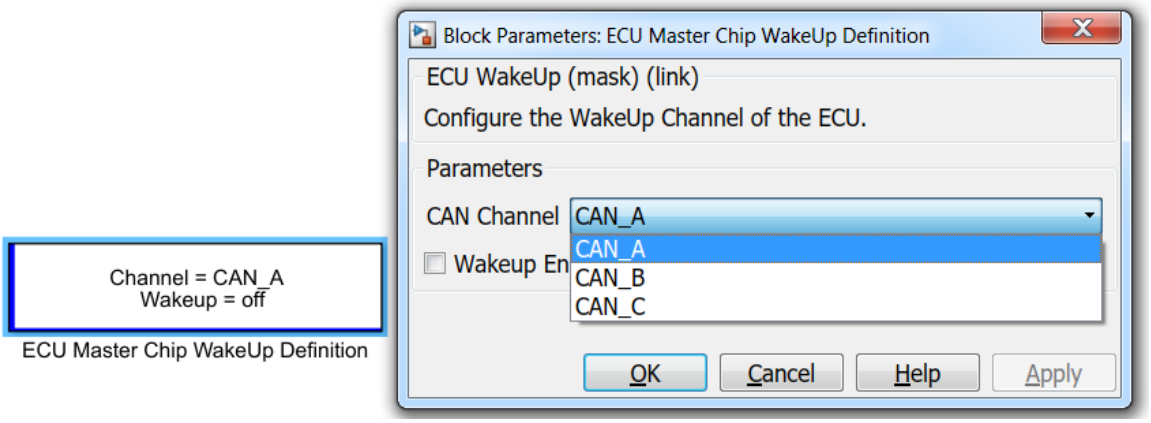
Block Parameters

Parameter Field	Value	Comments/Description
Trigger Type	Drop-down list	Block trigger signal type selection.
Select ECU Mode	Drop-down list	Work mode selection

4.11.5 ECU Master Chip Wake-Up Definition

Folder: EcoCoder Blocks/System Management Blocks

Description：



The block specifies the CAN channel that wakes up the VCU.

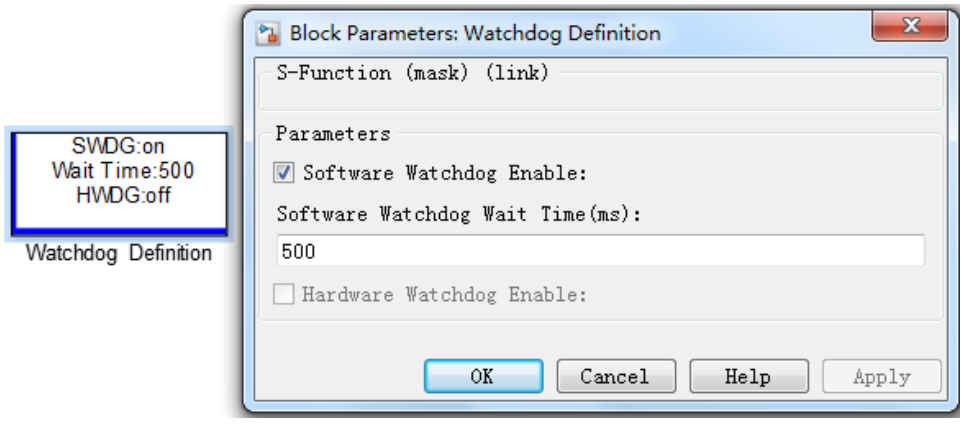
Block Parameters

Parameter Field	Value	Comments/Description
Trigger Type	Drop-down list	Wake-up CAN channel selection
Wakeup Enable	Check box	If checked: the specified CAN channel can wake up VCU.

4.11.6 Watchdog Definition

Folder: EcoCoder Blocks/System Management Blocks

Description:



Settings for software watchdog and hardware watchdog.

Block Parameters

Parameter Field	Value	Comments/Description
Software Watchdog Enable	Check box	If checked: Software watchdog is enabled.
Software Watchdog Wait Time(ms)	Numeric	The 'feeding dog' operation is executed at Task_L1ms, software will reset when timeout.
Hardware Watchdog Enable	Check box	If checked: Hardware watchdog enabled. (If this icon is greyed out, the specified VCU has no hardware watchdog built in)

4.11.7 Service Software Watchdog

Folder: EcoCoder Blocks/System Management Blocks

Description:

Software watchdog is used for resetting VCU software if the watchdog internal counter times out.

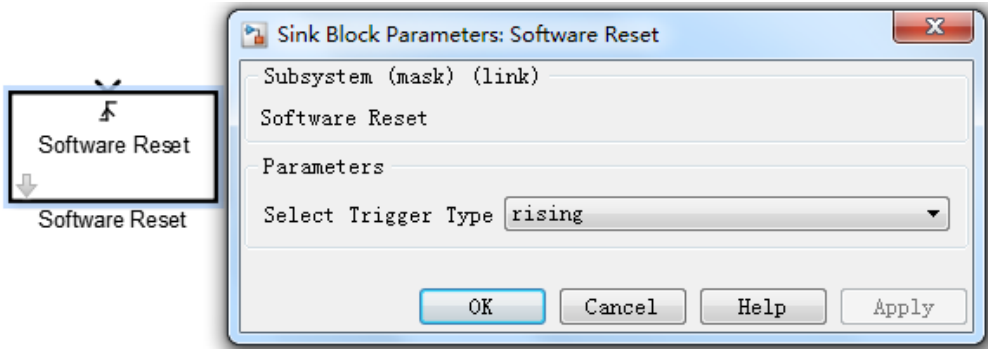
To enable this block, simply drag this block into your application software and schedule it as a low priority task using task scheduler. Every time this block being triggered by task scheduler will be taken as 'feed dog'. As a result, the scheduling period should be less than software watchdog timeout threshold.

Service Software Watchdog

4.11.8 Software Reset

Folder: EcoCoder Blocks/System Management Blocks

Description：



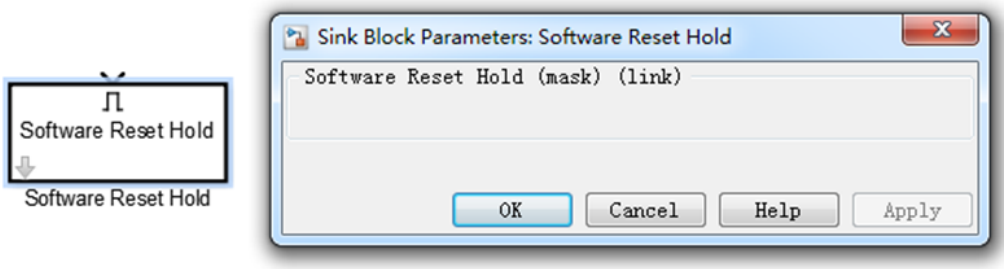
This block is used for triggering VCU software reset. If this block is called, the VCU software will be reset immediately.

Block Parameters

Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Block trigger signal type selection.

4.11.9 Software Reset Hold

This block is used for software reset.



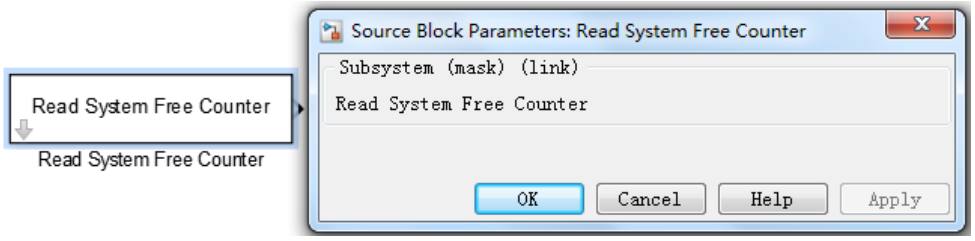
Input:

1) Trigger Signal: send reset signal if the trigger signal is 1

4.11.10 Read System Free Counter

Folder: EcoCoder Blocks/System Management Blocks

Description:



By calling the block, VCU main controller 32-bit free-running counter value will be read. The value can be used to calculate time interval between certain events or to generate random numbers, etc.

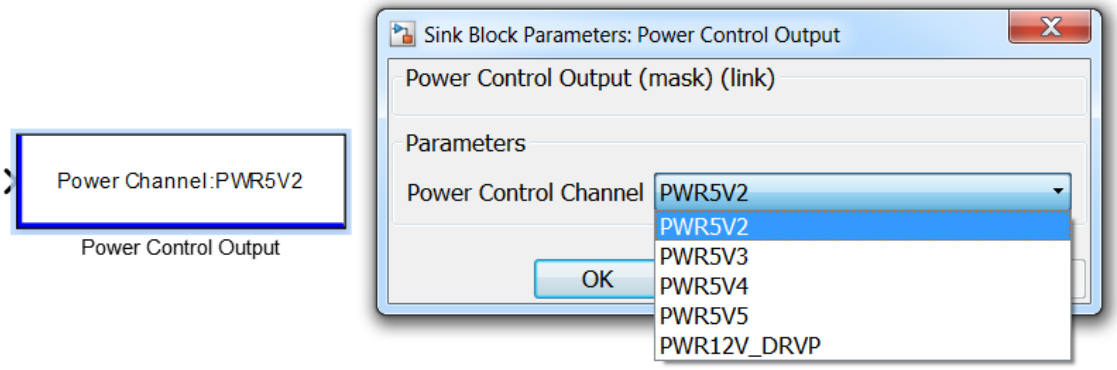
Block Output:

System free counter value.

4.11.11 Power Control Output

Folder: EcoCoder Blocks/System Management Blocks

Description:

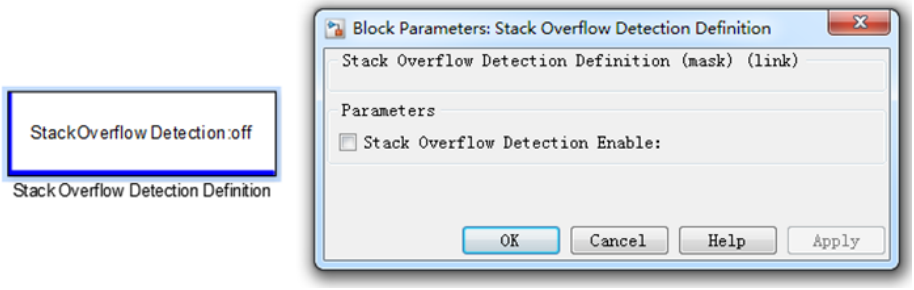


Block Parameters

Parameter Field	Value	Comments/Description
Power Control Channel	Drop-down list	Power channel selection.
input	Boolean	1: turning on power for corresponding channel. 0: turning off power for corresponding channel.

4.11.12 Stack Overflow Detection Definition

This block initializes the stack overflow detection.

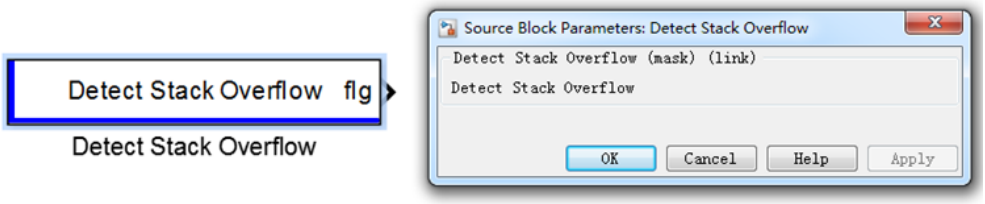


Parameter:

1) Stack Overflow Detection Enable: Check to enable the stack overflow detection.

4.11.13 Detect Stack Overflow

This block is used for detecting stack overflow.

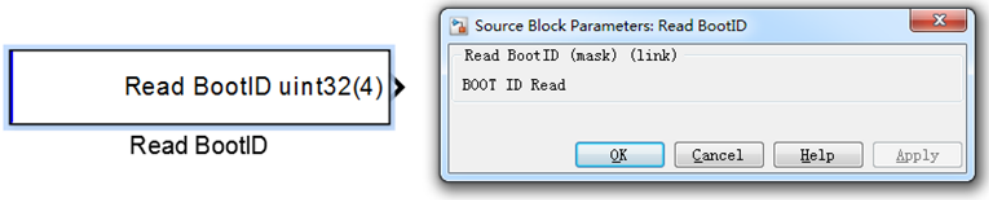


1) flg: Output 1 indicates stack overflow detected. Output 0 indicates stack overflow is not

detected.

4.11.14 Read BootID

This block is used for reading the hardware serial number.

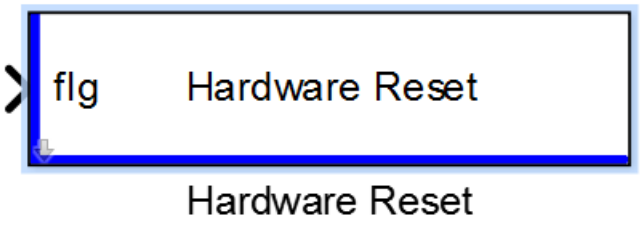


Output:

1) Uint32(4): 4 variables in unit32

4.11.15 Hardware Reset

This block is used for resetting the controller on the hardware level.



Parameter:

1) flg: send hardware reset command when reset flag is set to 1

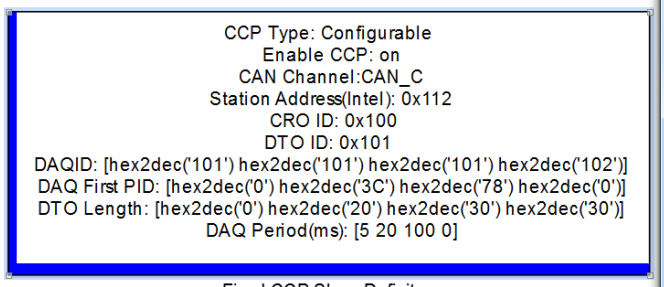
4.12 CCP

This block set includes CCP related implementations.

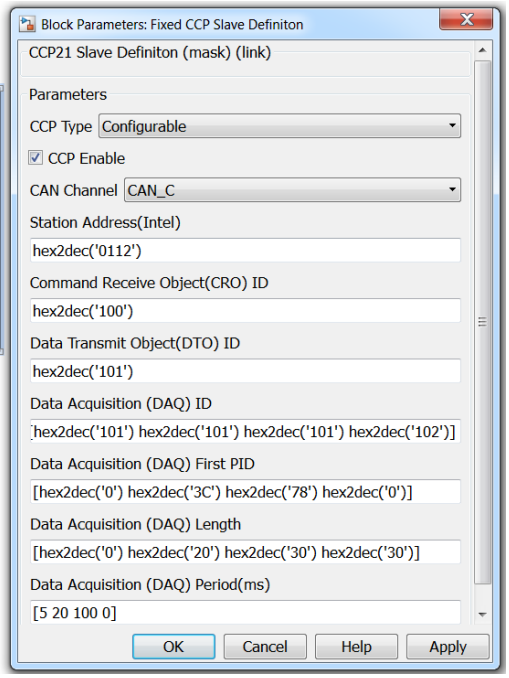
4.12.1 Fixed CCP Slave Definition

Folder: EcoCoder Blocks/CCP

Description:



Fixed CCP Slave Definiton



This block is used for setting up CCP related slave station definitions

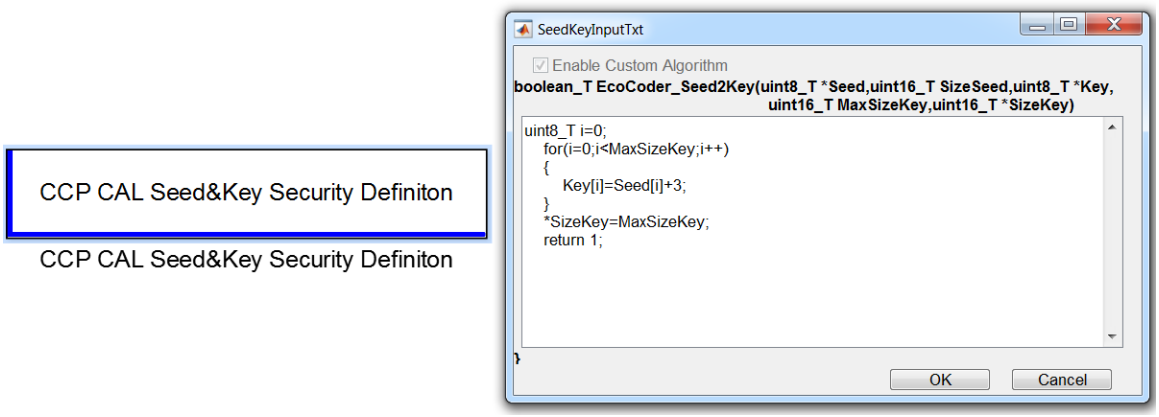
Block Parameters

Parameter Field	Value	Comments/Description
CCP Type	Drop-down list	Simple: Under this option, the only configurable parameter in this block is CAN channel. After building, users will get A2L, CAL and Mot (or Hex) file. To use EcoCAL, both A2L and CAL files need to be loaded. Configurable: Under this option, after building, users will get A2L and Mot (or Hex) file. To

		use EcoCAL, A2L and Mot (or Hex) files are needed.
CCP Enable	Check box	CCP enable
CAN Channel	Drop-down list	Specify CAN channel for CCP
Station Address(Intel)	Numeric	CCP Station Address
Command Receive Object(CRO) ID	Numeric	Specify Command Receive Object (CRO) ID (Master->Slave)
Data Transmit Object(DTO) ID	Numeric	Data Transmit Object (DTO) ID (Slave -> Master)
Data Acquisition (DAQ) ID	Numeric	CCP DAQ ID
Data Acquisition (DAQ) PID	Numeric	The first PID in the DAQ list.
Data Acquisition (DAQ) Length	Numeric	DAQ list length
Data Acquisition (DAQ) Period(ms)	Numeric	CCP DAQ period

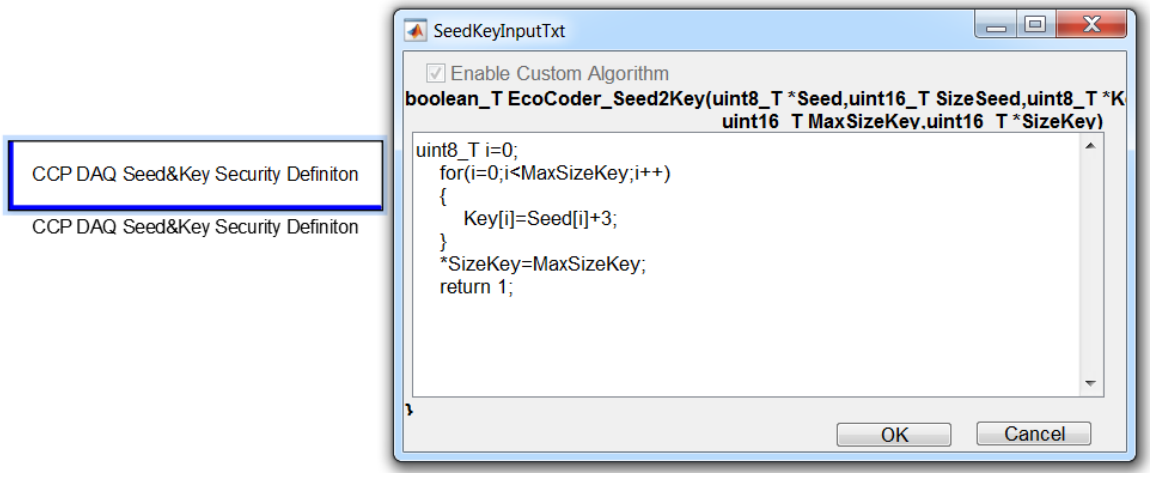
4.12.2 CCP/CAL Seed&Key Security Definition

This block is used to add authentication for calibration, the encryption algorithm of which can be customized. It also generates DLL file based on the user-provided seeds. The DLL file name is the name of MOT file plus "_CAL".



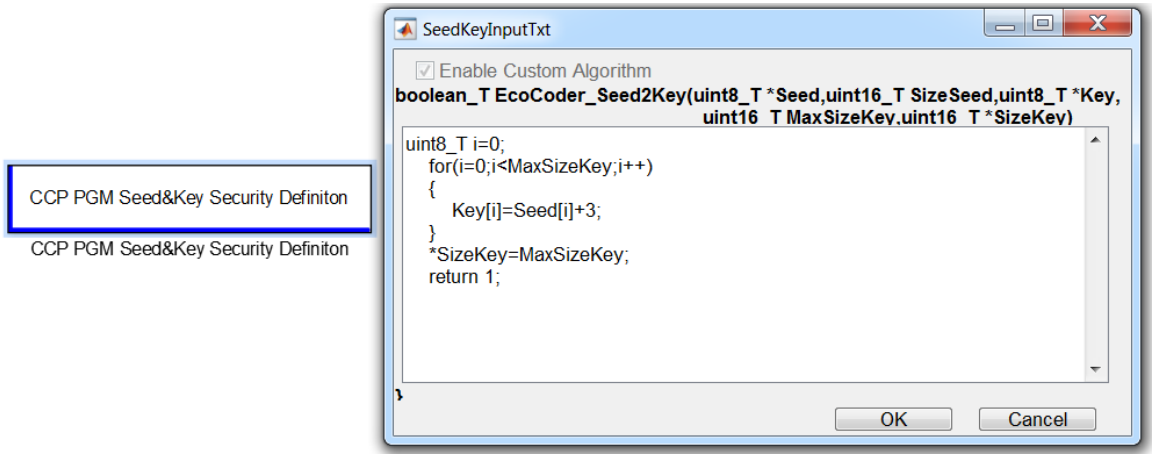
4.12.3 CCP DAQ Seed&Key Security Definition

This block is used to add authentication to the data measurement, the algorithm of which can be customized. It also generates DLL file based on the seeds. The DLL file name is the name of MOT file plus "_DAQ".



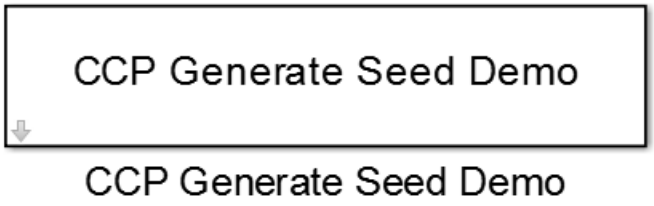
4.12.4 CCP PGM Seed&Key Security Definition

This block is used to add authentication to the data flash, the algorithm of which can be customized. It also generates DLL file according to the seeds. The DLL file name is the name of MOT file plus "_PGM".



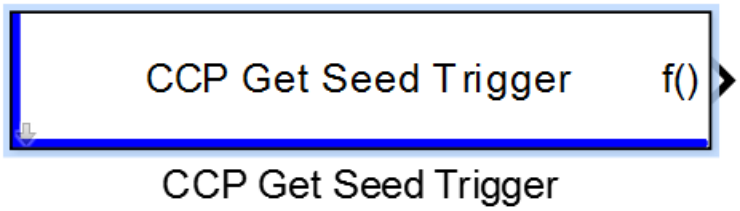
4.12.5 CCP Generate Seed Demo

This is a demo block for generating seeds. It is valid when the CCP Type of the *Fixed CCP Slave Definition* block is chosen to be “Configurable”. This module is implemented using the CCP Get Seed Trigger and CCP Set Seed blocks, it can be used as a demonstration Seed&Key function. This block is built for helping users to understand the process of seedkey generation. Users can customize this block as they need.



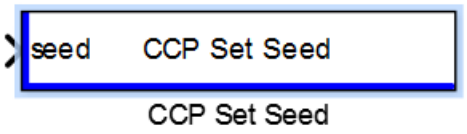
4.12.6 CCP Get Seed Trigger

This is getting seed task trigger block. It is valid when the *CCP Type* of the *Fixed CCP Slave Definition* block is chosen “Configurable”. Please refer to the *CCP Generate Seed Demo* block for details.



4.12.7 CCP Set Seed

This is setting seed block. It is valid when the CCP Type of the *Fixed CCP Slave Definition* block is chosen "*Configurable*". Please refer to the *CCP Generate Seed Demo* block for details.



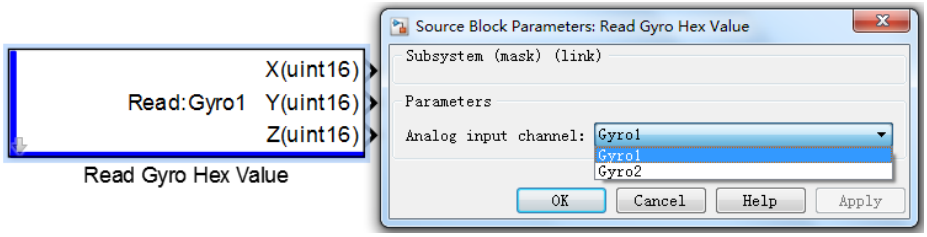
4.13 Sensors Blocks

4.13.1 Read Gyro Hex Value

Note that this block can only work with some specific VCU models. If the VCU that the user is working on does not support this function, there will be a pop-up notice when using this block.

The module reads the Hex values of the angular acceleration of the three axes of the gyroscope and outputs the raw data in uint16.

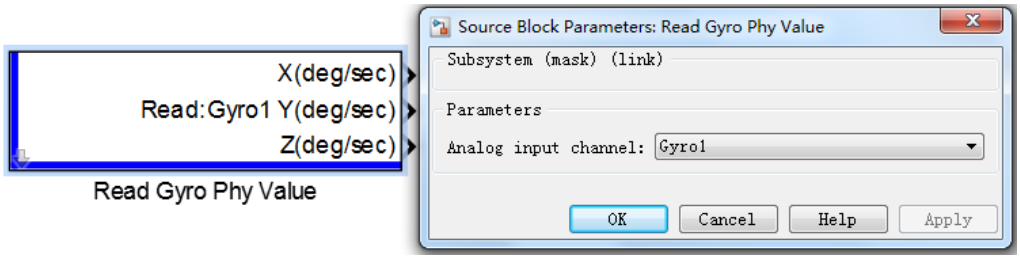
Raw data need to be multiplied by a slope of 0.05 and an accounted for an offset of -819.15 to give the acceleration physical value in deg / sec.



4.13.2 Read Gyro Phy Value

Note that this block can only work with some specific VCU models. If the VCU that the user is working on does not support this function, there will be a pop-up notice when using this block.

The module reads the angular acceleration of the three axes of the gyroscope and the output is physical value in deg / sec, data type is single.

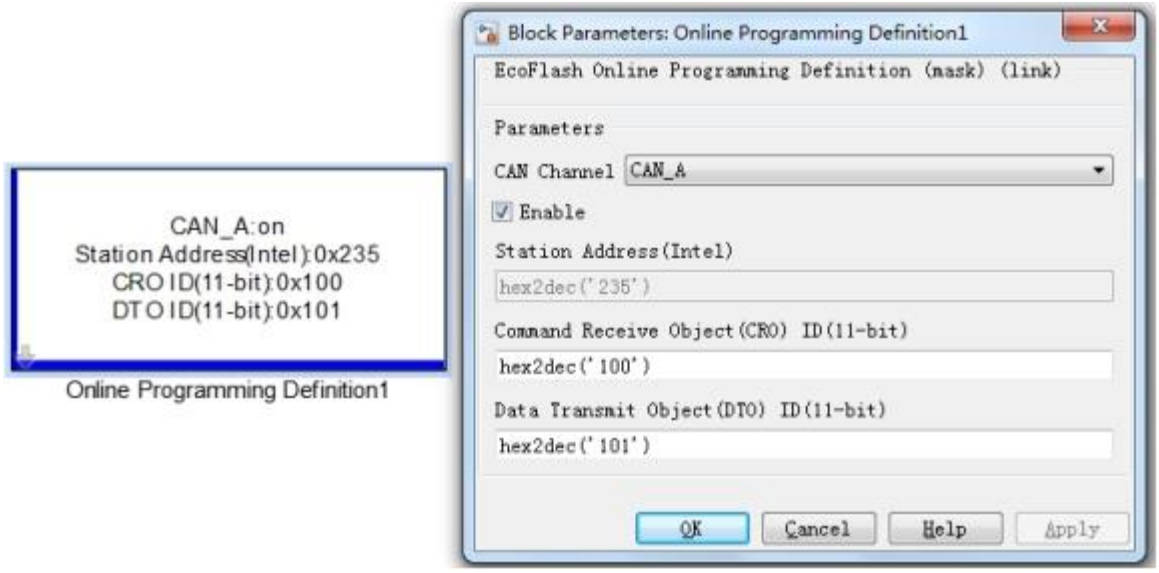


4.14 Programming Blocks

4.14.1 Online Programming Definition

Folder: EcoCoder Blocks/CCP

Description:



This block is used for the online programming parameter definition. Note that this block can only work with some specific VCU models. If the VCU that the user is working on does not support this function, there will be a pop-up notice when using this block.

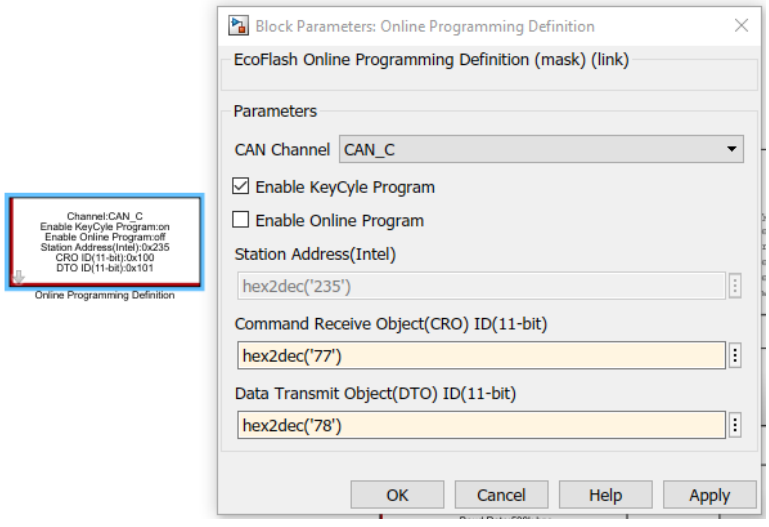
Users can use more than 1 of these blocks in their Simulink model to make multiple CAN channels support online programming function.

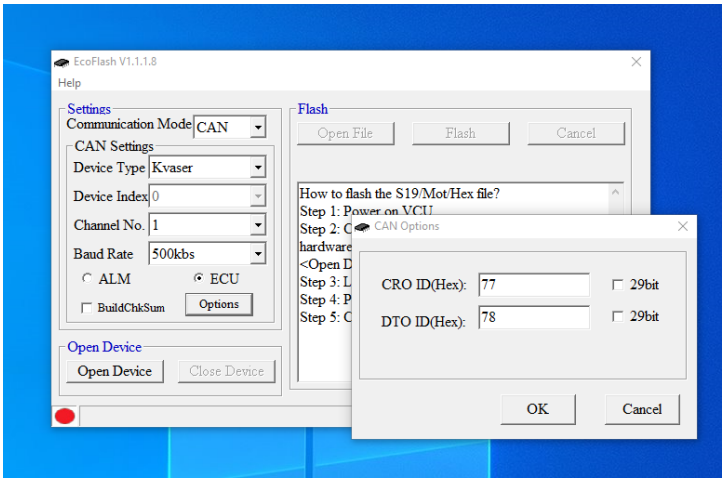
Block Parameters:

Parameter Field	Value	Comments/Description
Select CAN Channel	Drop-down list	Online programming CAN channel selection.
Enable KeyCycle Program	Check box	If checked: programming will require key cycle to start programming.
Enable Online Program	Check box	If checked: VCU programming will not require a key cycle.
Station Address (Intel)	Greyed out	This value cannot be changed for now.

Command Receive Object (CRO) ID (11-bit)	Alpha-numeric text	Specify Command Receive Object (CRO) ID (Master->Slave) It is recommended to remain as default.
Data Transmit Object (DTO) ID (11-bit)	Alpha-numeric text	Data Transmit Object (DTO) ID (Slave -> Master) It is recommended to remain as default.

Note: if you have more than 1 VCU on one CAN bus and you want to flash one of them, you need to change the CRO and DTO both in this block and in EcoFlash to make them the same, like shown below.

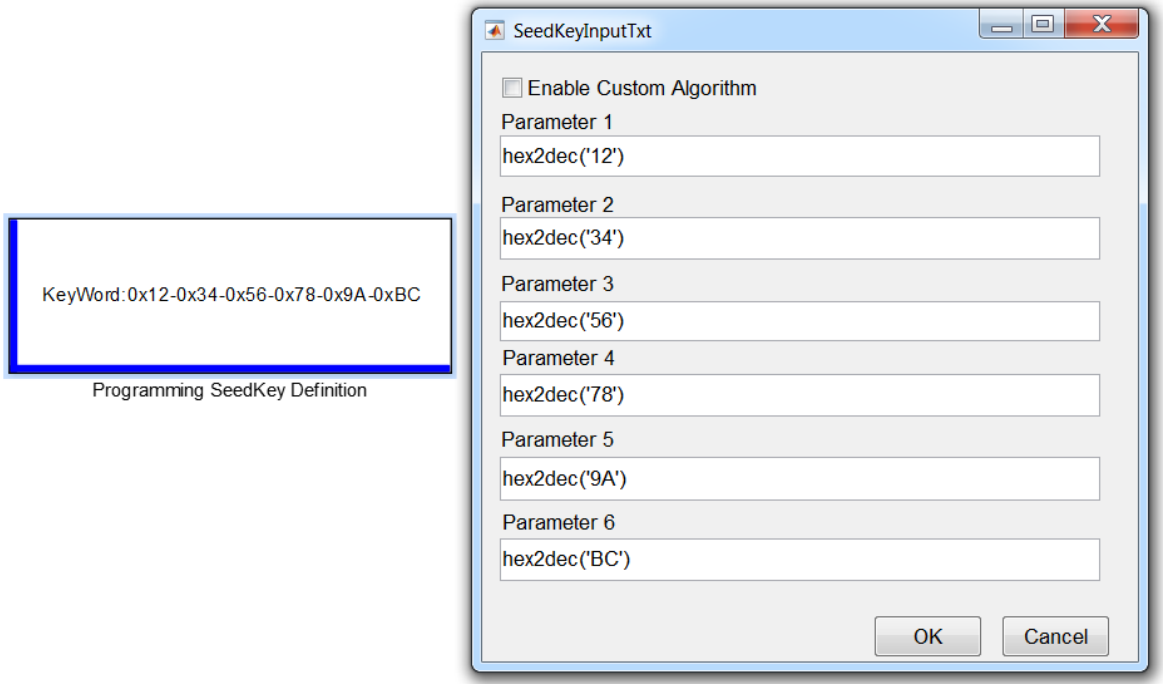




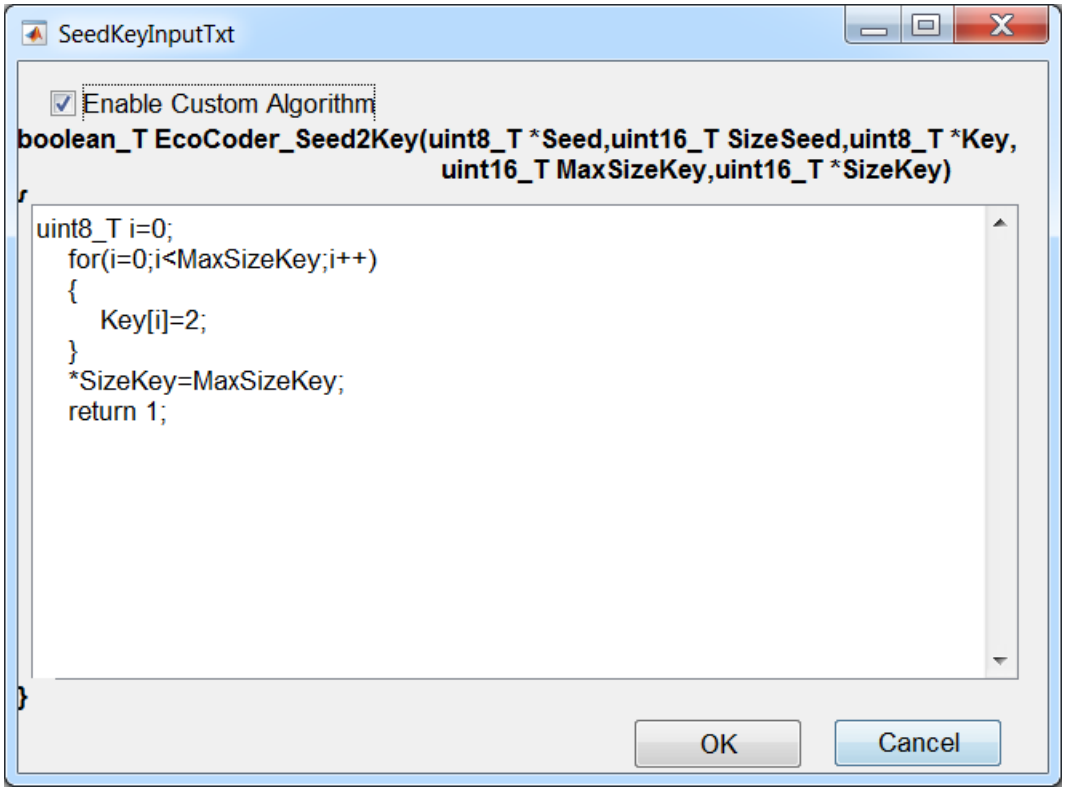
4.14.2 Programming Seed&Key Definition

The SeedKey function prevents unknown mot/hex files from being flashed to the VCU. The customer can use this block to define their own “password”. The DLL file and the MOT file will be generated at the same time when the building and compilation process is finished. The DLL file can be loaded in EcoFlash to authenticate the flashing process. Without this DLL file you generate, the data on VCU cannot be erased via EcoFlash. Please refer to the EcoFlash manual for more details.

This block can define and modify the flashing key. If the parameters keep unchanged, the default parameters will be used for flashing (0x12-0x34-0x56-0x78-0x9A-0xBC). If changed, the new parameters in the module will be used for generating the DLL file.



Programming SeedKey Definition

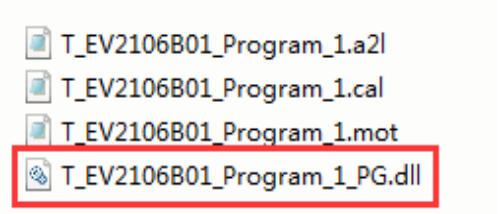


Parameters:

1.Enable Custom Algorithm: Enable the user-defined algorithm. If enabled, you can use a piece of C code to define key algorithm. If not, the key will be defined by setting the parameters.

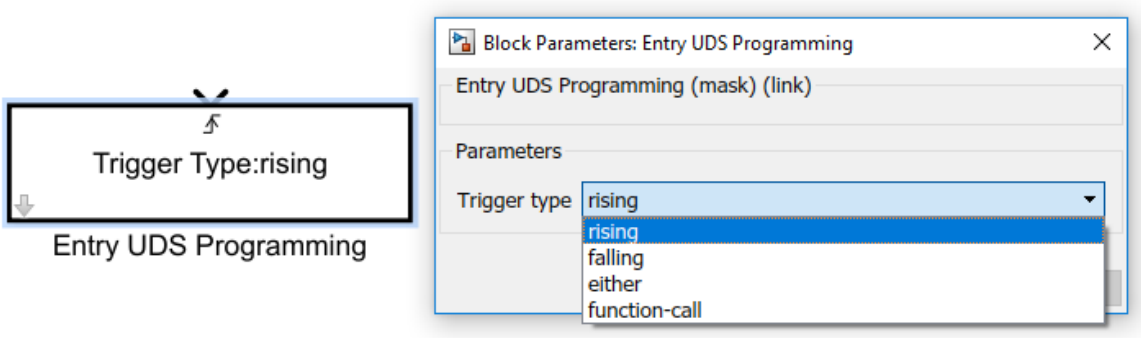
2. Parameter1 - Parameter6: Key flash setting, 6-byte key.

After automatically compiling and generating code, system will generate DLL file for key flash. EcoFlash will load the DLL file to match the key, if successfully matched, the flash is authenticated. Please refer the EcoFlash manual for details.



4.14.3 Entry UDS Programming

Enter the UDS programming mode by this block, which enables controller to update the program through the UDS protocol.

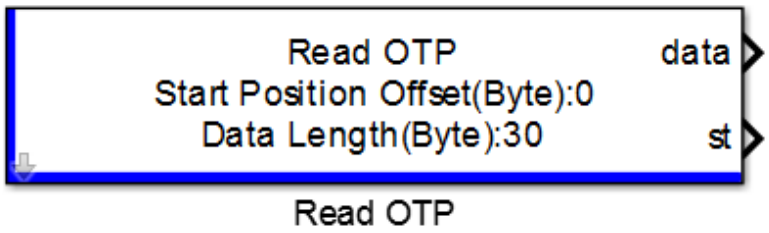


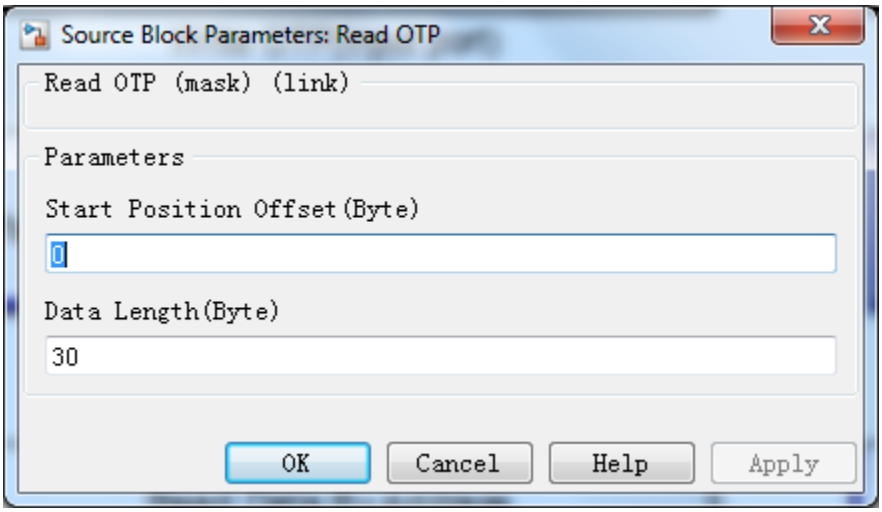
Parameter Field	Value	Comments/Description
Select trigger type	Drop-down list	Trigger type selection.

4.15 Advanced Data Blocks

OTP One-Time Programmable memory

4.15.1 Read OTP





Block Parameters:

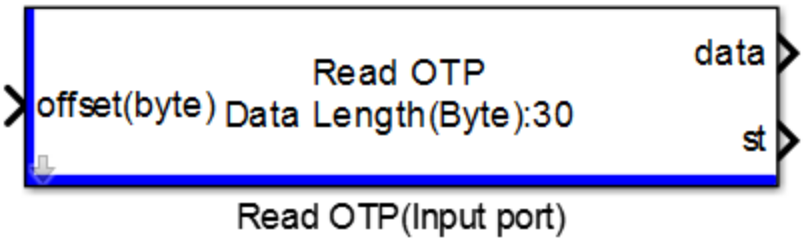
Parameter Field	Value	Comments/Description
Start Position Offset (Byte)	Numeric	Start address offset value
Data Length (Byte)	Numeric	Data length (the number of bytes the data takes)

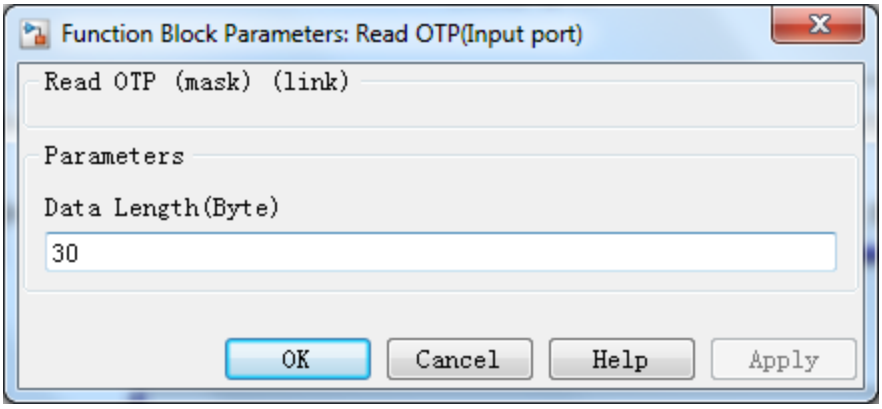
Block Outputs:

data: The data read from OTP area.

st: Data reading status, 0 stands for data reading successfully.

4.15.2 Read OTP (Input port)





Block Parameters:

Parameter Field	Value	Comments/Description
Data Length (Byte)	Numeric	The number of bytes that the data takes in OTP area.

Block Input:

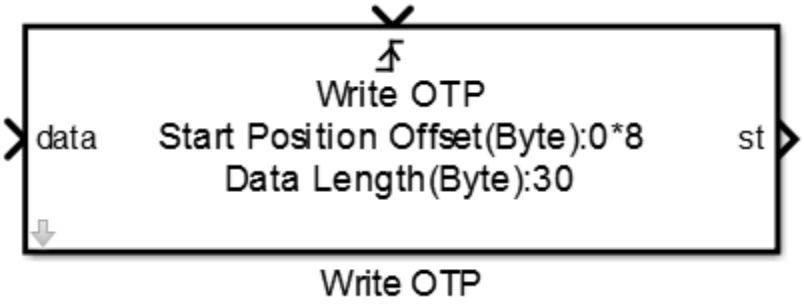
Offset (byte): the start position offset.

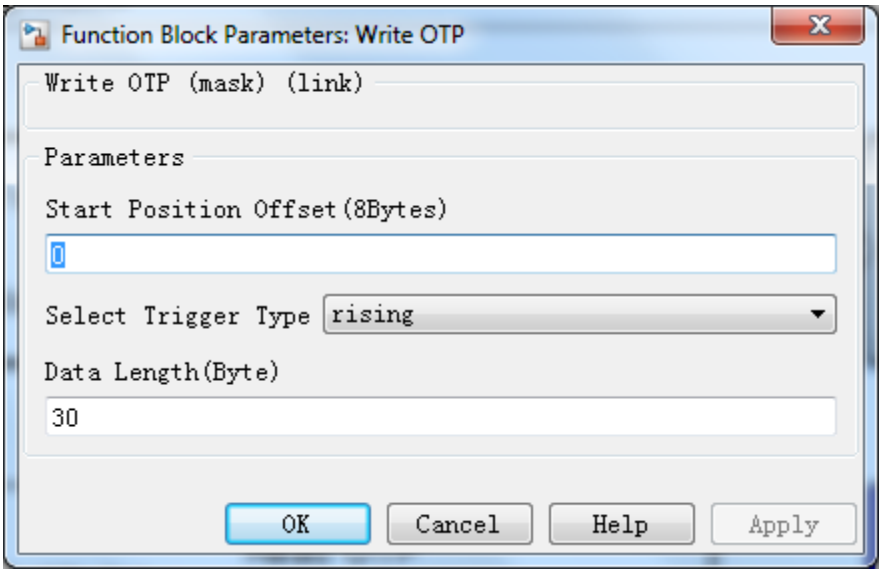
Block Output:

data: The data that has been read from OTP area.

st: Data reading status, 0 stands for data reading successfully.

4.15.3 Write OTP

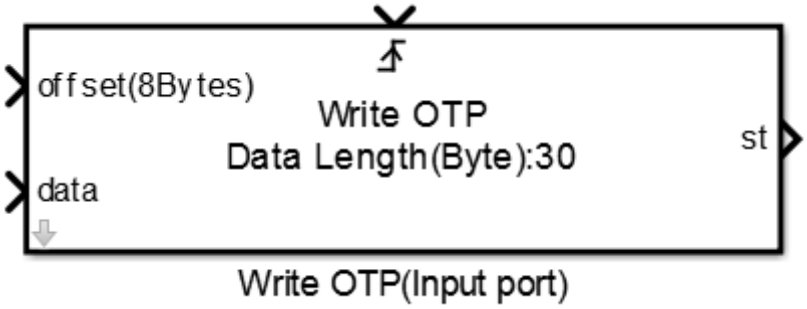


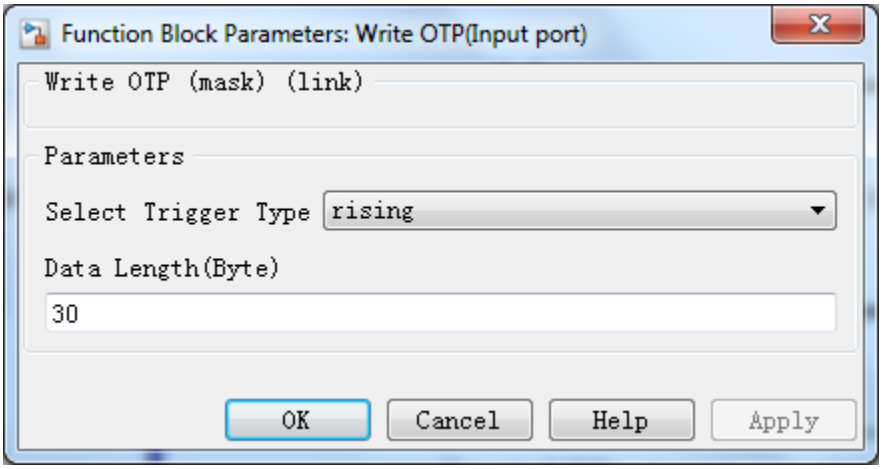


Block Parameters:

Parameter Field	Value	Comments/Description
Start Position Offset (8Bytes)	Numeric	Start address offset, the unit is 8 bytes.
Select Trigger Type	Drop-down list	Trigger type selection.
Data Length (Byte)	Numeric	The length of the data to be written.
Data (Block input)		The data to be written in OTP area.
st (Block output)	boolean	Data write status, 0 stands for writing successful; Non-0 value stands for writing unsuccessful.

4.15.4 Write OTP (Input port)





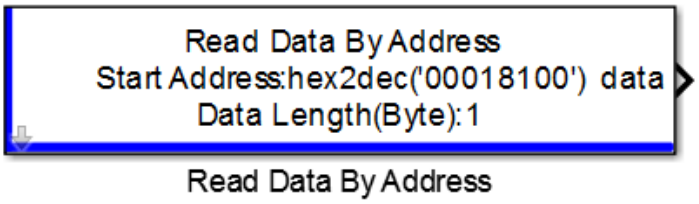
Block Parameters:

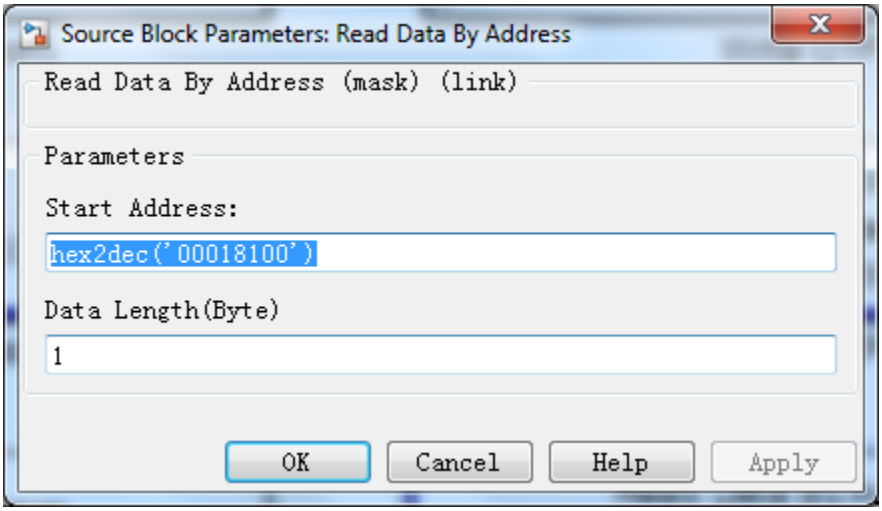
Parameter Field	Value	Comments/Description
Block Input: Start Position Offset (8Bytes)	Numeric	OTP writing start address offset, the unit is 8 bytes.
Select Trigger Type	Drop-down list	Block trigger type selection.
Data Length (Byte)	Numeric	The data length in byte.
Block Input:		Data input

data		
Block Output: st	boolean	Data write status, 0 stands for writing successful; Non-0 value stands for writing unsuccessful.

4.15.5 Read Data by Address

This block enables users to have access to memory by address.





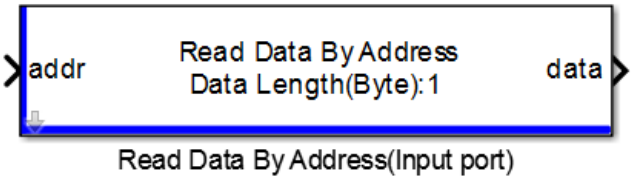
Block Parameters

Parameter Field	Value	Comments/Description
Start address	Numeric	Start address

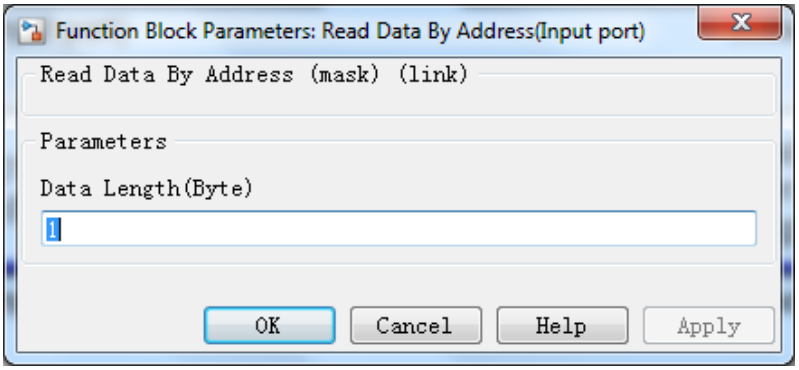
Data Length (Byte)	Numeric	Data length
Block Output: Data		Data output

4.15.6 Read Data by Address (Input port)

This block is the same as the “Read Data by Address” block, except for the method of specifying the address is changed. For this block, the address is specified by block input signal.



Read Data By Address(Input port)

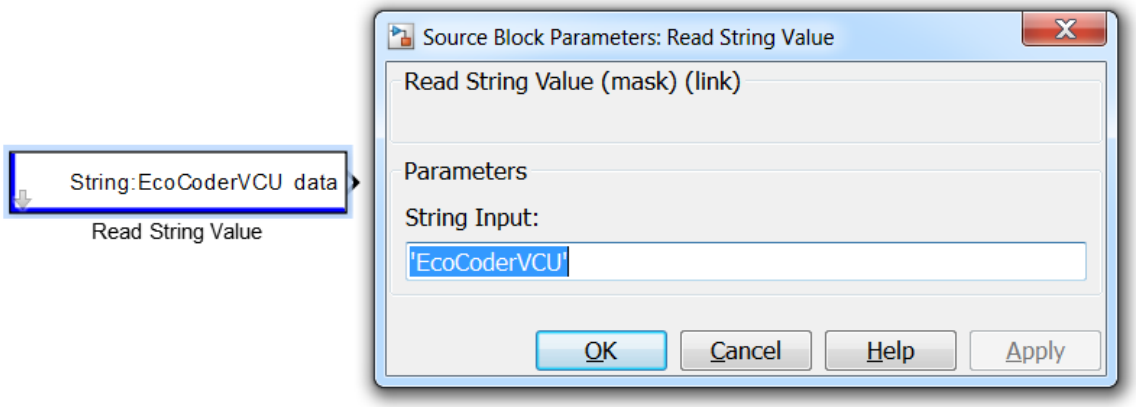


Block Parameters

Parameter Field	Value	Comments/Description
Start address(input)	Numeric	Start address
Data Length(Byte)	Numeric	Data read length
Block Output: data		Data output

4.15.7 Read String Value

This block can translate strings to corresponding ASCII numeric arrays.

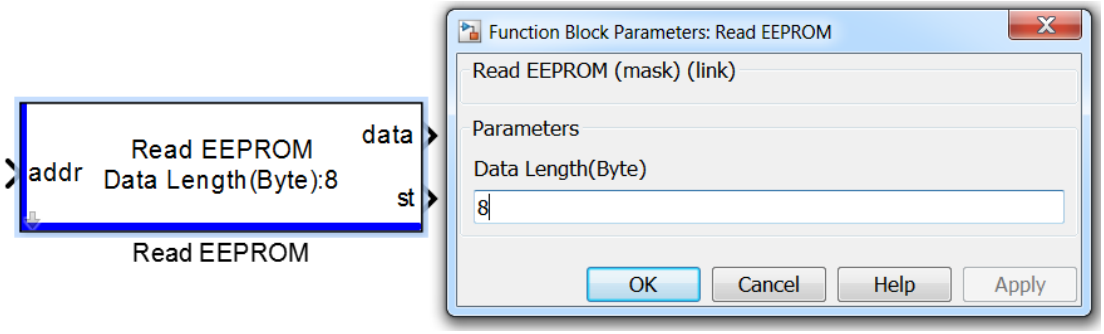


Block Parameters

Parameter Field	Value	Comments/Description
String Input	String	The string to be parsed
Block Output: Data	Numeric	The numeric arrays of the string ASCII.

4.15.8 Read EEPROM

This block reads data from EEPROM.



Parameter:

1) Data length: the length of the data to be read

Input:

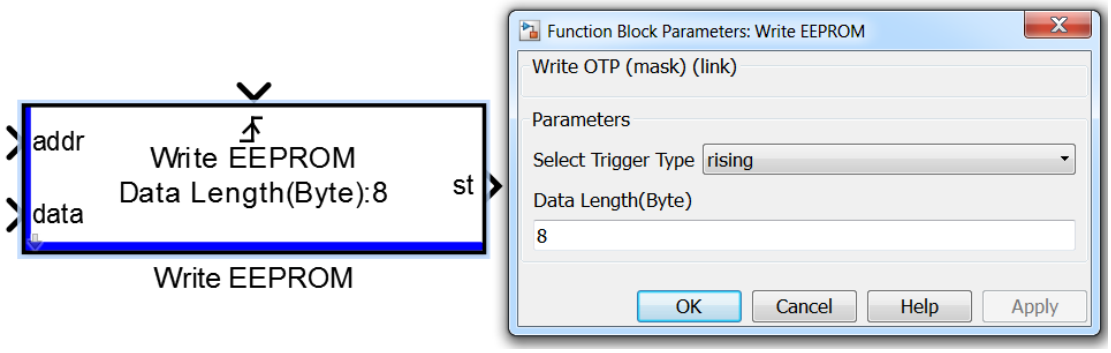
1) addr: the address of the data to be read

Output:

1) data: data read from EEPROM
2) st: Output status. Output 0 when data read successfully

4.15.9 Write EEPROM

This block is used for writing data to the EEPROM.



Parameter:

1) Select Trigger Type: trigger type selection
2) Data length: the length of the data to be written

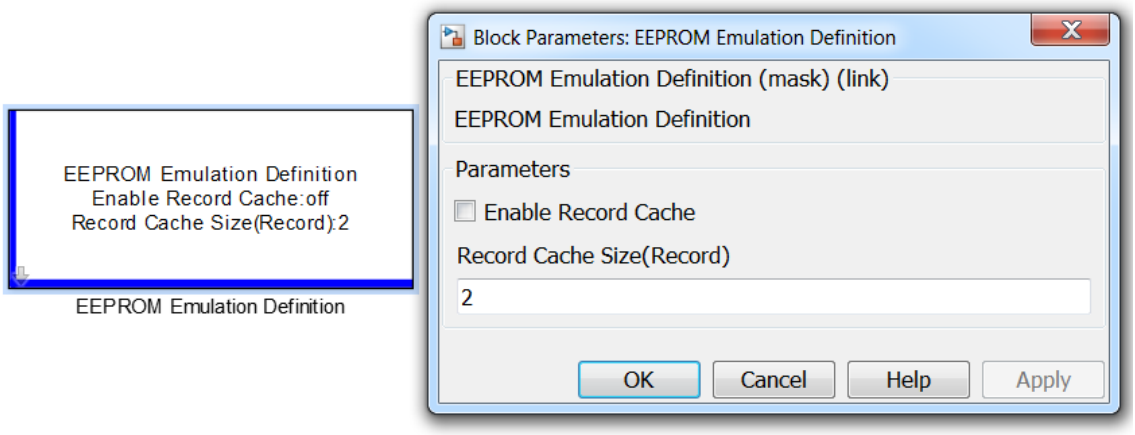
Input:

1) addr: the address to be written on
2) Data: the data to be written

Output:

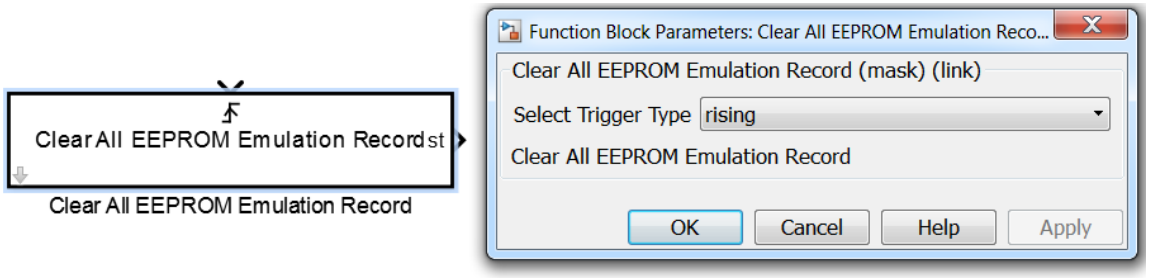
1) St: output status. Output 0 when data read successfully

4.15.10 EEPROM Emulation Definition



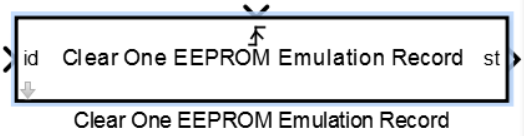
Parameter Field	Value	Comments/Description
Enable Record Cache	Check box	Enable logging of information, non-self-recording.
Record Cache Size (Record)	Numeric	The number of the cached record message.

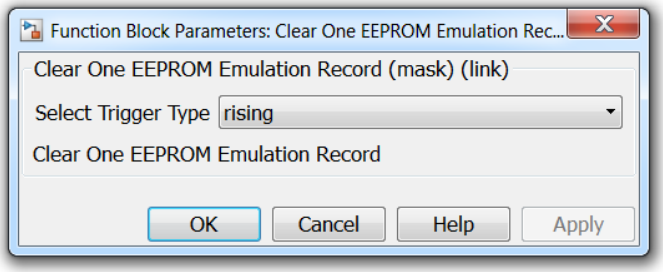
4.15.11 Clear ALL EEPROM Emulation Record



Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Trigger type selection
Block Output: st	Numeric	Output state, refer to Appendix Table 1.

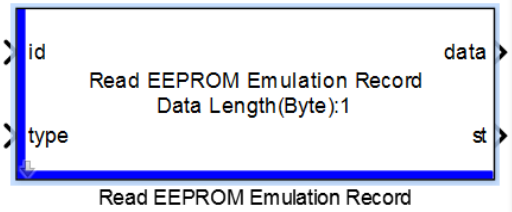
4.15.12 Clear One EEPROM Emulation Record

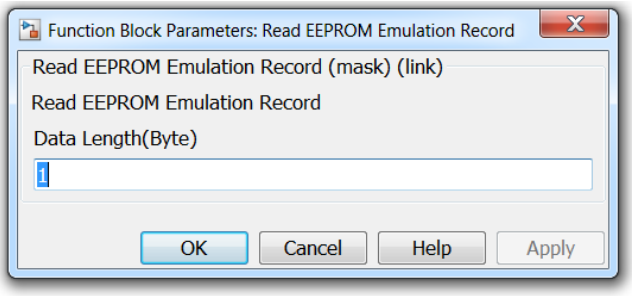




Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down list	Start address
Block Input: id		Record id
Block Output: st	Numeric	Output state, refer to Appendix Table 1

4.15.13 Read EEPROM Emulation Record

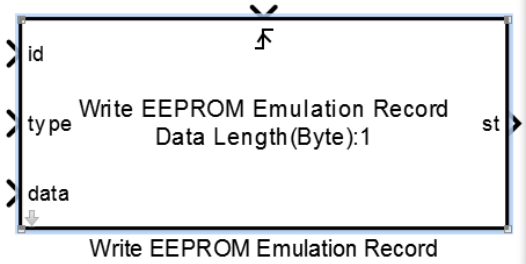


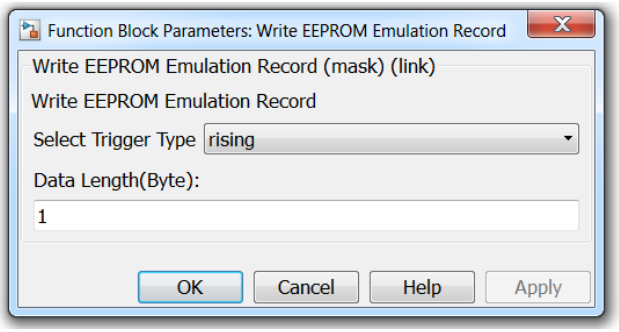


Parameter Field	Value	Comments/Description
Data Length (Byte)	Numeric	Data length in byte
Block Input: 1. id 2. type		1. The id of records to be read

		2. The type of records to be read, which inherits the data type of the previous block
Block Output: 1. data 2. st		1. Read Data 2. Output state, refer to Appendix Table 1

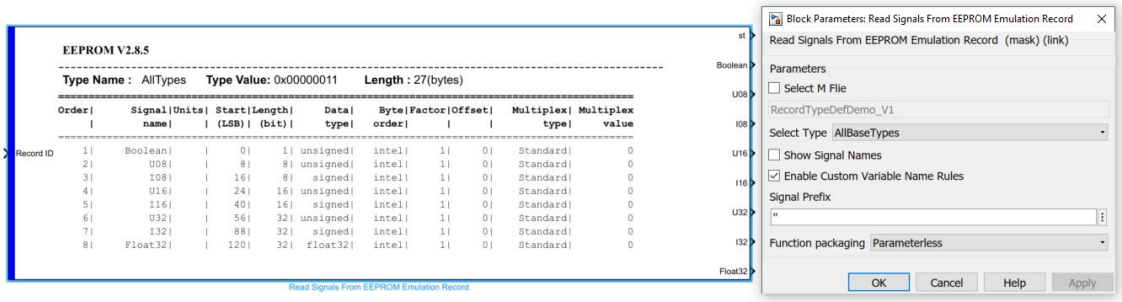
4.15.14 Write EEPROM Emulation Record





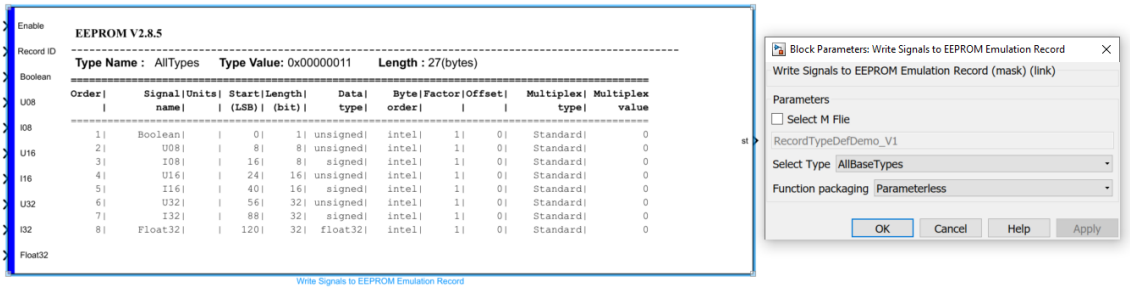
Parameter Field	Value	Comments/Description
Select Trigger Type	Drop-down	Trigger type selection
Data Length	Numeric	Data length
Block Input: 1. id 2. type 3. data		1. Id of records to be written 2. Type of records to be written 3. Data written
Block Output: st	Numeric	Output state, refer to Appendix Table 1

4.15.15 Read Signals from EEPROM Emulation Record



Parameter Field	Value	Comments/Description
Select M file	Check box	Add M file to MATLAB path, specify the M file name, then click “OK” and “APPLY”, finally double-click the module again to select the Message.
Select Type	Drop-down list	Record type selection
Show Signal Names	Check box	if enabled: the name of the signal will be displayed on the output signal line.
Signal Prefix		User-defined variable prefix on the signal line.
Block Input: Record ID		The record ID to be read and parsed
Block Output: st		Output state, refer Appendix Table 1 Note: The value of the signal after parsing record is the actual physical value

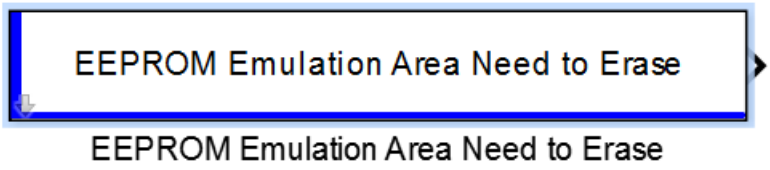
4.15.16 Write Signals to EEPROM Emulation Record



Parameter Field	Value	Comments/Description
Select M file	Check box	Add M file to MATLAB path, (users can change the default M file name) then click “OK” and “APPLY”, finally double-click the module again to select the Message.
Select Type	Drop-down list	Record type selection
Block Input: 1. Enable 2. Record ID		1. Write record enable control, it is recommended to control by edge triggering signal. 2. The record ID to be written
Block Output: st		Output state, refer Appendix Table 1

4.15.17 EEPROM Emulation Area Need to Erase

This block is used for determining whether if the EEPROM emulation area need to be erased.

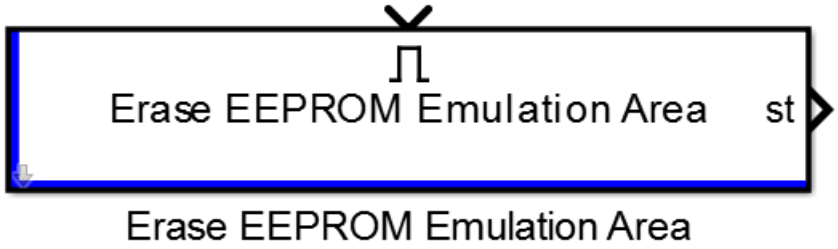


Output:

1) Flg: output 1 when the emulation area needs to be erased. Ouput 0 when the emulation area does not need to be erased.

4.15.18 Erase EEPROM Emulation Area

This block is used for erasing the EEPROM emulation area. This block needs to be used with “EEPROM Emulation Area Need to Erase” block.



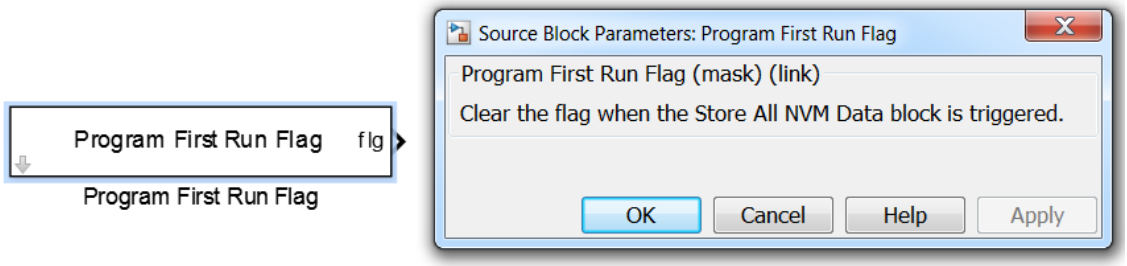
Input:

1) Trigger: input 1 to erase the emulation area

Output:

1) St: current state, please refer to the appendix for details

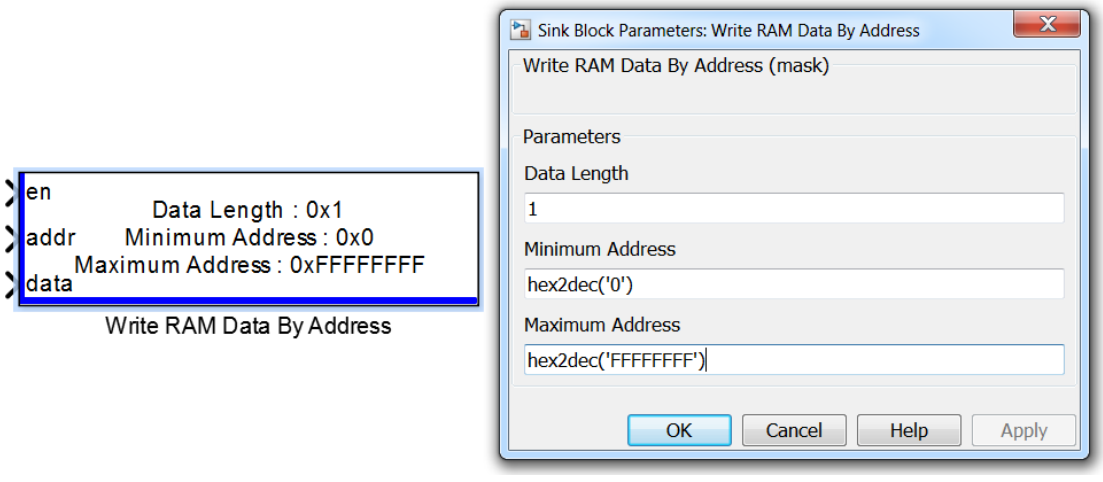
4.15.19 Program First Run Flag



Parameter Field	Value	Comments/Description
Block Output: Flg	Boolean	Outputs the flag value, 1 stands for the first run of the program. The flag becomes 0 when the Store ALL NVM Data module is used and triggered successfully.

4.15.20 Write RAM Data by Address

This block writes data to RAW based on the given address.



Parameter:

1) Data Length (Byte): Data length in byte

3) Minimum Address: no writing will be done when the address value is lower than this minimum value

4) Maximum Address: no writing will be done when the address value is higher than this maximum value

Input:

1) en: enable the writing operation

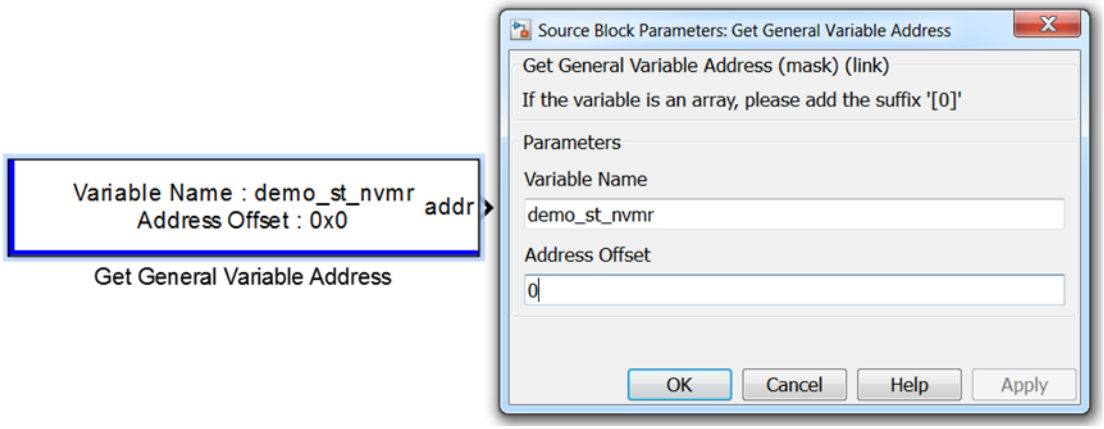
5) addr: start address

Output:

1) data

4.15.21 Get General Variable Address

This block can get address based on variable name. Use this block to work with arrays efficiently. Be careful on the array index boundary.



Parameter:

1) Variable Name: the name of the variable

2) Address offset: the offset of the address

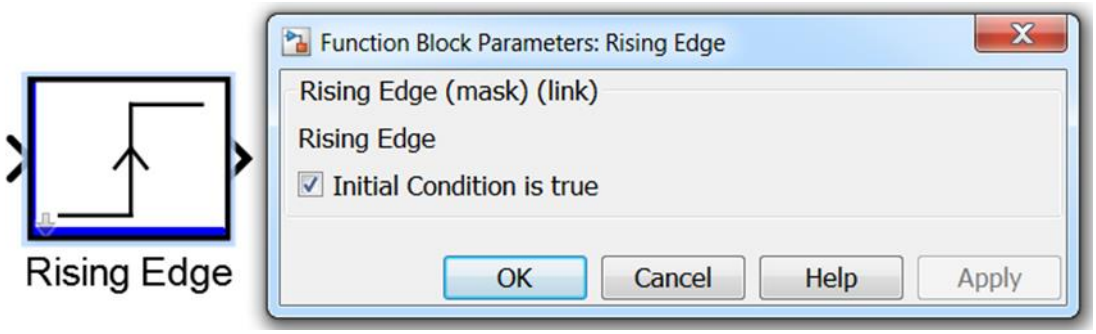
Output:

1) addr: the address of variable

4.16 Application Base Blocks

4.16.1 Rising Edge

This block is used for detecting rising edges.

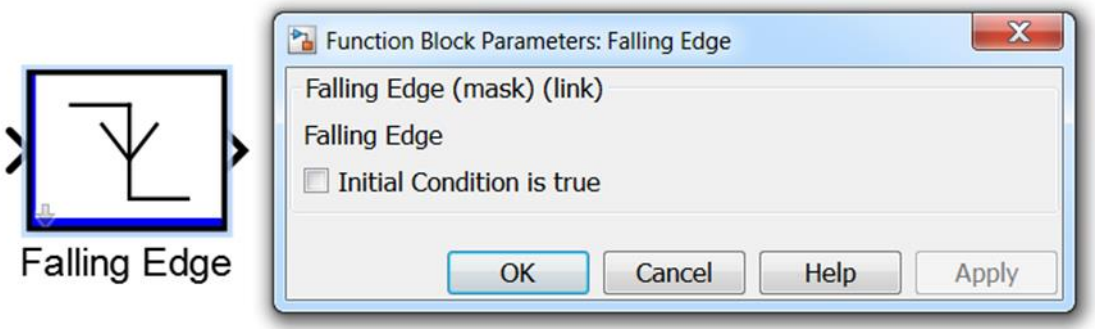


Parameter:

1) Initial Condition is true: initialization option, check the box to initialize the value as 1.

4.16.2 Falling Edge

This block is used for detecting falling edges.



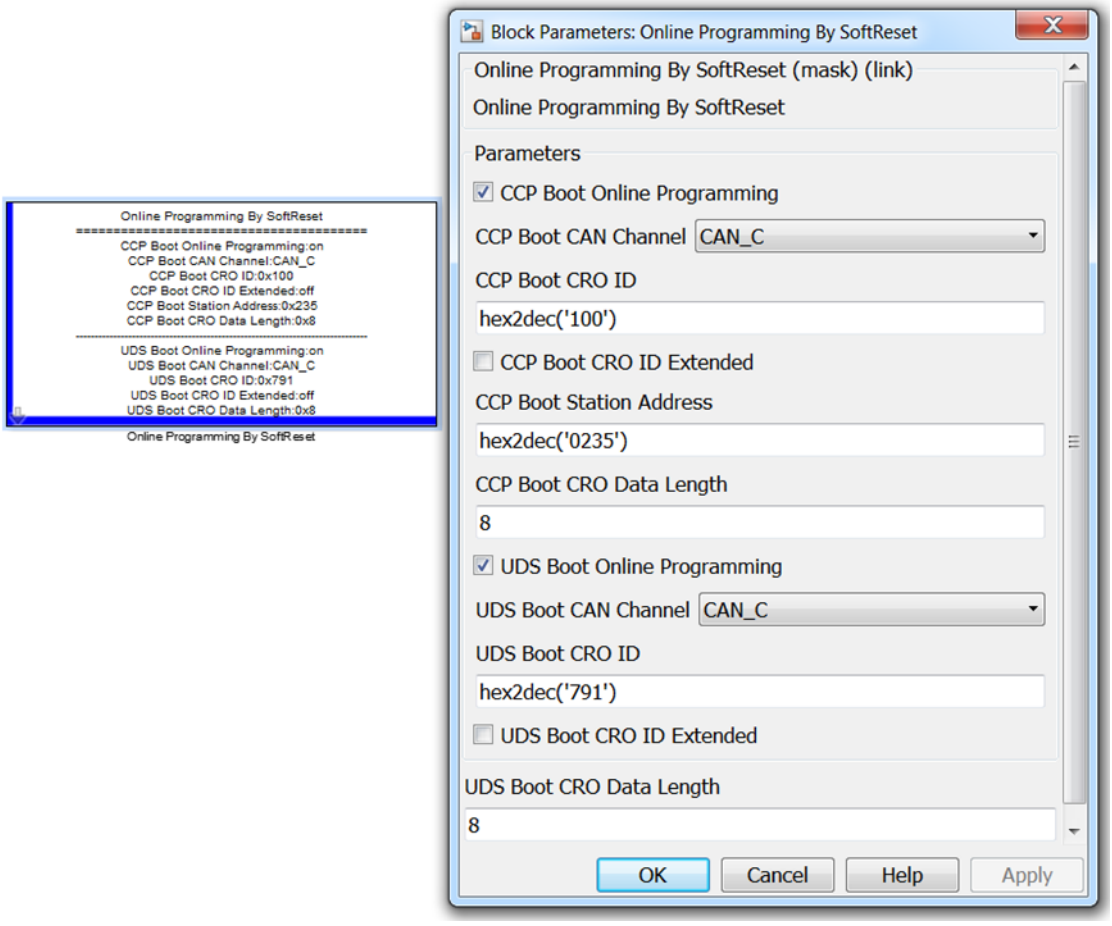
Parameter:

1) Initial Condition is true: initialization option, check the box to initialize the value as 1.

4.16.3 Online Programming by SoftReset

This block helps user to reprogram the controller online by reset the controller using

software.



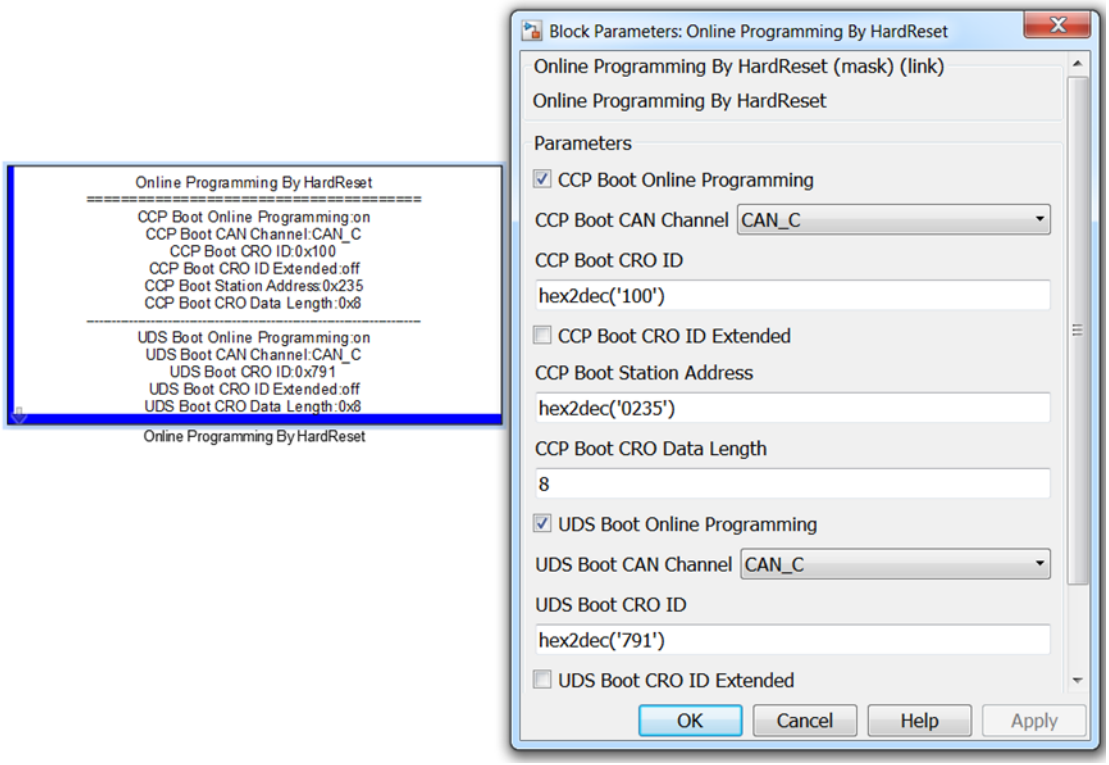
Parameter:

1) CCP Boot Online Programming: Check for enabling online programming
2) CCP Boot CAN Channel: Select CCP channel for online programming
3) CCP Boot CRO ID: Set CRO ID for online programming
4) CCP Boot CRO ID Extended: Select extended CRO frame for online programing
5) CCP Boot Station Address: Select CCP address for online programming station
6) CCP Boot CRO Data length: Set CCP CRO data length
7) UDS Boot Online Programming: Check to enable UDS online programming
8) UDS Boot CAN Channel: select UDS CAN channel for online programing
9) UDS Boot CRO ID: Set UDS ID for online programming
10) UDS Boot CRO ID Extended: Set extended CRO frame for UDS online programming

11) UDS Boot CRO Data Length: Set UDS CRO data length for online programming

4.16.4 Online Programing By HardReset

This block can reset the controller hardware for online programming.



Parameter:

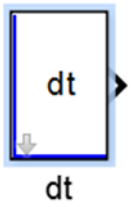
1) CCP Boot Online Programming: Check for enabling online programming
2) CCP Boot CAN Channel: Select CCP channel for online programming
3) CCP Boot CRO ID: Set CRO ID for online programming
4) CCP Boot CRO ID Extended: Select extended CRO frame for online programing
5) CCP Boot Station Address: Select CCP address for online programming station
6) CCP Boot CRO Data length: Set CCP CRO data length
7) UDS Boot Online Programming: Check to enable UDS online programming
8) UDS Boot CAN Channel: select UDS CAN channel for online programing
9) UDS Boot CRO ID: Set UDS ID for online programming

10) UDS Boot CRO ID Extended: Set extended CRO frame for UDS online programming

11) UDS Boot CRO Data Length: Set UDS CRO data length for online programming

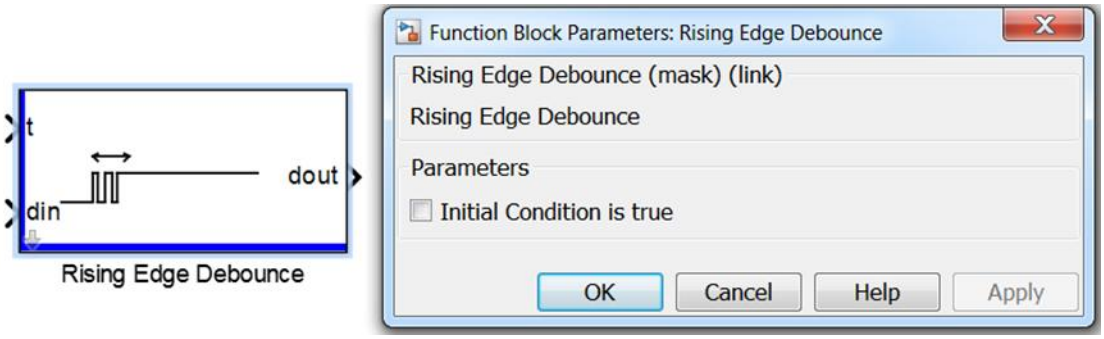
4.16.5 dt – time step length

This block is used for acquiring time step length for current task. (unit: second)



4.16.6 Rising Edge Debounce

This block debounces and/or delay the rising edge from input.



Parameter:

1) Initial Condition is true: Initializing option. Check to set initial value to 1.

Input:

1) t: rising edge duration. (unit: second)

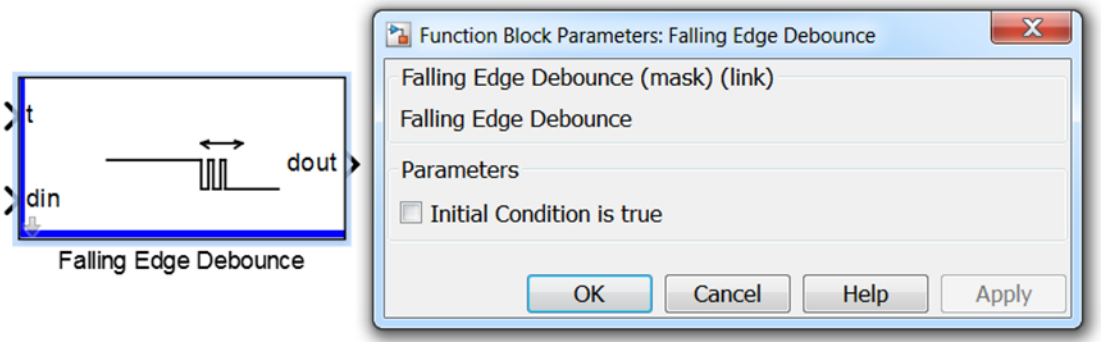
2) din: original input

Output:

1) dout: signal after debounce.

4.16.7 Falling Edge Debounce

This block debounces and/or delay the falling edge from input.



Parameter:

1) Initial Condition is true: Initialization option. Check to set initial value to 1.

Input:

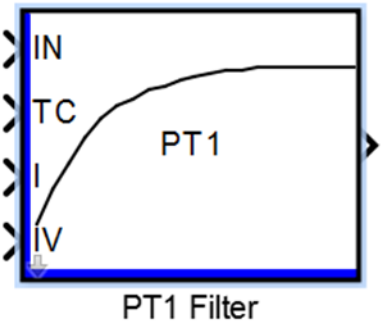
1) t: falling edge duration. (unit: second)
2) din: original input

Output:

1) dout: signal after debouncing.

4.16.8 PT1 Filter

This block is used for filtering signals.



Input:

1) IN: original input signal

12) TC: Low-pass filter factor (unit: second)

13) I: Enabling initialization. Set output value as IV

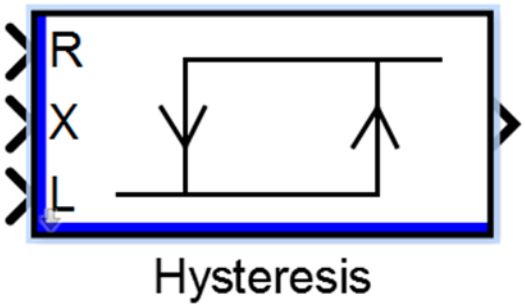
14) IV: Initialization value

Output:

1) Filtered signal

4.16.9 Hysteresis

This block converts analog signal to digital signal.



Input:

1) R: Upper threshold

15) X: Input signal

16) L: Lower threshold

Output:

1) Output signal: Output signal 1 when the input signal X is larger than upper threshold R. Output signal 0 or retaining the previous value when input signal X is smaller than L.

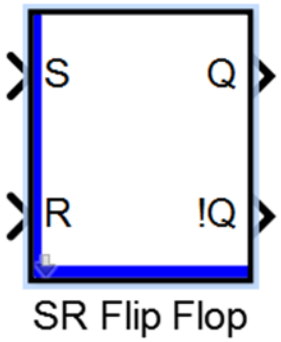
4.16.10 SR Flip Flop

!Q is the inverted Q.

When R=1 → Q=0

When R=0 and S=1 → Q=1

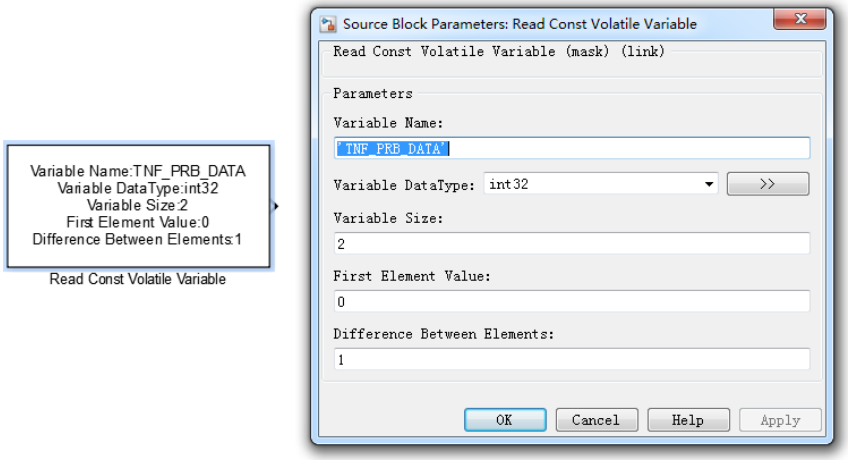
When R=0 and S=0 → Q=Previous value



4.17 Volatile Variable

4.17.1 Read Const Volatile Variable

This block is used for reading constant volatile variable.



Parameter:

1) Variable Name: the name of the variable
2) Variable DataType: the type of the data
3) Variable Size: the length of the variable
4) First Element Value: the value of the first element

5) Difference Between Elements: the difference between each element

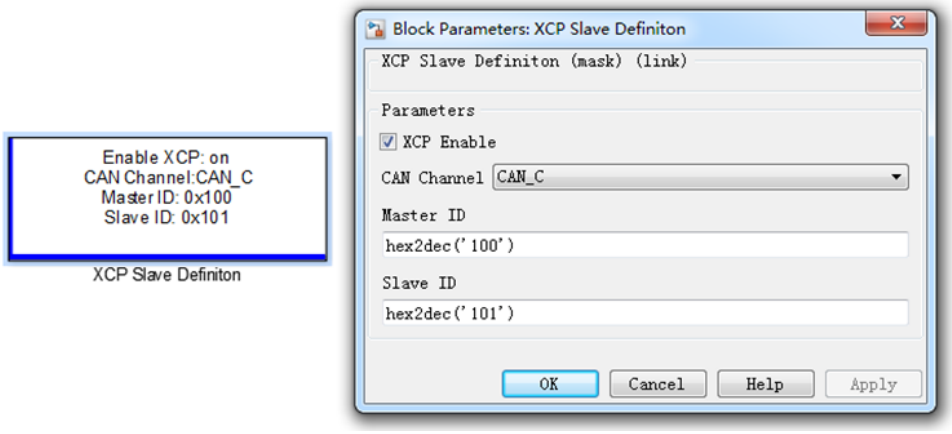
Output:

1) Read value from the specific variable

4.18 XCP Module

4.18.1 XCP Slave Definition

This block is used for initializing XCP module.



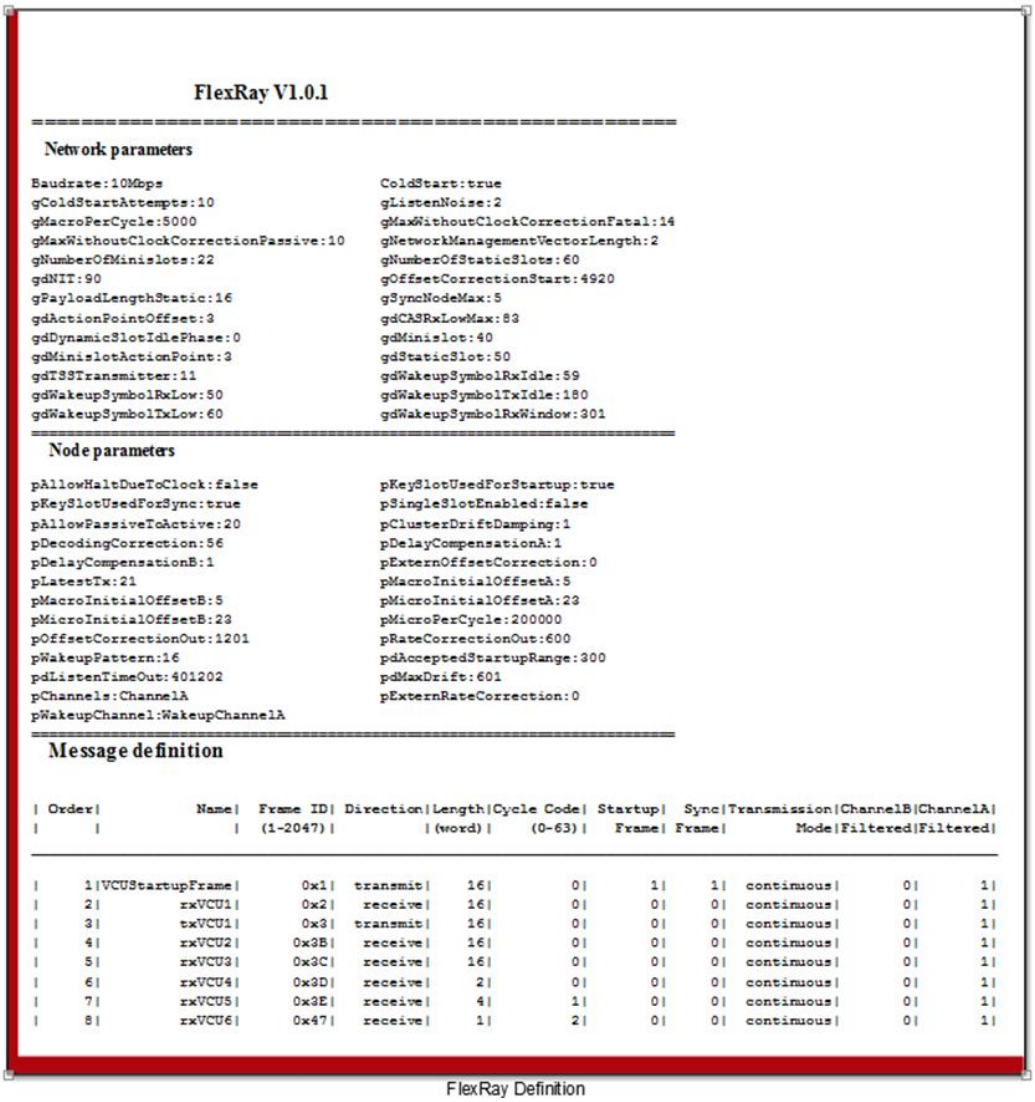
Parameter:

1) XCP Enable: Check to enable the XCP, uncheck to disable the XCP
2) CAN Channel: CAN channel selection
3) Master ID: address of the master node
4) Slave ID: address of the slave node

4.19 FlexRay Module

4.19.1 FlexRay Definition

This block is used for configuring FlexRay network. Startup frame and Sync Frame should be placed at the beginning when defining the messages.



FlexRay V1.0.1

Network parameters

Baudrate:10Mbps	ColdStart:true
gColdStartAttempts:10	gListenNoise:2
gMacroPerCycle:5000	gMaxWithoutClockCorrectionFatal:14
gMaxWithoutClockCorrectionPassive:10	gNetworkManagementVectorLength:2
gNumberOfMinislots:22	gNumberOfStaticSlots:60
gdNIT:90	gOffsetCorrectionStart:4920
gPayloadLengthStatic:16	gSyncNodeMax:5
gdActionPointOffset:3	gdCASRxLowMax:83
gdDynamicSlotIdlePhase:0	gdMinislot:40
gdMinislotActionPoint:3	gdStaticSlot:50
gdTSSTransmitter:11	gdWakeupSymbolRxIdle:59
gdWakeupSymbolRxLow:50	gdWakeupSymbolTxIdle:180
gdWakeupSymbolTxLow:60	gdWakeupSymbolRxWindow:301

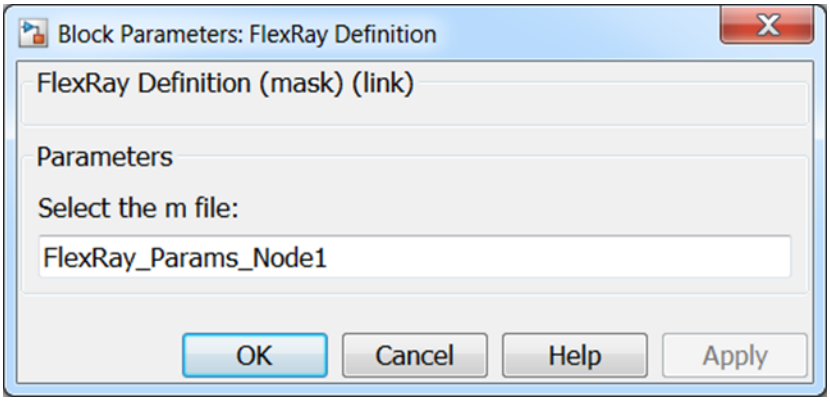
Node parameters

pAllowHaltDueToClock:false	pKeySlotUsedForStartup:true
pKeySlotUsedForSync:true	pSingleSlotEnabled:false
pAllowPassiveToActive:20	pClusterDriftDamping:1
pDecodingCorrection:56	pDelayCompensationA:1
pDelayCompensationB:1	pExternOffsetCorrection:0
pLatestTx:21	pMacroInitialOffsetA:5
pMacroInitialOffsetB:5	pMicroInitialOffsetA:23
pMicroInitialOffsetB:23	pMicroPerCycle:200000
pOffsetCorrectionOut:1201	pRateCorrectionOut:600
pWakeupPattern:16	pdAcceptedStartupRange:300
pdListenTimeOut:401202	pdMaxDrift:601
pChannels:ChannelA	pExternRateCorrection:0
pWakeupChannel:WakeupChannelA	

Message definition

Order	Name	Frame ID (1-2047)	Direction	Length (word)	Cycle Code (0-63)	Startup Frame	Sync Frame	Transmission Mode	ChannelB Filtered	ChannelA Filtered
1	VCUStartupFrame	0x1	transmit	16	0	1	1	continuous	0	1
2	rxVCU1	0x2	receive	16	0	0	0	continuous	0	1
3	txVCU1	0x3	transmit	16	0	0	0	continuous	0	1
4	rxVCU2	0x3B	receive	16	0	0	0	continuous	0	1
5	rxVCU3	0x3C	receive	16	0	0	0	continuous	0	1
6	rxVCU4	0x3D	receive	2	0	0	0	continuous	0	1
7	rxVCU5	0x3E	receive	4	1	0	0	continuous	0	1
8	rxVCU6	0x47	receive	1	2	0	0	continuous	0	1

FlexRay Definition



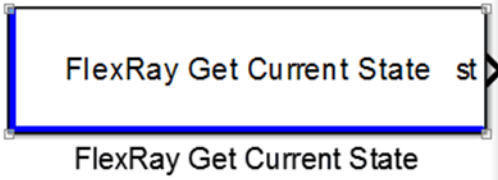
Parameter:

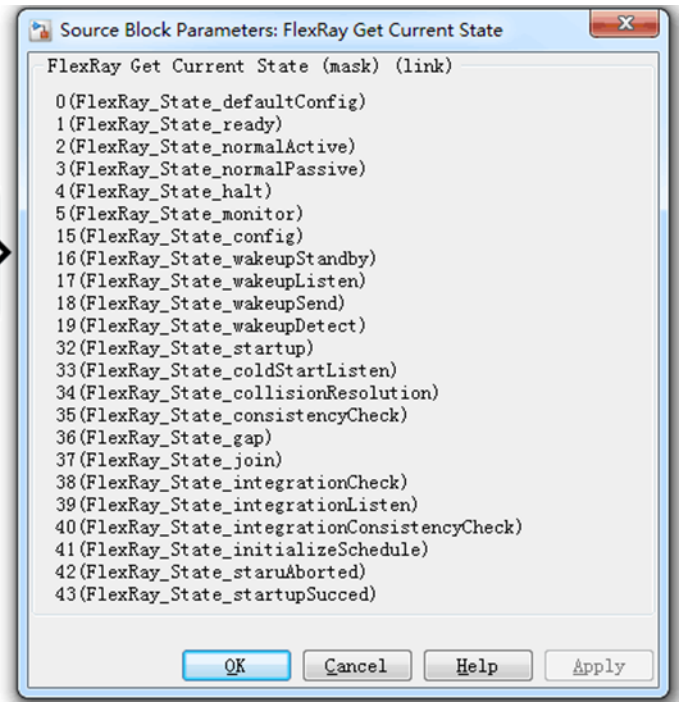
1) Select the m file: select the m file contains network parameters, node parameters, and

message definition. Please refer to the examples for how to construct a m file.

4.19.2 FlexRay Get Current State

This block is used for getting the status of the FlexRay network.



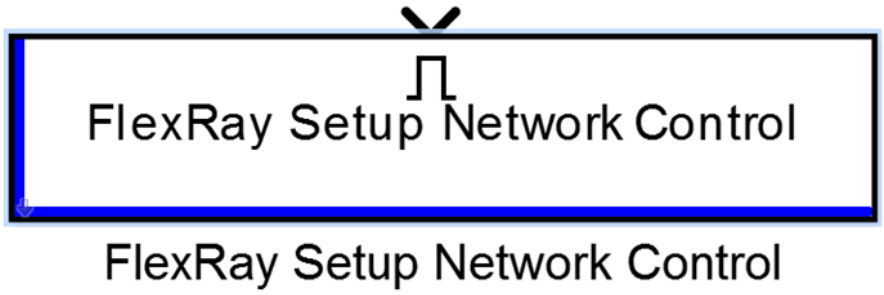


Output:

1) St: status of the network

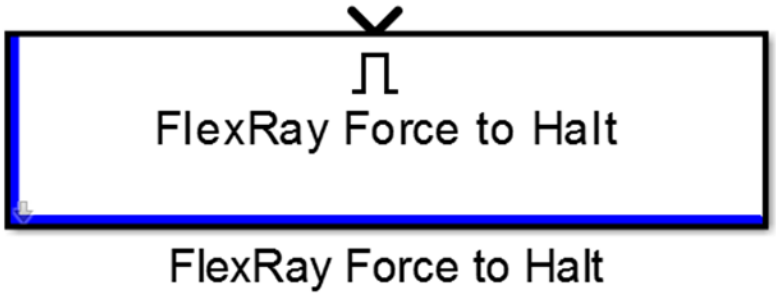
4.19.3 FlexRay Setup Network Control

This block is mandatory when building a network. It is recommended to used high priority 1ms task schedule for this block.



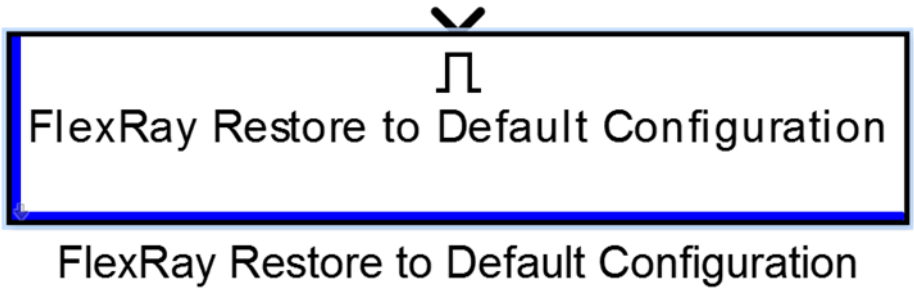
4.19.4 FlexRay Force to Halt

This block is used for force halt the FlexRay communication.



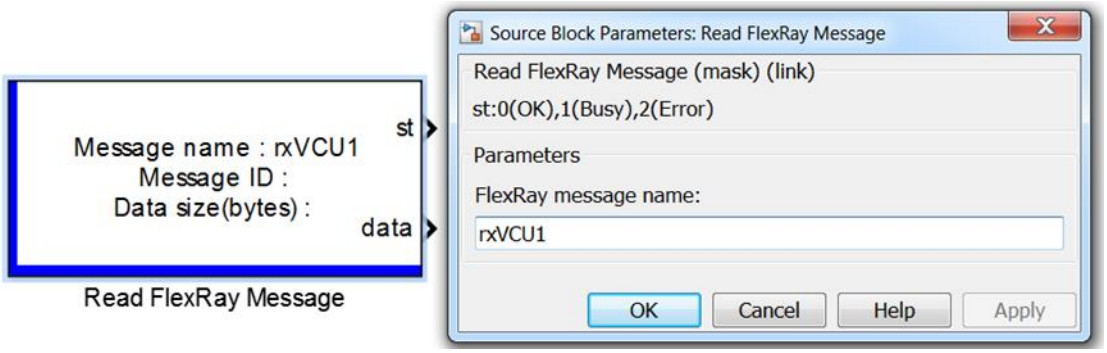
4.19.5 FlexRay Restore to Default Configuration

Call this block to restore the default configuration. It is useful when encounter network errors.



4.19.6 Read FlexRay Message

This block is used for reading messages from the network. Messages need to be predefined in FlexRay Definition block.



Parameter:

1) FlexRay message name: the name of the message.

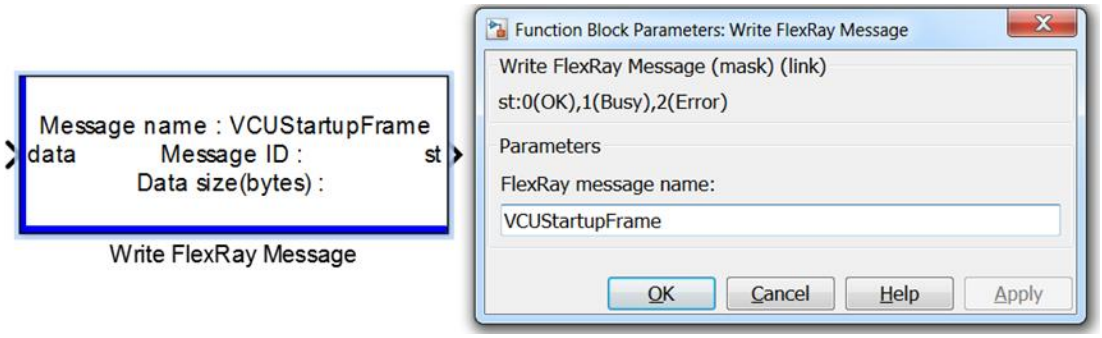
Output:

1) st: status flag, output 0 means message read successfully

5) data: data from the message

4.19.7 Write FlexRay Message

This block is used for sending message on FlexRay. The messages need to be predefined in the FlexRay Definition block.



Parameter:

1) FlexRay message name: the name of the message

Input:

1) data: the data in the message

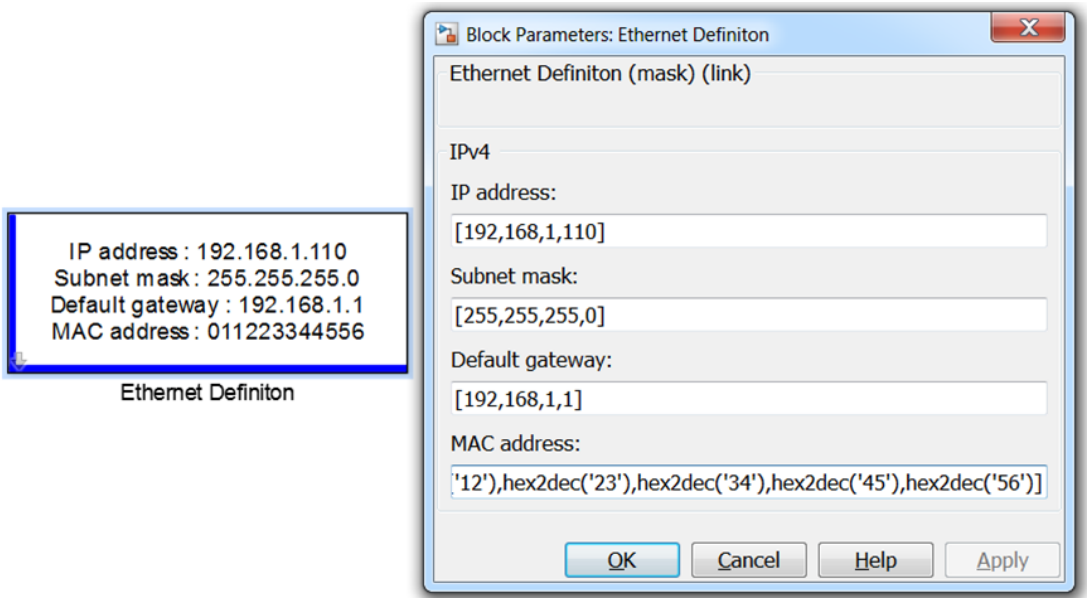
Output:

1) st: status of writing message. Outputs 0 means the message has been written successfully.

4.20 Ethernet

4.20.1 Ethernet Definition

This block is used for configuring the Ethernet.

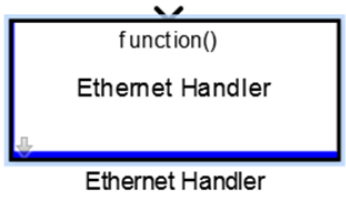


Parameter:

1) IP address
2) Subnet mask
3) Default gateway
4) MAC address

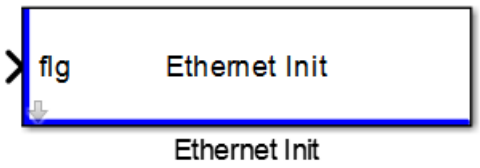
4.20.2 Ethernet Handler

Ethernet upper-level protocol will only work when this block is called regularly.



4.20.3 Ethernet Init

Call this block after PHY chip is powered on to initialize Ethernet. Please note, this block can only be called once after every power cycle.



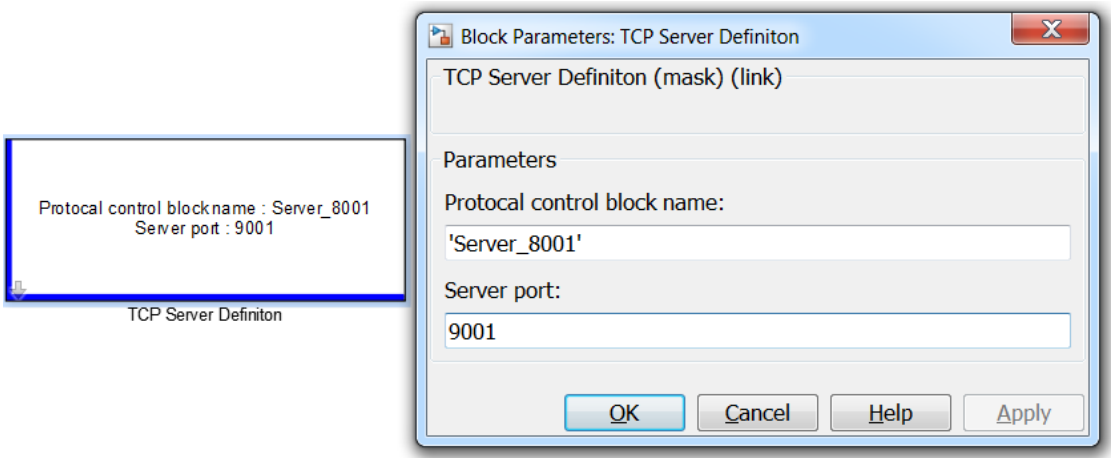
Input:

1) Flg: input 1 to initialize Ethernet.

4.21 TCP Protocol Blocks

4.21.1 TCP Server Definition

This module is used to define the TCP server. A server port currently only supports one client connection. The new client connection will replace the previous connection.

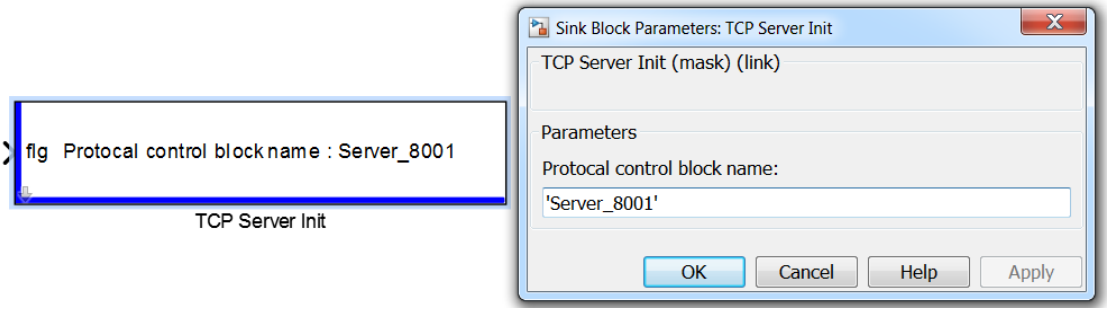


Parameter:

1) Protocol control block name: Name of the block
2) Server port

4.21.2 TCP Server Init

This block is used to initialize the TCP server after the Ethernet is initialized. It is recommended to run this block no more than once every power cycle.



Parameter:

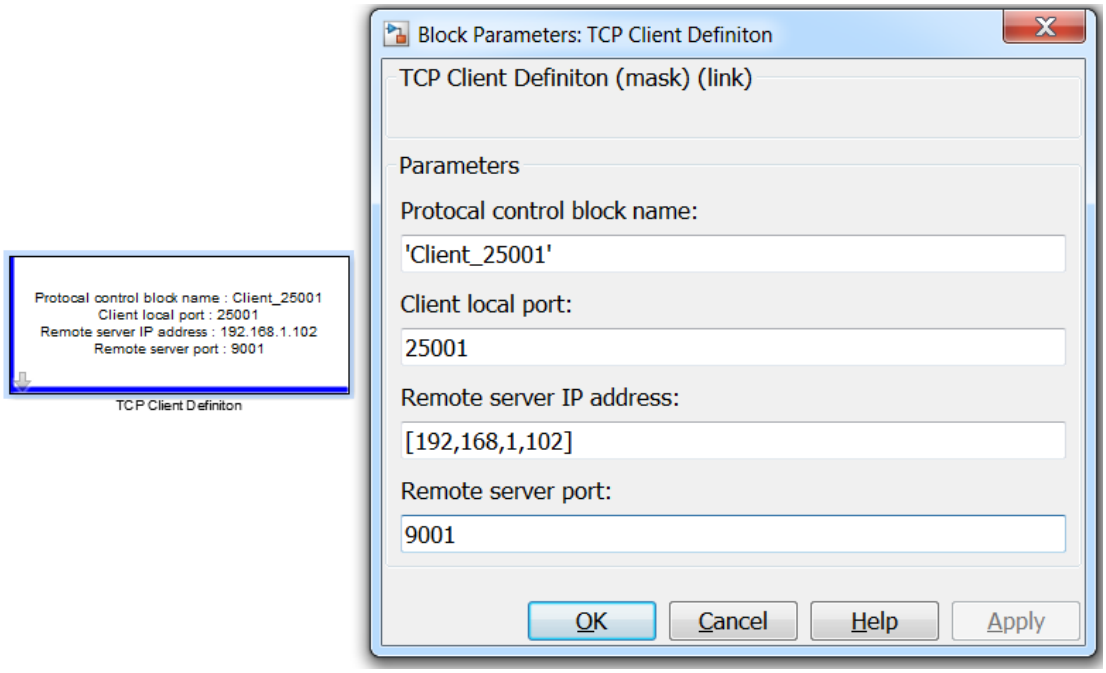
1) Protocol control block name: Name of the control block.

Input:

1) Flg: Proceed initialization when flg is set to 1

4.21.3 TCP Client Definition

This block is used to define a TCP client.

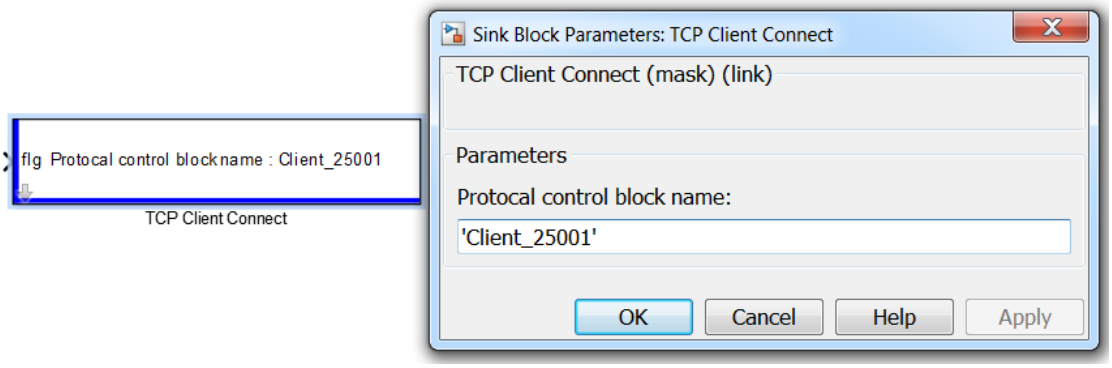


Parameter:

1) Protocol control block name: name of the protocol control block
2) Client local port
3) Remote server IP address
4) Remote server port

4.21.4 TCP Client Connect

Use this block to connect to a server after Ethernet Init block is used.



Parameter:

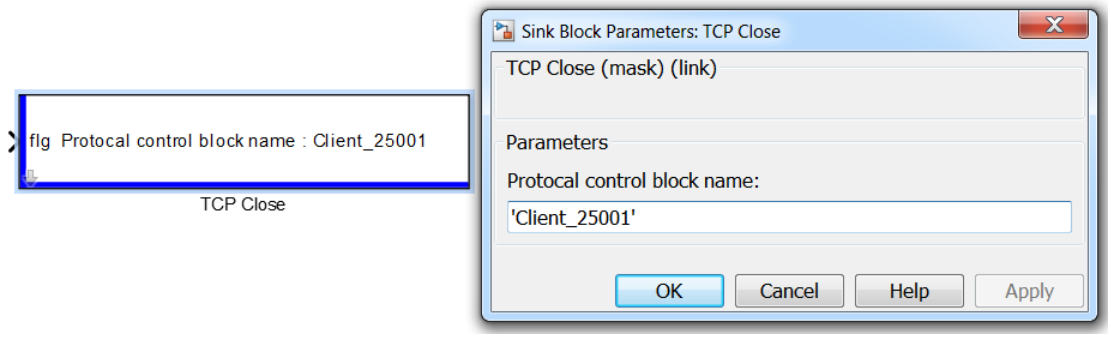
1) Protocol control block name: name of the control block

Input:

1) Flg: set flg to 1 to start the connection.

4.21.5 TCP Close

This block closes the current connection.



Parameter:

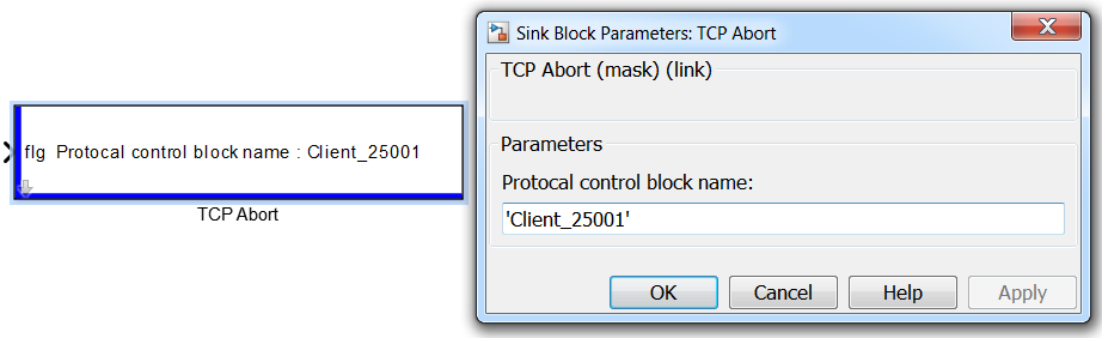
1) Protocol control block name: name of the control block

Input:

1) Flg: set flg to 1 to close the connection.

4.21.6 TCP Abort

This block aborts the current connection.



Parameter:

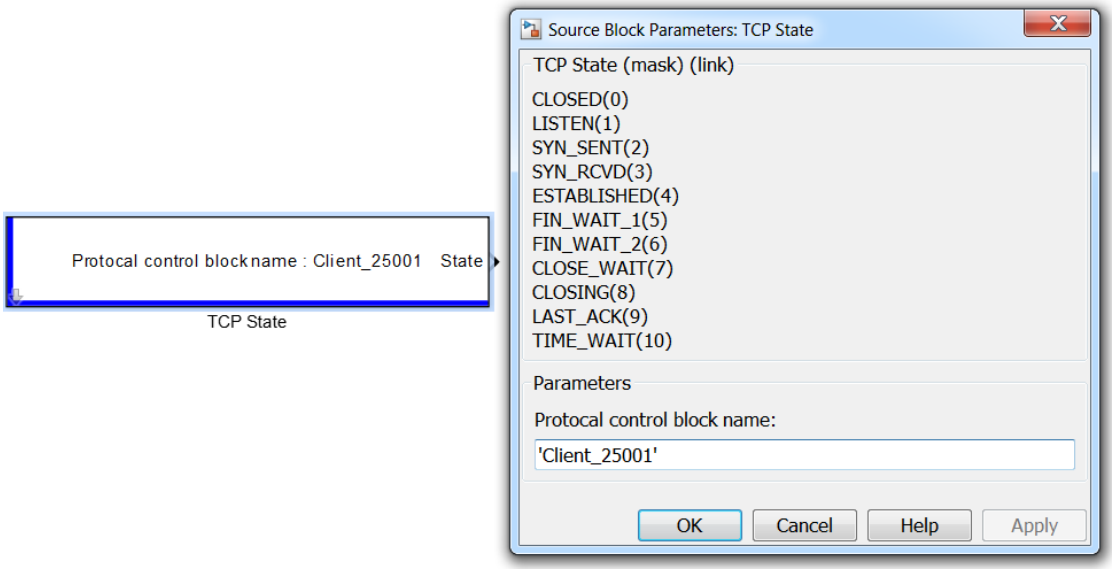
1) Protocol control block name: name of the control block

Input:

2) Flg: set flg to 1 to abort the connection.

4.21.7 TCP State

Use this block to request the state of the current connection.



Parameter:

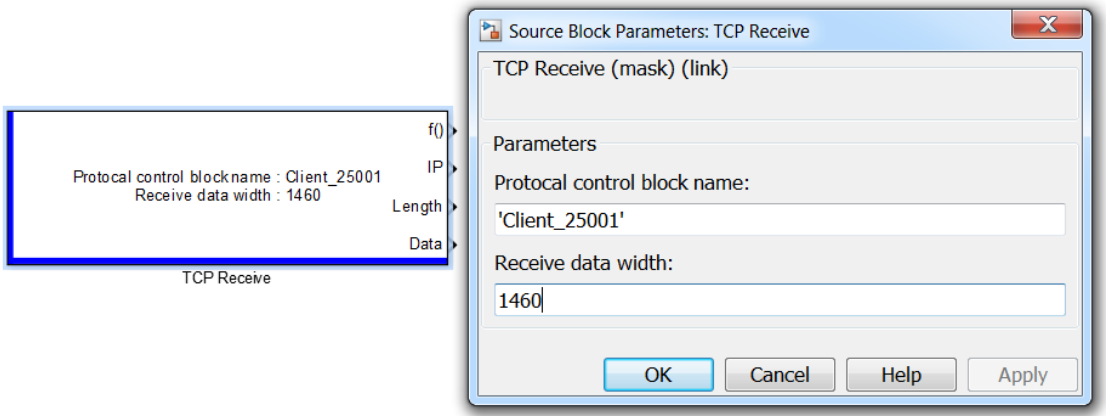
1) Protocol control block name: name of the control block

Output:

1) State: current state of the connection.

4.21.8 TCP Receive

This block is used for receiving TCP data.



Parameter:

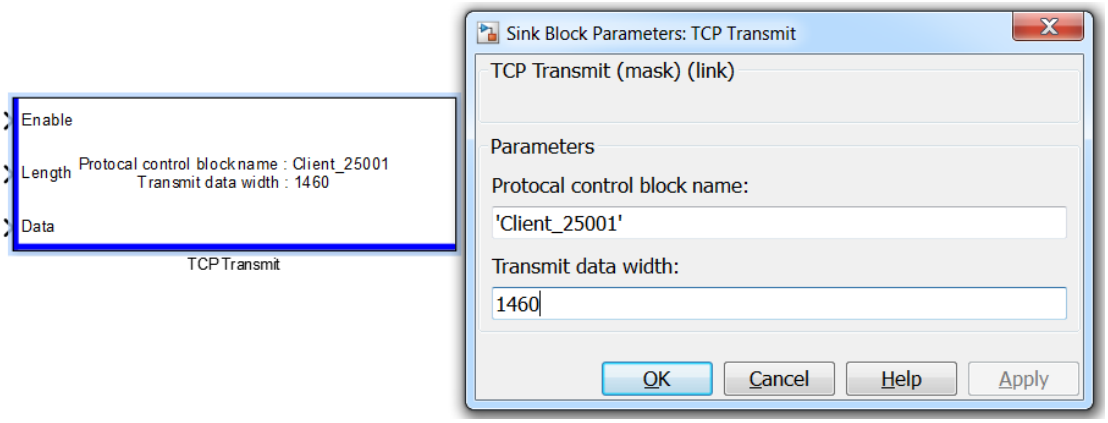
1) Protocol control block name: name of the control block
2) Receive data width: the width of the Data port

Output:

1) f(): the function to run when data is received
2) IP: IP address of the sender
3) Length: actual received data length
4) Data: received data

4.21.9 TCP Transmit

This block is used for transmitting TCP data.



Parameter:

1) Protocol control block name: name of the control block
2) Transmit data width: the width of the Data

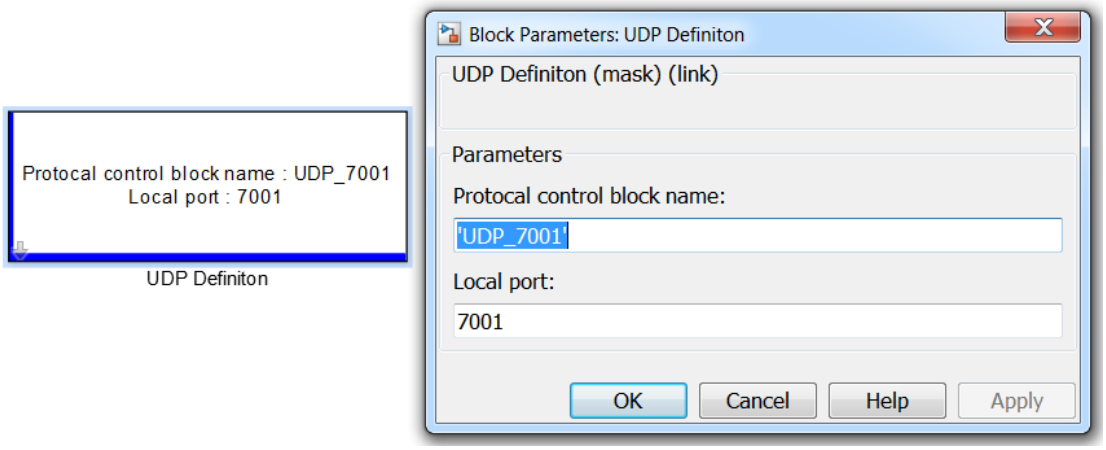
Input:

1) Enable: set it to 1 to enable the transmit, set it to 0 disable the transmit
2) Length: the length of the data to be transmit
3) Data: the Data to be transmitted

4.22 UDP Blocks

4.22.1 UDP Definition

This block is used for defining UDP ports.



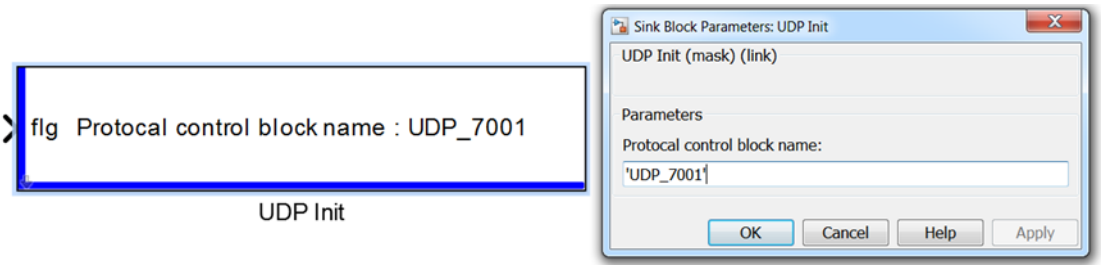
Parameter:

1) Protocol control block name: the name of the protocol control block

5) Local Port: local port configuration

4.22.2 UDP Init

This block is used for initiating the UDP ports. User should use this block after the "Ethernet Init" block is called. It is recommended to call "UDP Init" block once every power cycle.



Parameter:

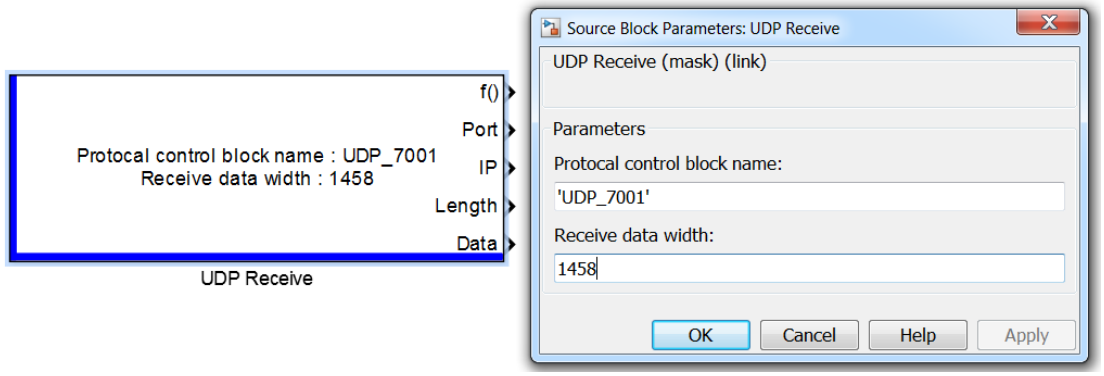
1) Protocol control block name: the name of the protocol control block

Input:

1) Flg: input 1 to enable the initialization.

4.22.3 UDP Receive

This block is used for receiving UDP data.



Parameter:

1) Protocol control block name: the name of the protocol control block

6) Receive data width: the data width of the output port

Output:

1) f(): the trigger function to be run when any data received

7) Port: sender's data port

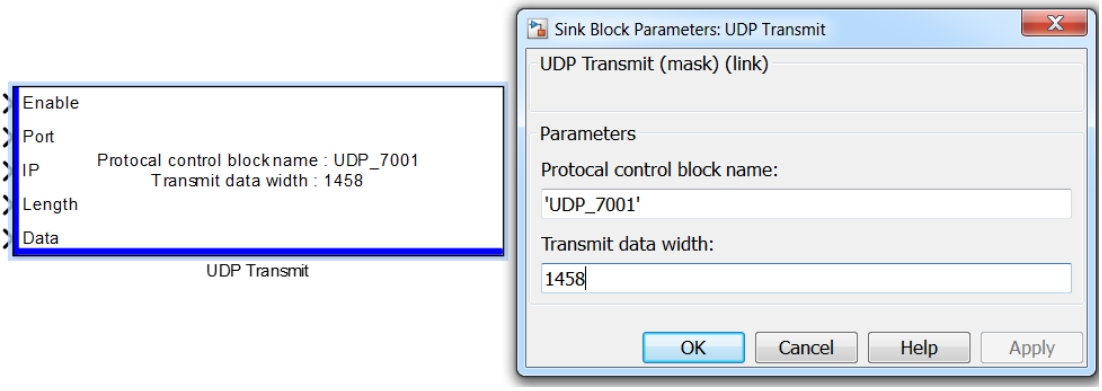
8) IP: IP address of the sender

9) Length: the actual length of the received data

10) Data: received data

4.22.4 UDP Transmit

This block is used for transmitting UDP data.



Parameter:

1) Protocol control block name: the name of the protocol control block.

11) Transmit data width: the data width of the data input port

Input:

1) Enable: set to 1 to enable the transmission, set to 0 to disable the transmission

12) Port: the data port of the receiver

13) IP: the target IP

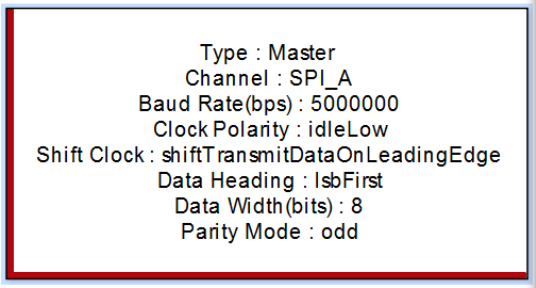
14) Length: the length of the data to be transmitted

15) Data: the data to be transmitted

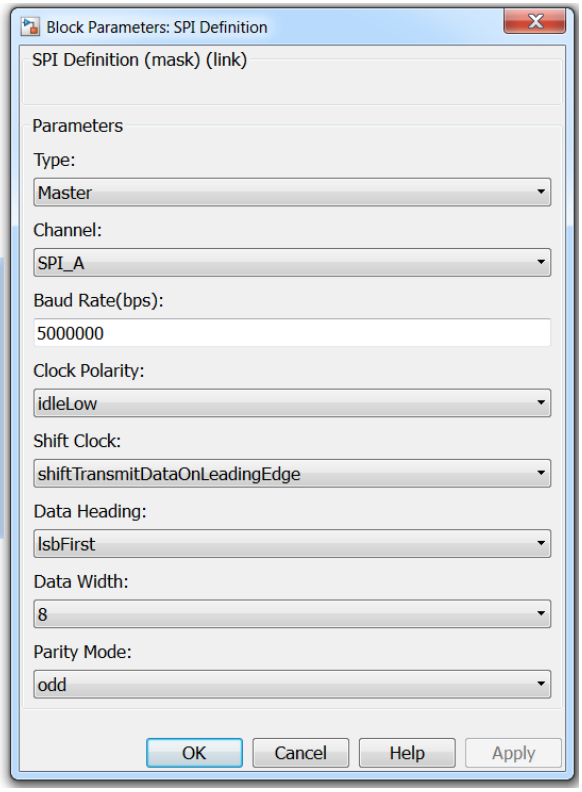
4.23 SPI Blocks

4.23.1 SPI Definition

Use this block to configure the SPI driver.



SPI Definition

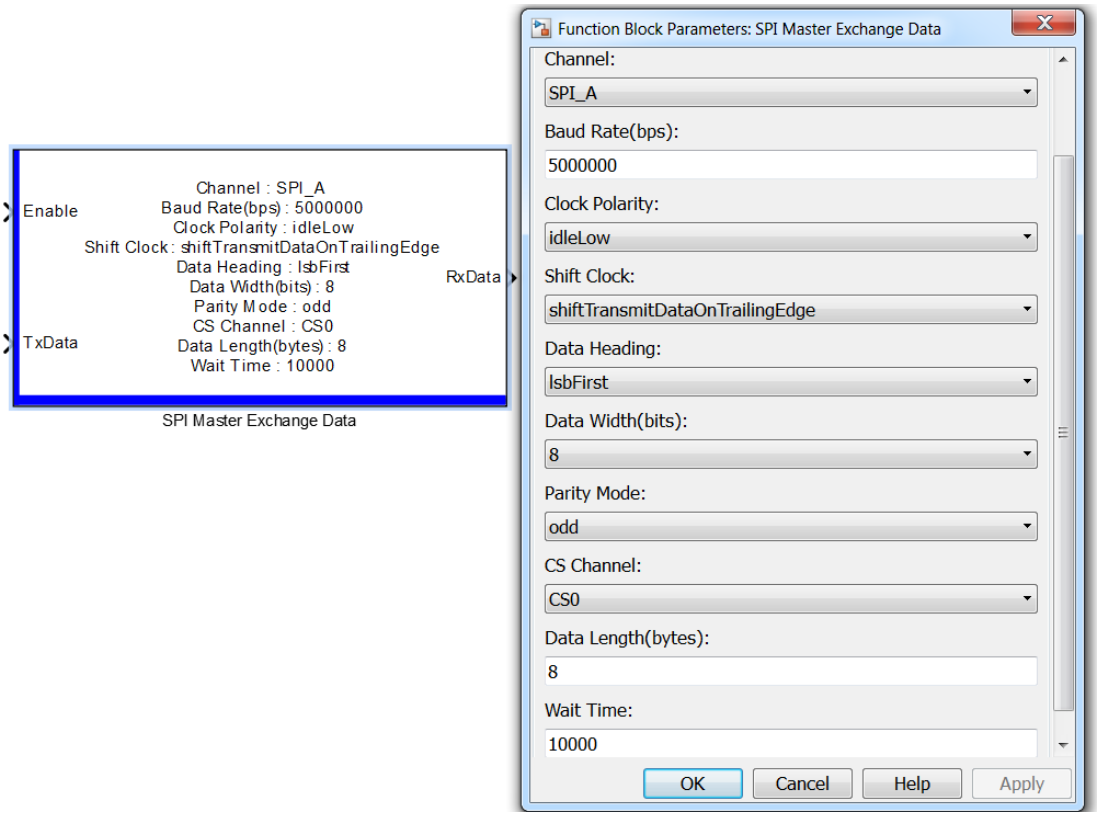


Parameter:

1) Type: Master/Slave mode selection
2) Channel: channel selection
3) Baud Rate (bps): baud rate setting
4) Clock Polarity: clock polarity selection
5) Shift Clock: Clock shift setting
6) Data Heading: Endianness
7) Data Width: the width of the data
8) Parity Mode: odd/even parity mode

4.23.2 SPI Master Exchange Data

Use this block to exchange data when the “SPI Definition” block is configured as master node.



Parameter:

1) Channel: channel selection
2) Baud Rate (bps): baud rate setting
3) Clock Polarity: clock polarity selection
4) Shift Clock: Clock shift setting
5) Data Heading: Endianness
6) Data Width: the width of the data
7) Parity Mode: odd/even parity mode
8) CS Channel: CS Channel selection
9) Data Length(bytes): the length of the input port (TxData) and the output port (RxData).

10) Wait Time: the time to wait

Input:

1) Enable: set to 1 to enable the transmission, set to 0 to disable the transmission

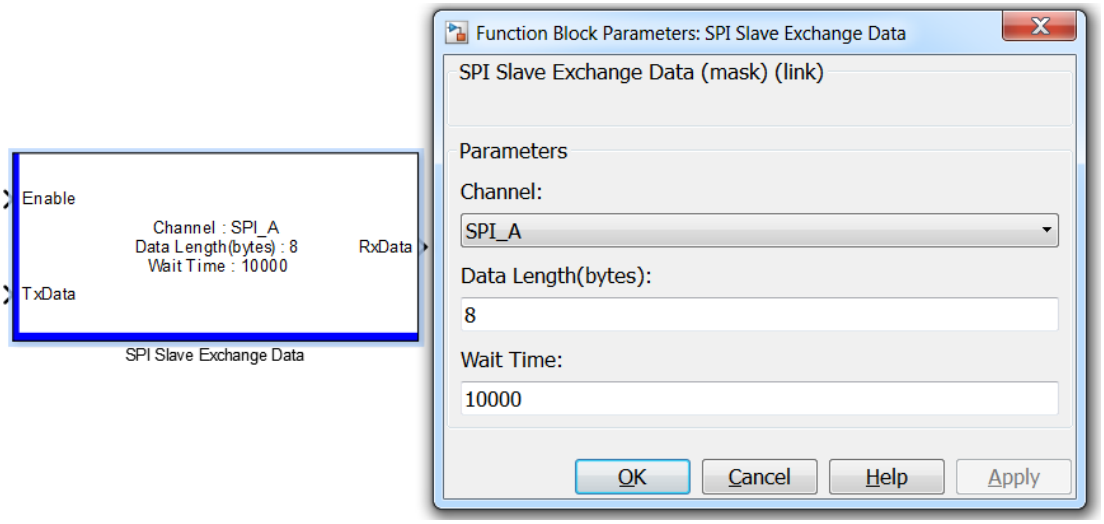
16) TxData: the data to be transmitted

Output:

1) RxData: the data to be received

4.23.3 SPI Slave Exchange Data

Use this block to exchange data when the “SPI Definition” block is configured as slave node.



Parameter:

1) Channel: channel selection

2) Data Length(bytes): the length of the input port (TxData) and the output port (RxData).

3) Wait Time: the time to wait

Input:

1) Enable: set to 1 to enable the transmission, set to 0 to disable the transmission

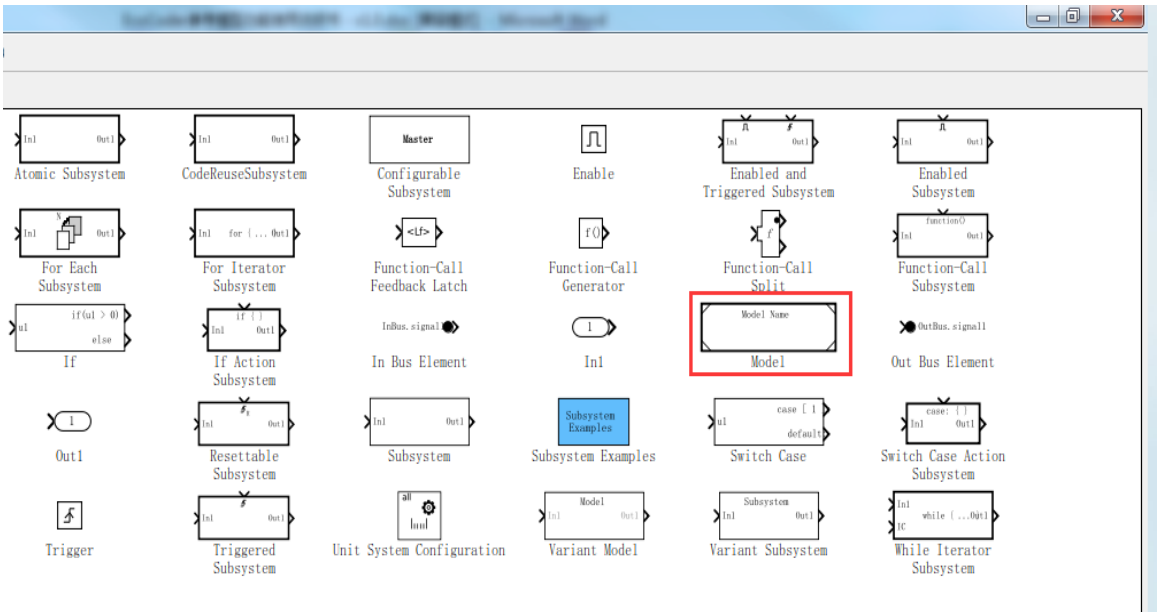
2) TxData: the data to be transmitted

Output:

1) RxData: the data to be received

4.24 Model Reference

Model reference in Matlab Simulink:

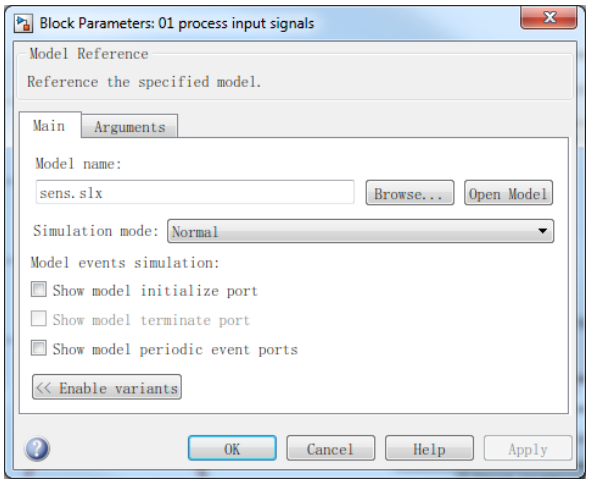


The 'Model' block in Simulink library 'Ports & Subsystems' could be used to include a sub-model in a parent model. The blocks included in the 'Model' block are regarded as referenced models, and the model that includes the referenced models is named Parent Model. The referenced model can be used as an independent model to run simulation independently; it can also be used as the model reference and take part in the simulation in the parent model.

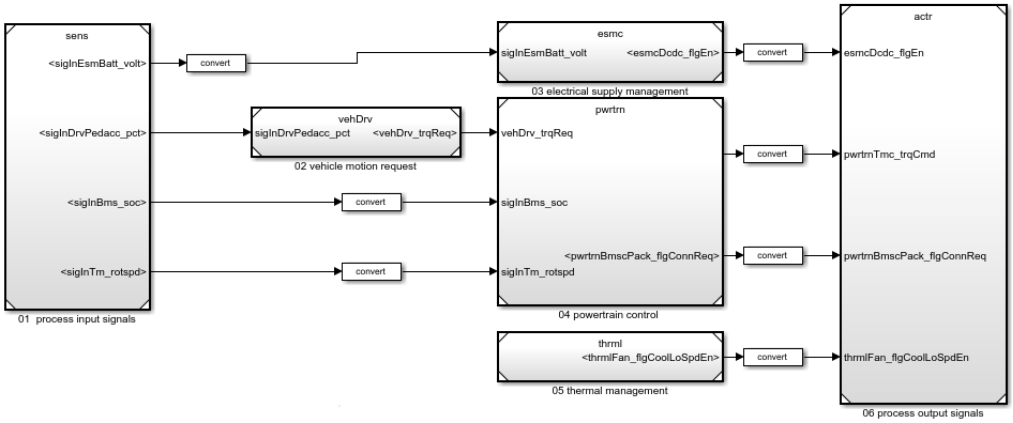
For more information about the Model Reference, please refer to:

https://www.mathworks.com/help/simulink/model-reference.html

Add the Simulink block 'Model' to the current Simulink model, and double click, the user will be able to add the referenced model in the popup window Block Parameters.



After the referenced model is successfully added, the inputs and outputs of the referenced model will appear on both sides of the 'Model' block. The user can connect the inputs and outputs on the 'Model' block to the parent model.



4.24.1 Configurations for Parent Models and Referenced Models

When building Simulink models with EcoCoder, the configurations of the referenced model(s) and the parent model must be kept the same, otherwise there will be errors during the code generation or simulation and the ongoing process will stop due to the error. There are two methods to keep the same configuration for referenced model(s) and parent model:

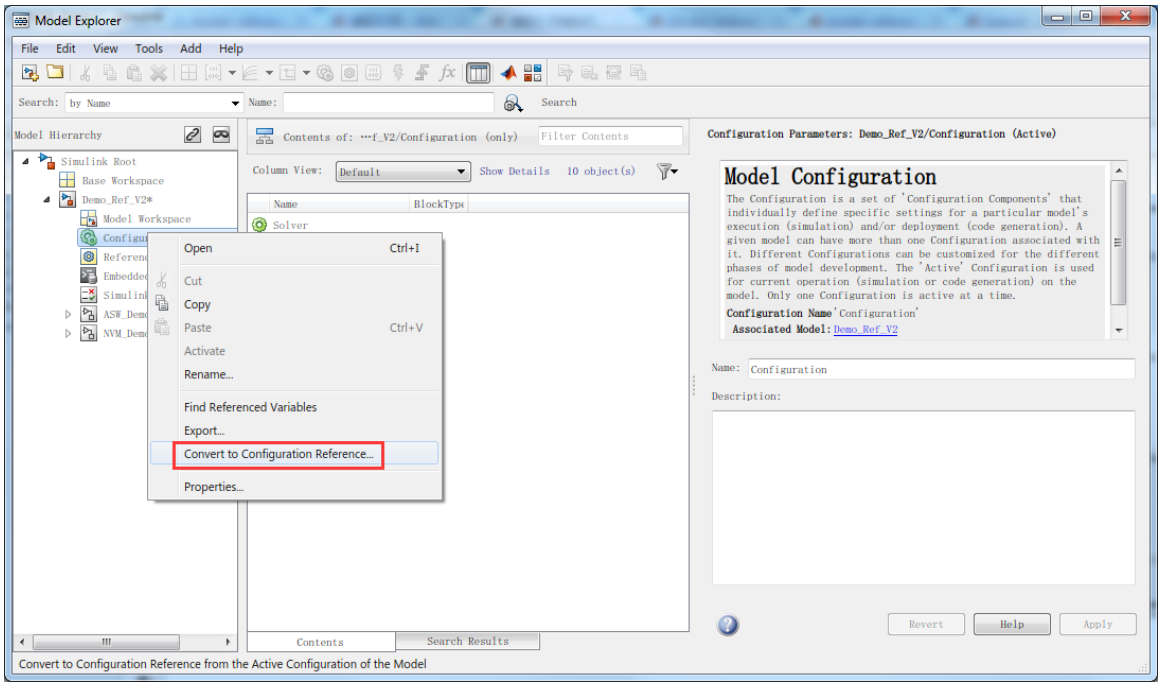
1. Use 'Configuration Reference'

   Or

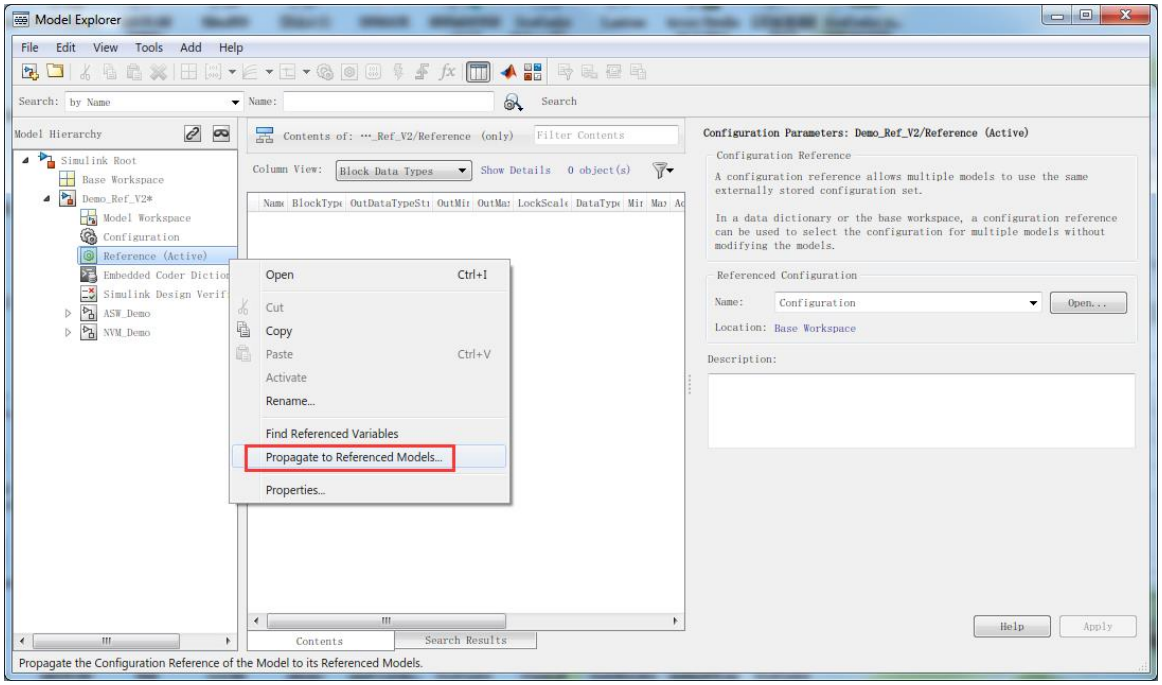
2. Copy the configuration of the parent model to the referenced model(s)

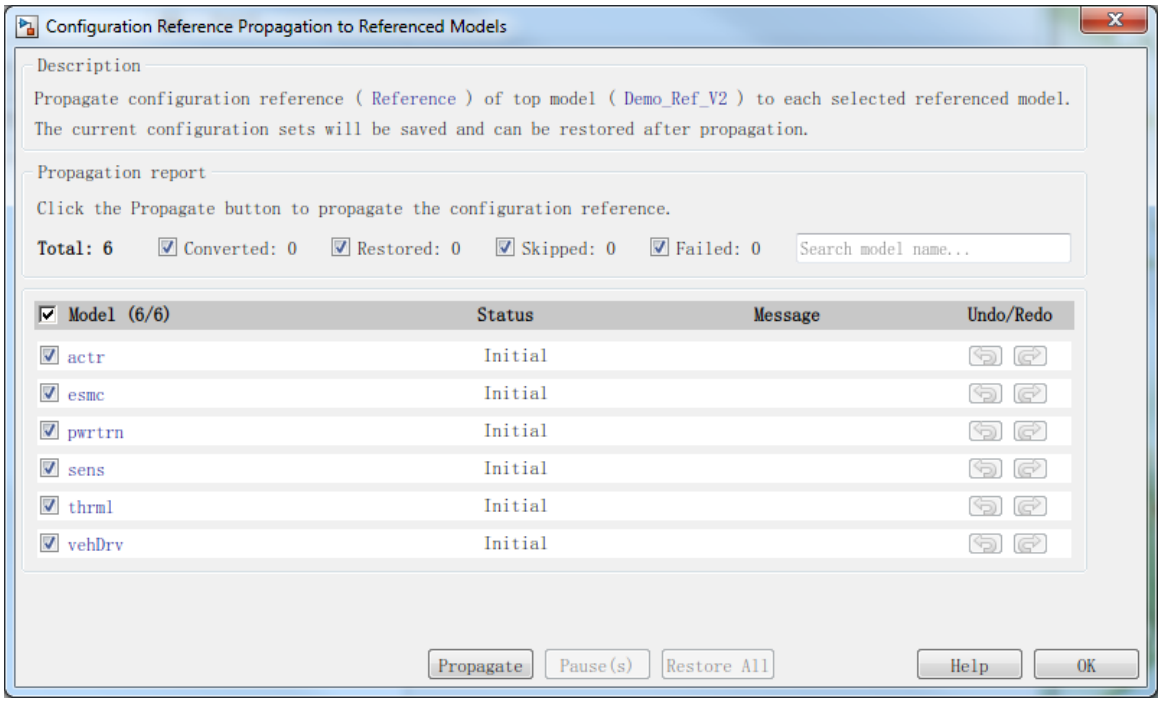
4.24.2 Configuration Reference

In order to keep the same configuration for parent model and referenced model(s), the user can use 'Configuration Reference' function. In Model Explorer, by clicking 'Convert Configuration Reference', the configuration of the parent model will be converted to configuration reference.



Then right click 'Reference', select 'Propagates to Referenced Models' in the menu to apply the configuration in the referenced model(s).





After the conversion of the configuration file, please save the configuration file in the Matlab Workspace, the default name is ‘configuration’.

Please remember to load the saved configuration file to Matlab Workspace before opening the parent model.

For more information about Configuration Reference, please refer to:

https://www.mathworks.com/help/simulink/ug/more-about-configuration-references.html?#responsive_offcanvas

4.24.3 Copy Parent Model Configuration File to Referenced Model

In Model Explorer, the configuration file for different models could by copied by right clicking the file. In this case, the user can also copy the configuration file of the parent model to the referenced model(s) and activate the configuration file in the referenced model by right clicking the configuration.

4.24.4 EcoCoder Blocks in Model Reference

All blocks in the EcoCoder have been updated to be compatible with Model Reference. But in order to keep the generated code with good normative and readability, users need to pay special attention to some of the EcoCoder blocks. Specifically, all EcoCoder blocks with blue edges could be used in both parent model and referenced model(s); while the EcoCoder blocks with red edges will only be valid in the parent model (*They can be used in the referenced models to pass the simulation, but will not be generated in the C code*)

ADC

All the blocks in ADC can be used and will be valid in both parent model and referenced model(s).

Advanced Data Blocks

All the blocks in Advanced Data Blocks can be used in both parent model and referenced model(s).

However, for the EEPROM Emulation Definition block, only the definition in parent model is valid, and will be generated into C code. If this block is defined in the referenced model(s), the definition will not be generated into C code.

Application Base Blocks

All the blocks in Application Base blocks can be used and will be valid in both parent model and referenced model(s).

Calibration & Measurement

In Calibration & Measurement, the Calibration Data Check can only be used in in the parent model, for all other blocks in Calibration & Measurement, they can be used in both parent model and referenced model(s). If these blocks are defined in the referenced model(s), the definitions will not be generated into C code.

CAN

All the blocks in CAN can be used in both parent model and referenced model(s).

The CAN channel Definition block and CAN Wake-up Frame Definition block are valid and will be generated into C code only when they are defined in the parent model.

CCP

In CCP, all blocks can only be used in the parent model and shall not be used in the referenced model(s).

Diagnostic Blocks

All blocks in Diagnostic Block can be used and will be valid in both parent model and referenced model(s).

Digital I/O

All the blocks in Digital I/O can be used in both parent model and referenced model(s).

However, the H-Bridge Definition block, IPWM interrupt Handler Definition block, PWM Definition block and PWM IO Frequency Range Definition blocks are valid and will be generated into C code only when they are defined in the parent model. If these blocks are defined in the referenced model(s), they will not be generated into C code.

Non-Volatile Memory Blocks

In the Non-Volatile Memory Blocks, NVM Definition block can only be used in the parent model and shall not be used in the referenced model(s).

And Fixed NVM Definition block is valid and will be generated into C code only when it is defined in the parent model, if Fixed NVM Definition block is defined in the referenced model(s), it will not be generated into C code (Unlike Non-Volatile Memory block, user can still keep the Fixed NVM Definition block in the referenced model(s)). The parameters of the Fixed NVM Definition in all models shall be kept the same.

All other blocks in Non-Volatile Memory blocks can be used in both parent model and referenced model(s).

Programming Blocks

All blocks in the Programming Blocks can only be used in the parent model.

SCI

All the blocks in SCI can be used in both parent model and referenced model(s).

However, the SCI Definition block is valid and will be generated into C code only when they are defined in the parent model. If this block is defined in the referenced model(s), it will not be generated into C code.

System Management Blocks

All the blocks in System Management Blocks can be used in both parent model and referenced model(s).

However, the ECU Master Chip Wake-up Definition block, Stack Overflow Detection Definition block and Watchdog Definition block are valid and will be generated into C code only when they are defined in the parent model. If these blocks are defined in the referenced model(s), they will not be generated into C code.

Task Scheduler

All the blocks in Task Scheduler can be used and will be valid in both parent model and referenced model(s).

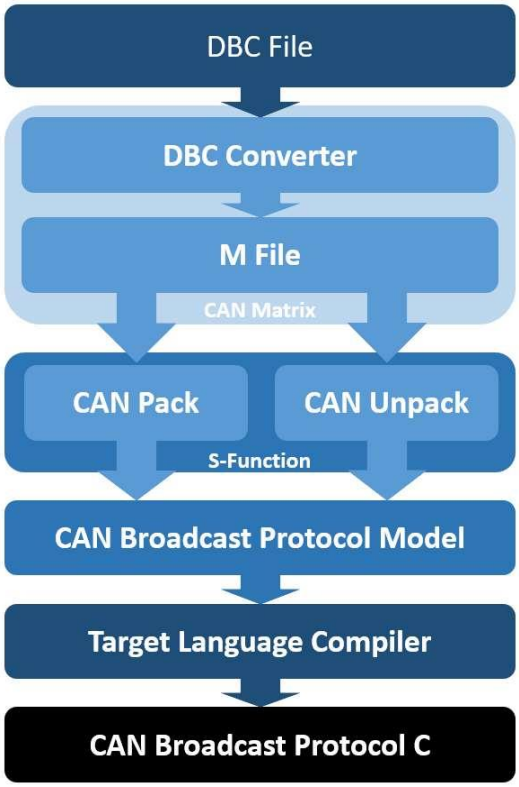
Chapter 5 CAN Theory of Ecotron

5.1 Introduction

Controller Area Network (CAN) nowadays is very widely used on the vehicle control system. Ecotron VCUs provide multiple CAN channels (3-5 channels, depending on the specific VCU model) and enables the VCUs to communicate with multiple electronic control units on the vehicle.

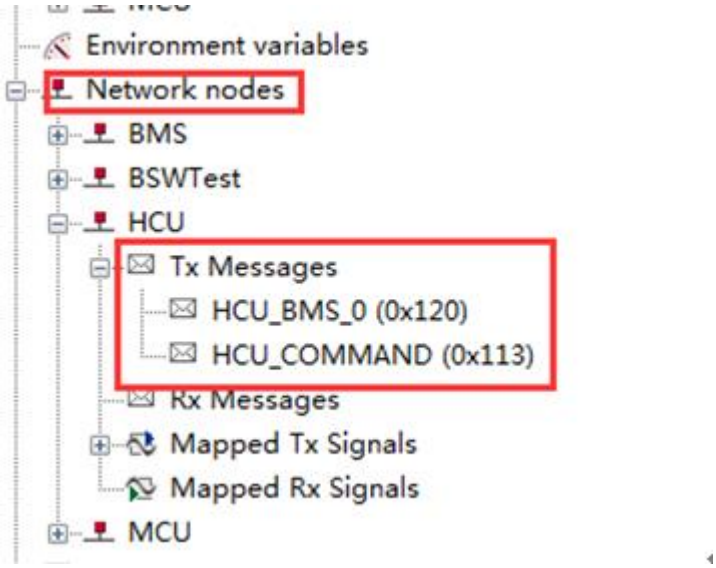
5.2 CAN Implementation

To use Ecotron VCUs, the user needs to convert DBC file into .m file and then use the .m file to define and initialize the CAN communications. The process is intuitive, user-friendly, and can give users more flexibility for CAN communication implementation.



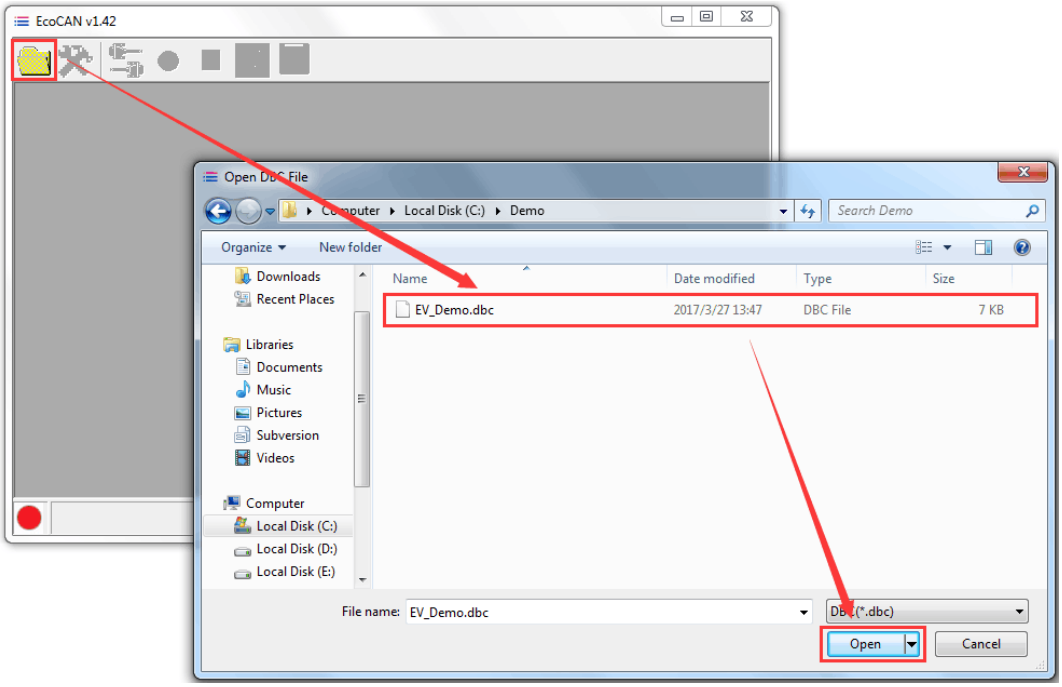
5.2.1 Convert DBC to m File

User can convert DBC to .m file automatically using the software EcoCAN that can be found in EcoCAL. If you want to know more about EcoCAL, please refer to the manual *EcoCAL manual for EV*.

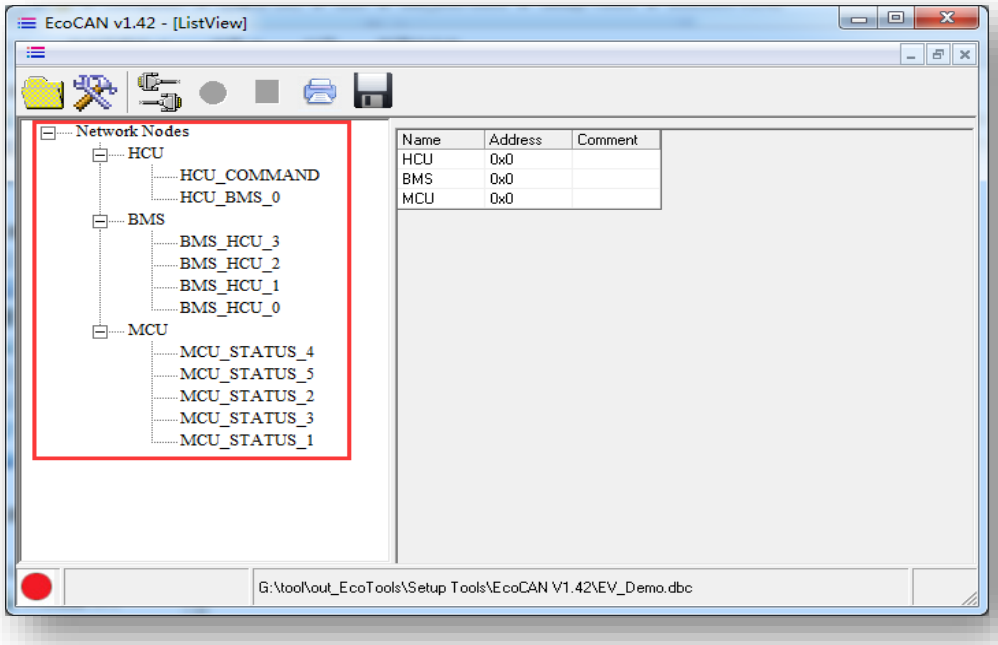


Process:

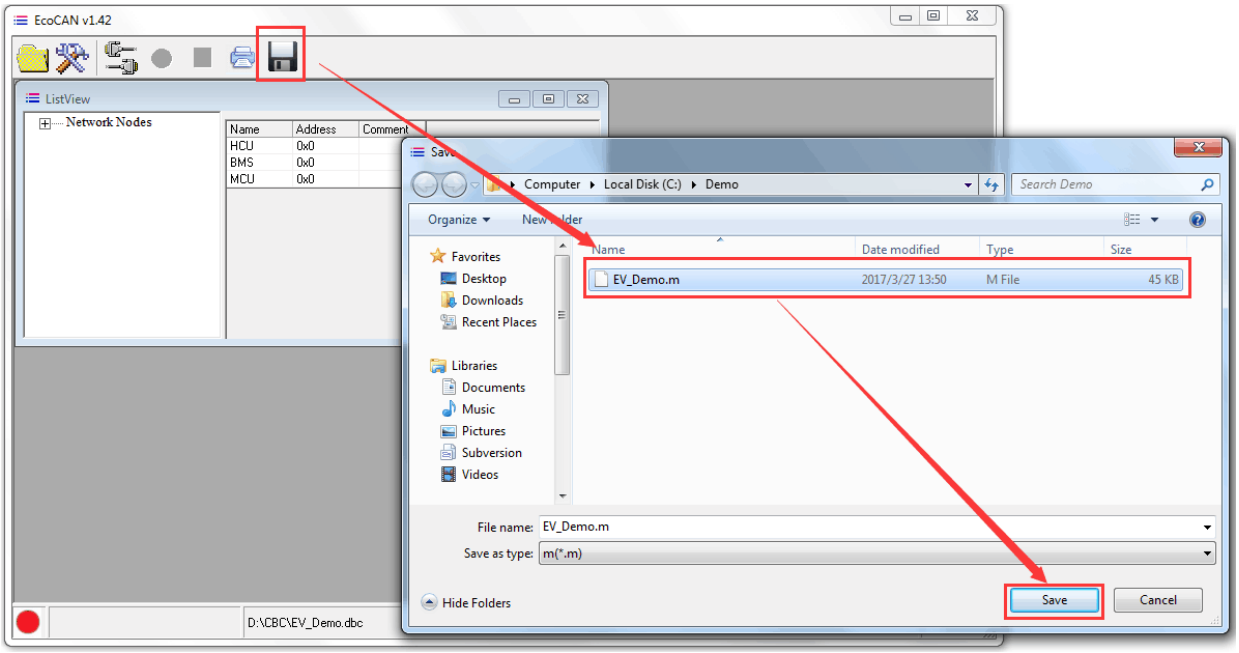
1. Open the DBC file to be converted in EcoCAN.



2. After DBC file being loaded, the following window will pop-up.

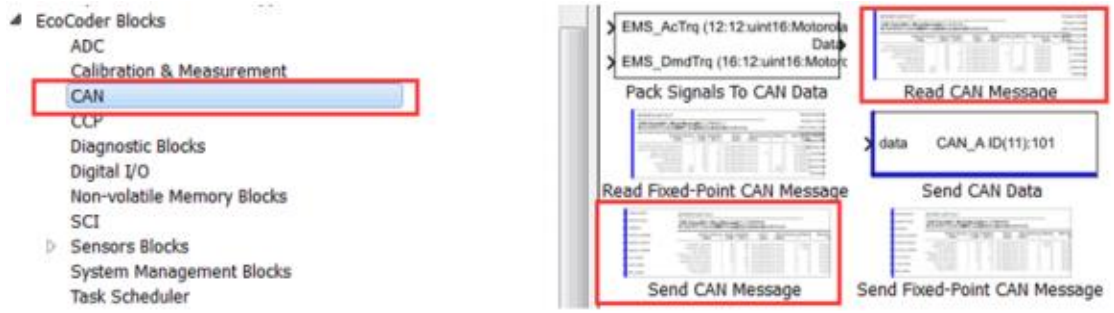


3. Click the indicated button and export the DBC file to m file, users can specify the saving path.



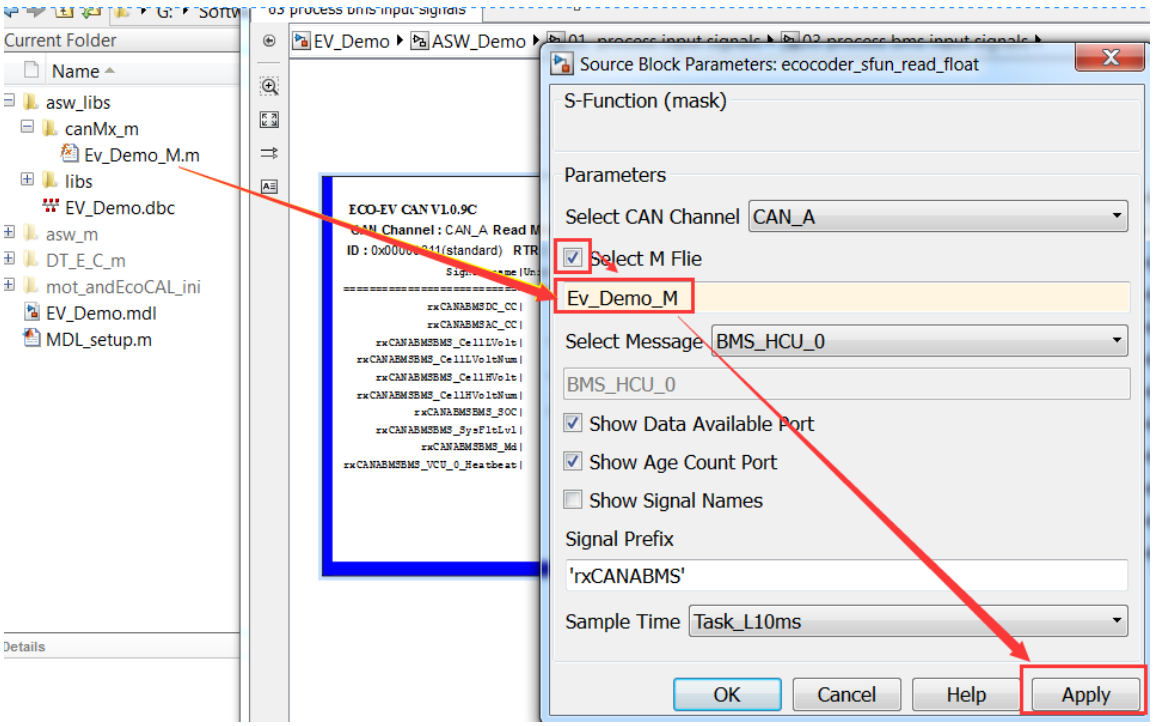
5.2.2 EcoCoder CAN Blocks

Please select 'Read CAN Message' or 'Send CAN Message' if fixed-point tool has not been installed in Matlab.



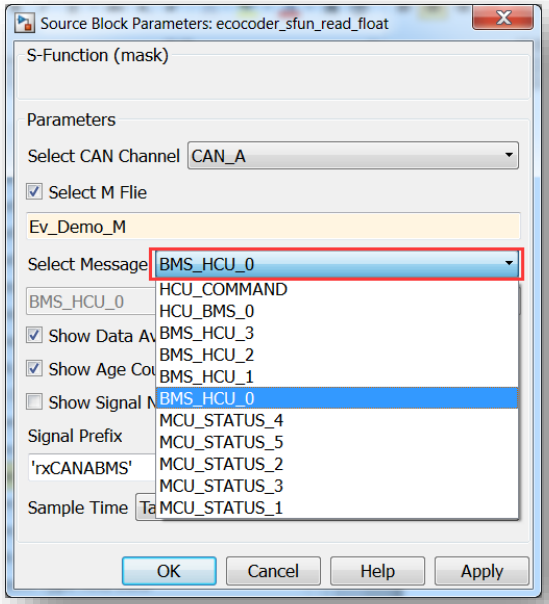
5.2.3 Select m file

This m file can help parse out signals in messages. Users need to save the .m file in the folder where your model is.

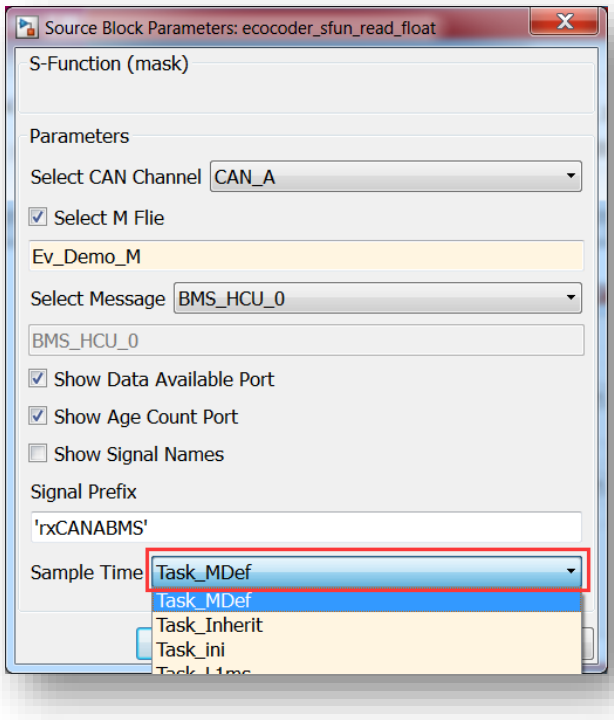


5.2.4 Select Message

This step will let users pick the specific CAN messages that need to be parsed.



5.2.5 Select Sample Time



Task_Inherit:

If 'Task_Inherit' is selected, the block will be executed every time when the subsystem that includes this block is executed.

Task_ini:

The block will only be executed during the initialization process when VCU is powered on.

Task_MDef:

The sample time will be decided according to the interval value in the .m file that is shown below. (This value comes from DBC file and is editable).

```
%Message Number:1
case 'HCU_COMMAND'
    ECOCAN.HCU_COMMAND = struct;
    ECOCAN.HCU_COMMAND .name = 'HCU_COMMAND';
    ECOCAN.HCU_COMMAND.description = 'HCU_COMMAND';
    ECOCAN.HCU_COMMAND.protocol  = 'ECOCAN';
    ECOCAN.HCU_COMMAND.id = hex2dec('113');
    ECOCAN.HCU_COMMAND.idext = 'SIANDARD';
    ECOCAN.HCU_COMMAND.payload_size =8;
    ECOCAN.HCU_COMMAND.interval =-1;
```

Interval	**(-t，-1)**	**(0,0.005)**	**[0.005,0.01)**	**[0.01,0.02)**	**[0.02,0.05)**
Sample Time	Task_Inherit	Task_H1ms	Task_H5ms	Task_H10ms	Task_L20ms
Interval	[0.05,0.1)	[0.1,0.2)	[0.2,0.5)	[0.5,1)	[1,10)
Sample Time	Task_L50ms	Task_L100ms	Task_L200ms	Task_L500ms	Task_L1000ms

Chapter 6 Memory Management

6.1 Introduction

When application software gets more complicated and larger, memory management will become an important aspect of VCU software development.

6.2 Storage device

Ecotron VCU includes two types of storage device, Flash and RAM.

Flash is the memory which stores basic software, application software, constant, calibration and NVM variable data, the data in Flash will not be lost after powering off the VCU. Contents in Flash would be copied to RAM during VCU power-up process. NVM variable data is recommended to be saved to Flash before VCU power off. Calibration can be implemented 'on the fly', and calibration variable data can be burned back to Flash manually through EcoCAL, the calibration software developed by Ecotron.

RAM (Random Access Memory) directly works with CPU by storing software needed data and code during VCU runtime. Different from Flash, the data in RAM would be lost when VCU powers down.

6.3 Data Storage

6.3.1 Calibration/Measurement Variable

Please refer to section 4.10.1 to 4.10.4 for definition, initialization, reading and writing calibration and measurement variables.

The only special part is writing of calibration variable. It is achieved through EcoCAL 'Program' or 'Download' function. EcoCAL is an advanced calibration tool developed by Ecotron. 'Download' option can save calibration data to VCU RAM, while 'Program' option can write calibration data to VCU flash. 'Upload' can help upload the existing calibration data from VCU flash to PC.

6.3.2 Non-Volatile Variable

There are two types of non-volatile variables, NVM variables and Fixed NVM variables, for different application purposes. Please refer to section 4.8 for definition, reading and writing non-volatile variables.

Ecotron non-volatile memory theory is described in Appendix A.

Chapter 7 Custom Variable Type

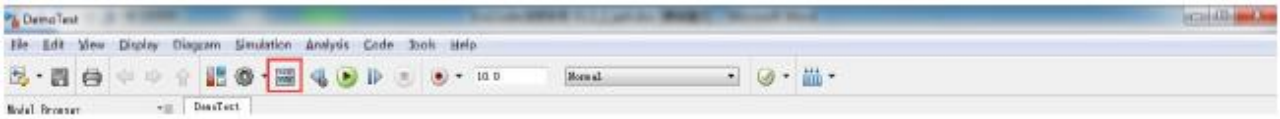
There are two ways to define monitoring/calibration/NVM variables. One is to custom variable type, and the other is to use the definition block in the EcoCoder library.

The method in this chapter eliminates the need for software engineers to load multiple monitor/calibration/NVM variable blocks during simulation by simply defining the variables in "Base Workspace" and save them as m files.

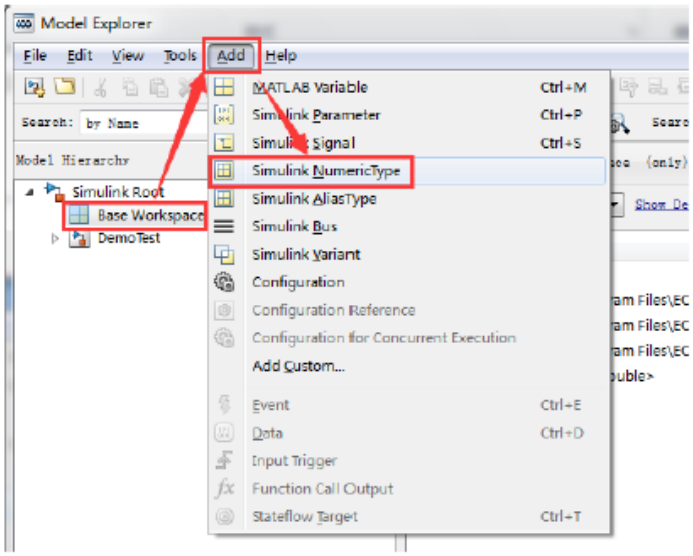
EcoObj is a custom data storage definition package. It is an extension of the simulated signal object and the simulation parameter object. Define custom data objects and classes by using the ASAP2 standard. It generates the product code and the ASAP2 file (or a2l file). You can use EcoObj or MATLAB's "Model Explorer" to define types and variables in M files, the following sections describe the graphical definition method.

7.1 Customize Variable Types

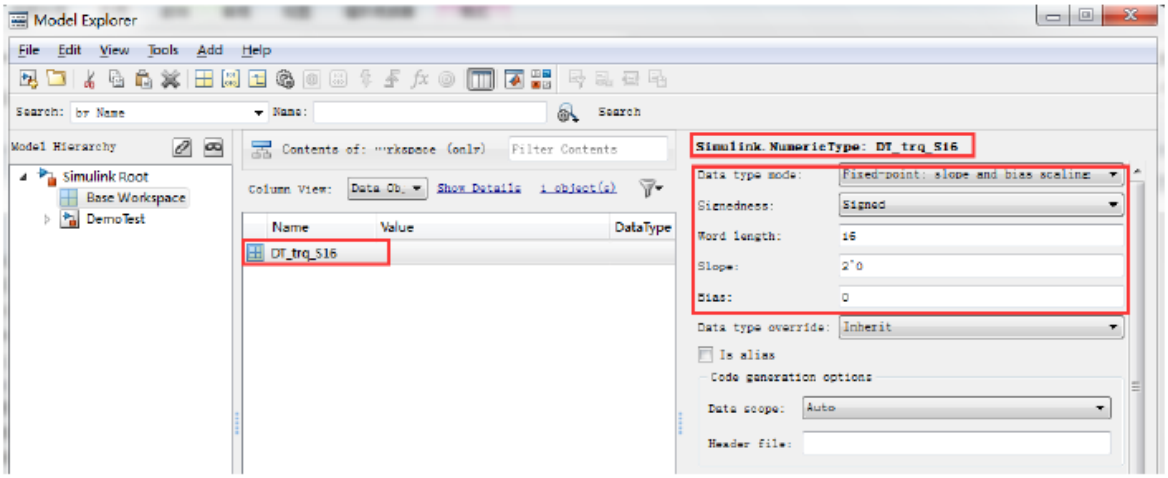
1. Open “Model Explorer”



2. Base Workspace > Add > Simulink Numeric Type



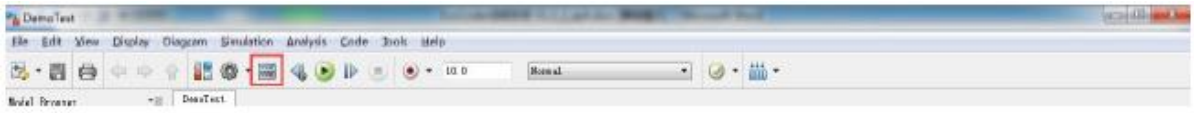
3. Name the variable and set the properties through the window on the right



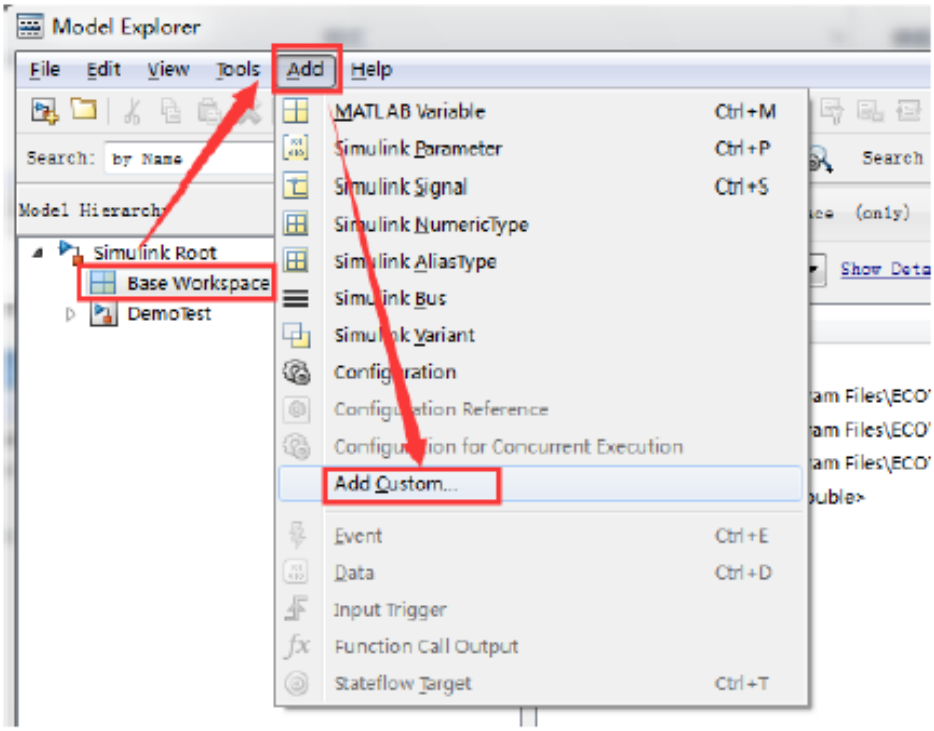
7.2 Add Variables to Workspace

Add "EcoObj.Signal" or "EcoObj.Parameter" to "Base Workspace" via "Model Explorer" as shown below.

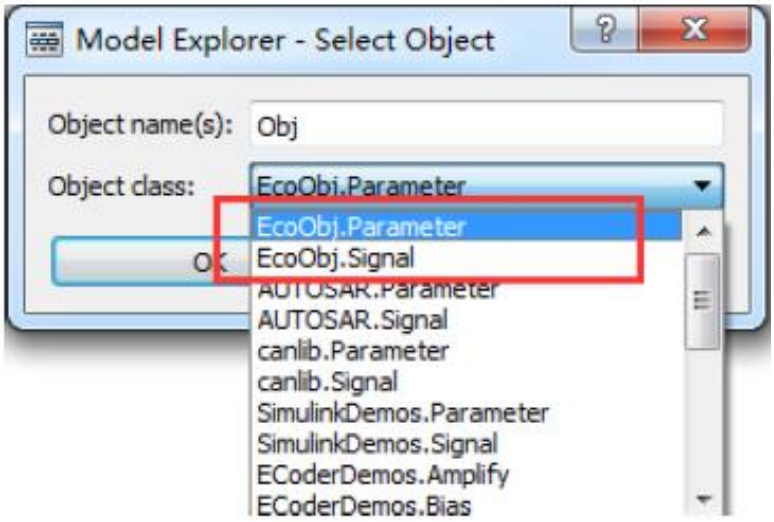
1. Open “Model Explorer”



2. Base Workspace > Add > Add Custom…

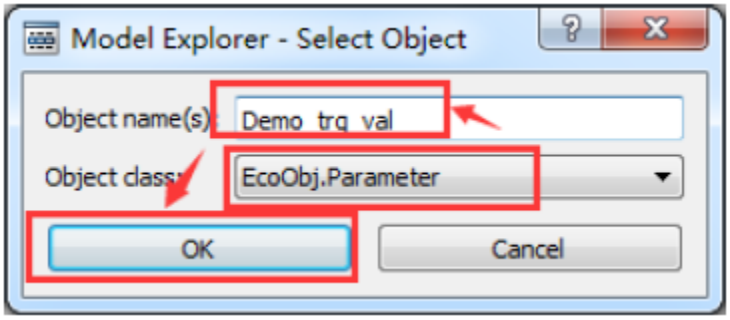


3. Click “Add Custom”, choose “EcoObj.Signal” or “EcoObj.Parameter”.

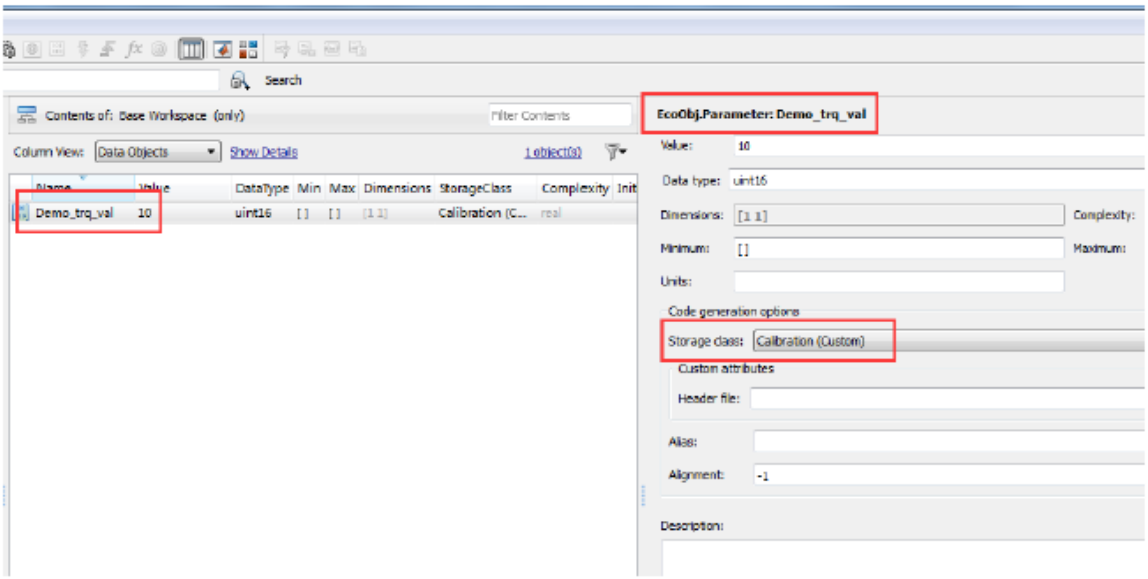


7.3 Customize Calibration Variables

1. Choose “EcoObj.Parameter”, name the variable then click “OK”.

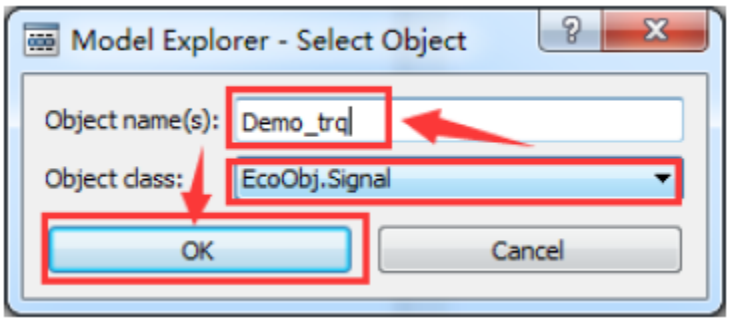


2. Set the properties through the window on the right. To define calibration variables, “Calibration(Custom)” must be chosen in “Storage Class”.

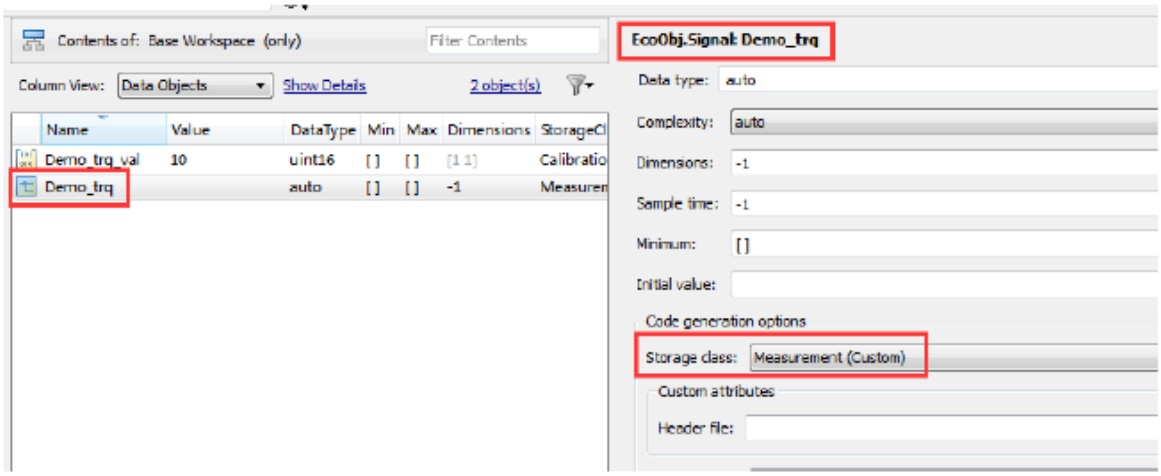


7.4 Customize measurement Variables

1. Choose “EcoObj.Signal”, name the variable then click “OK”.

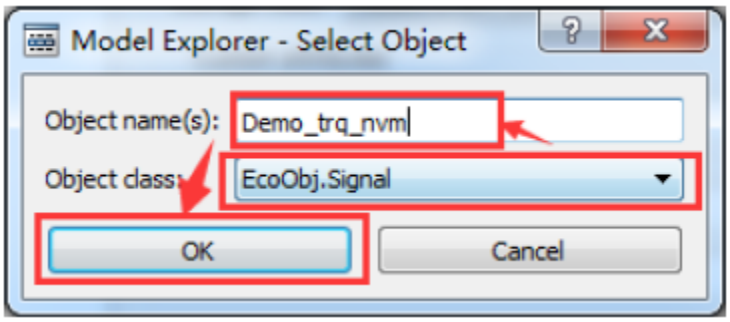


2. Set the properties through the window on the right. To define measurement variables “Measurement (Custom)” must be chosen in “Storage Class”.

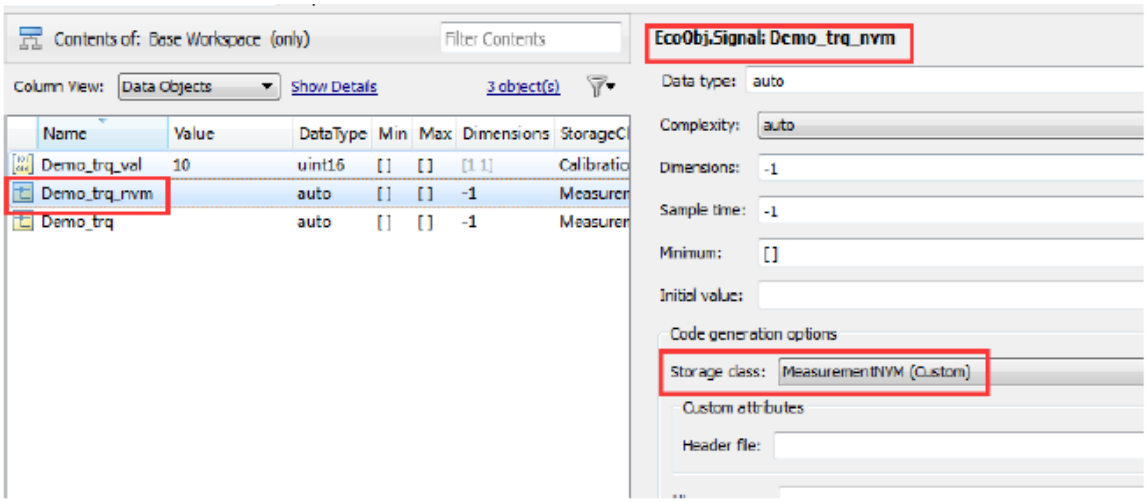


7.5 Customize NVM Variables

1. Choose “EcoObj.Signal”, name the variable then click “OK”.

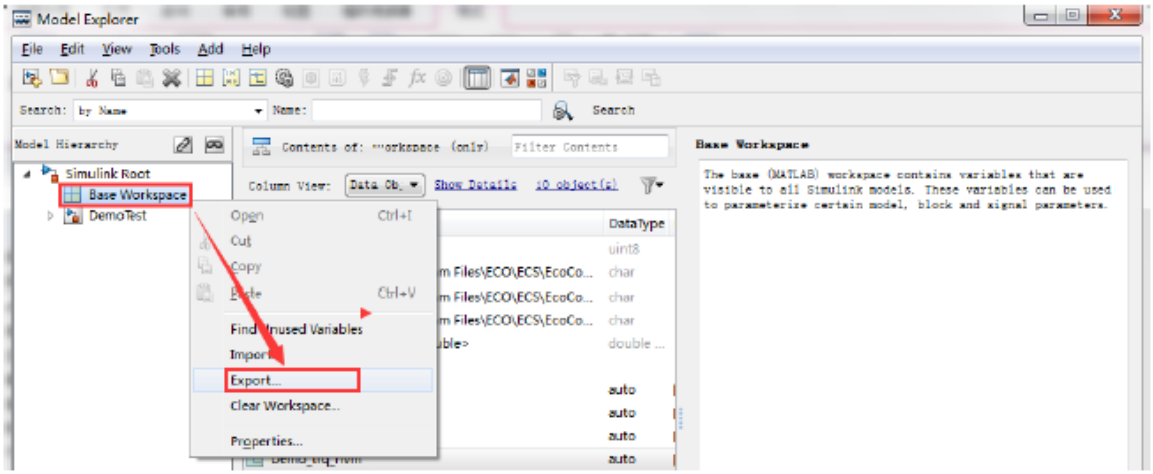


2. Set the properties through the window on the right. To define NVM variables, “MeasurementNvm (Custom)” must be chosen in “Storage Class”.

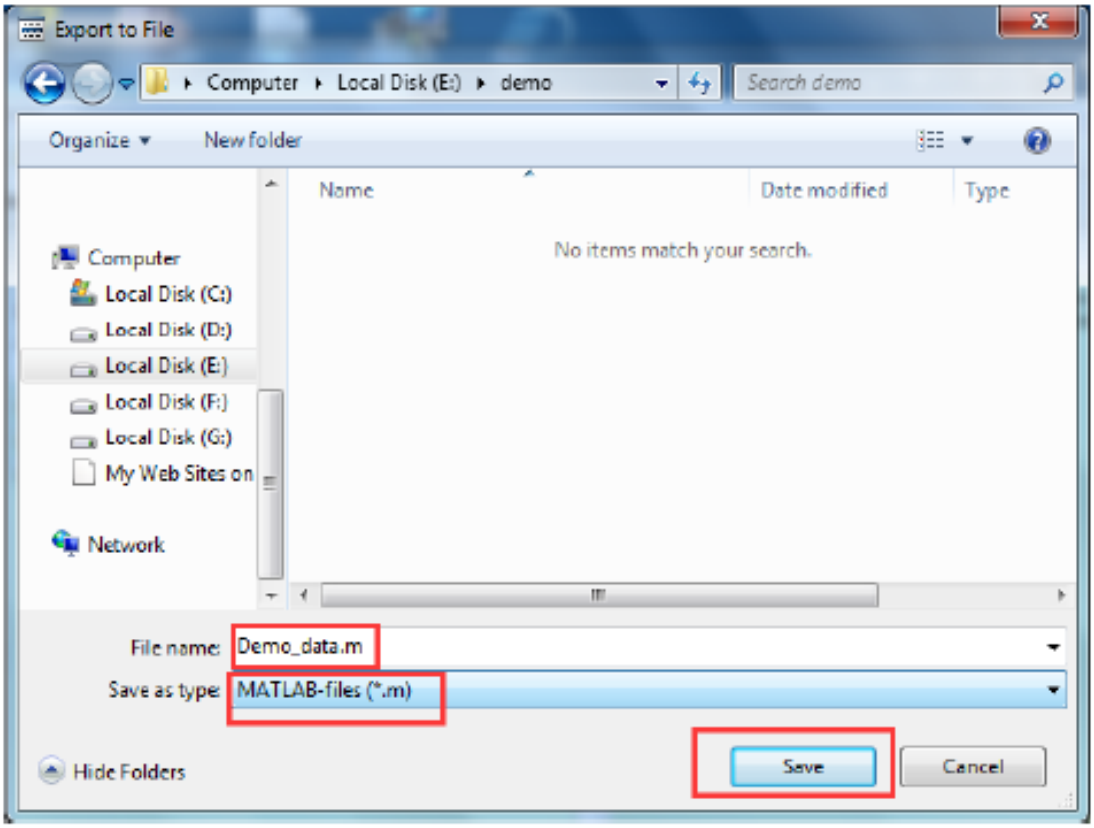


7.6 Save the Variables to M file

1. Base Workspace > Export…

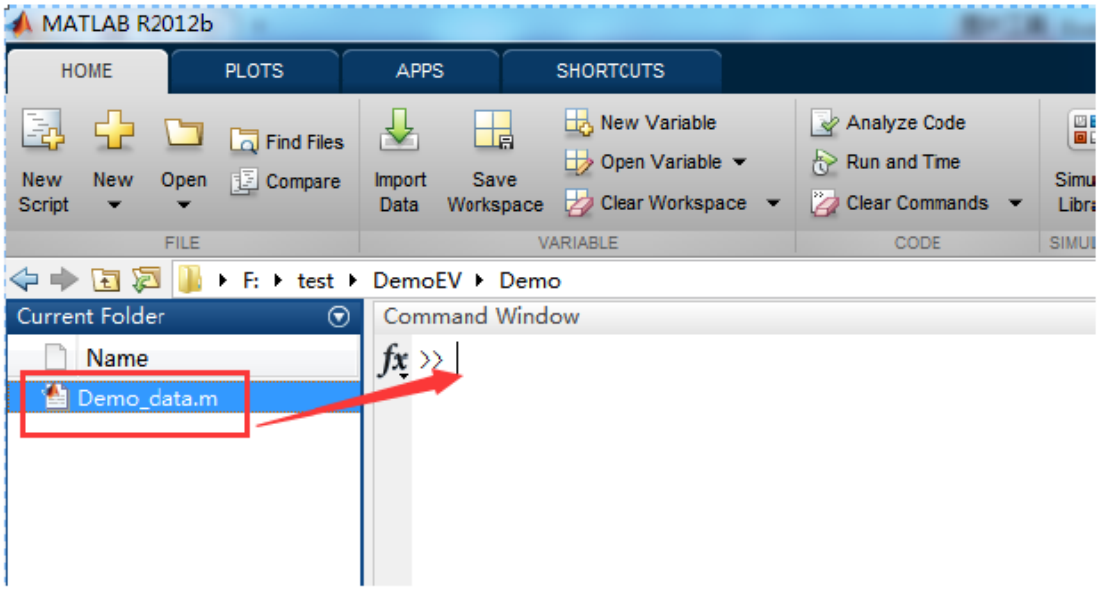


2. Click “Export…”, as shown below, save the file to “Demo_data.m”.

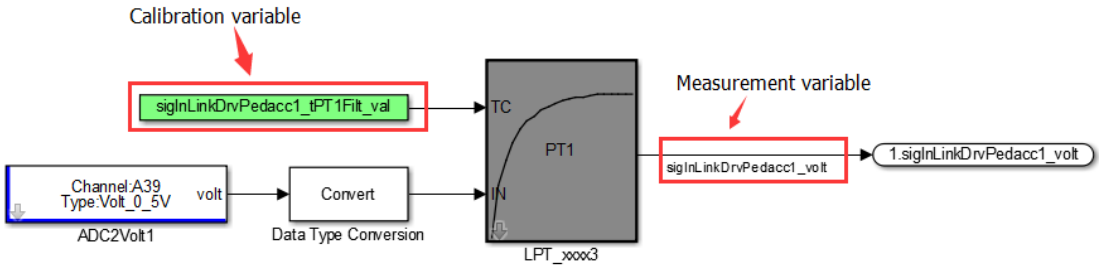


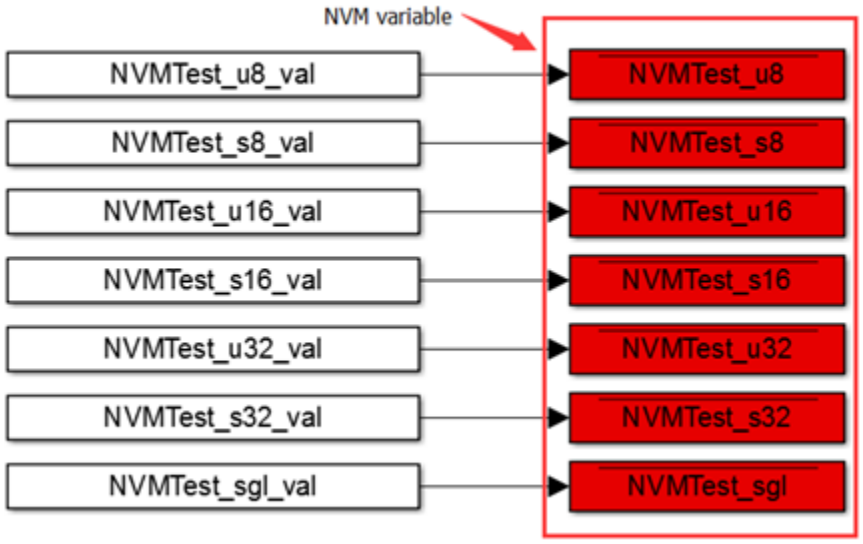
7.7 Load M file to Workspace

Drag “Demo_data.m” file to the “Command Window”.



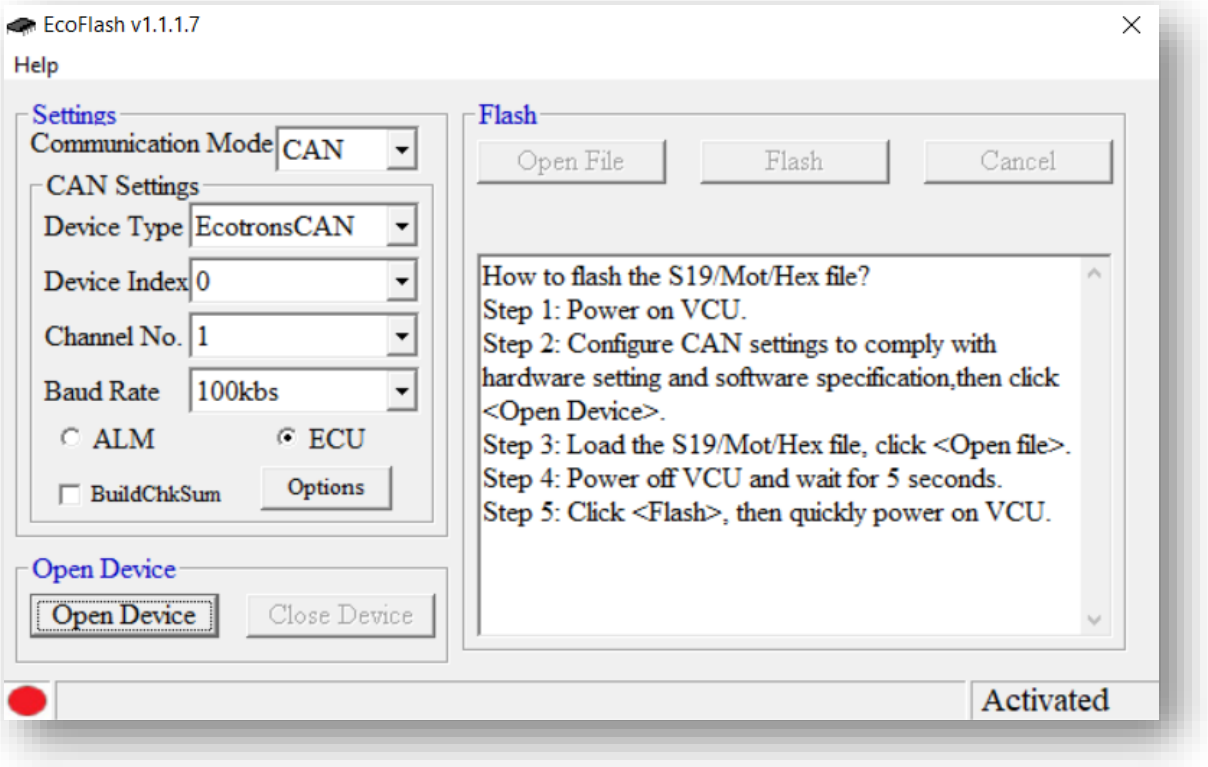
7.8 Model Example





Chapter 8 Programming VCU with EcoFlash

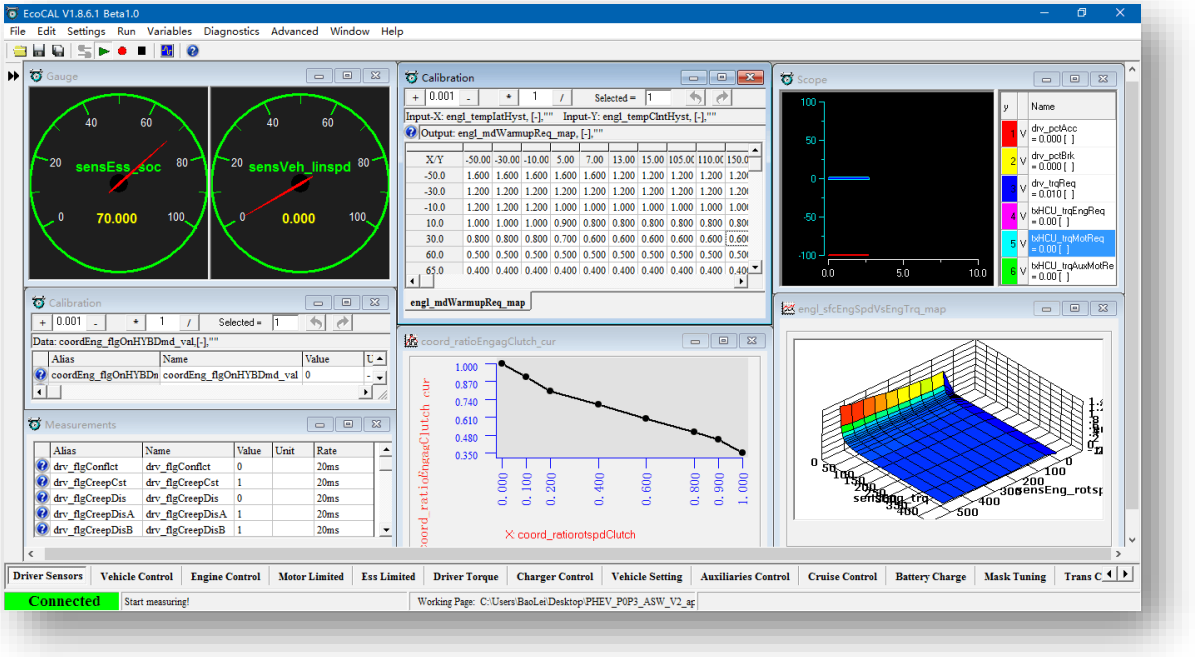
EcoFlash is a user-friendly GUI for programming VCU. Please refer to EcoFlash manual for operation of the software. The below picture provides a quick glance of EcoFlash.



Chapter 9 Measurement and Calibration with EcoCAL

EcoCAL is dedicated for data measurement, calibration, logging and analysis.

It is a professional calibration tool developed by Ecotron. EcoCAL is based on CCP/XCP protocols and uses the CAN bus for communication between master-slave stations. It provides great convenience for VCU in-vehicle testing and prototype development.



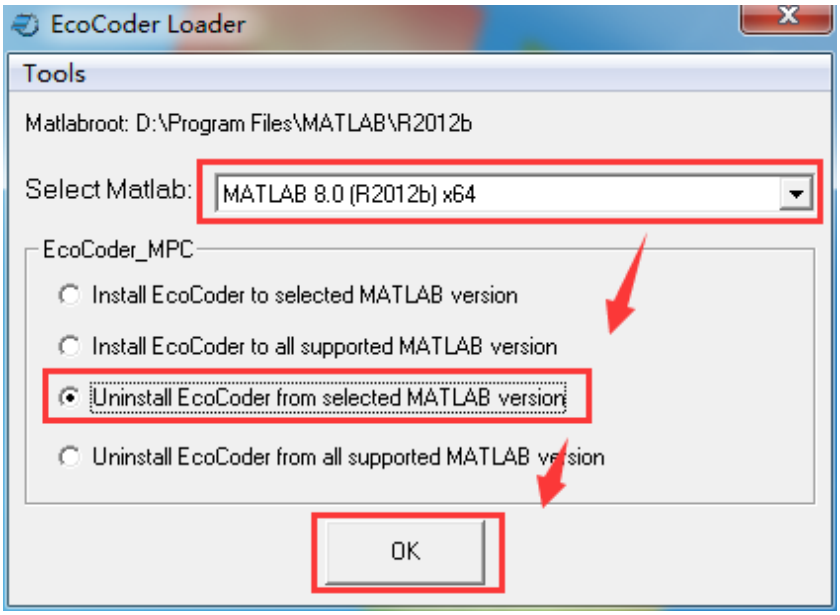
Chapter 10 Uninstall EcoCoder

10.1 Uninstall EcoCoder from MATLAB

Note: You have to close all MATLAB applications before uninstalling.

1. Double-click ‘EcoCoder Loader’.

2. Choose MATLAB version, and select ‘Uninstall EcoCoder from selected MATLAB version’, then click ‘OK’.

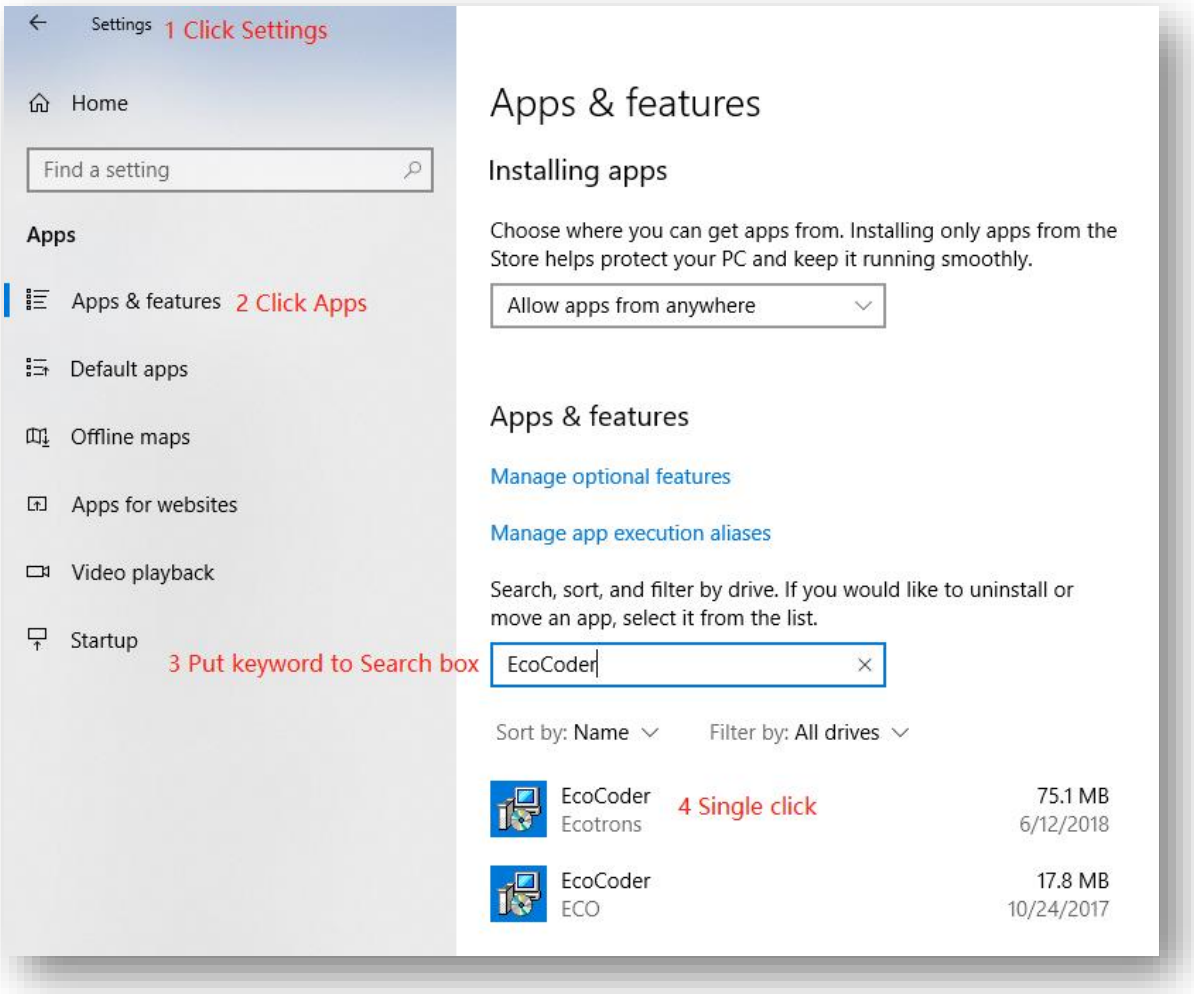


10.2 Uninstall EcoCoder from Windows System

For EcoCoder update, the user will need to uninstall older version EcoCoder from Window before installing new version EcoCoder.

Note: You have to close all open MATLAB applications before uninstalling.

1. Click ‘Start’ and then click ‘settings’, follow steps in picture below



Chapter 11 FAQs

11.1 Q1. The m file exported from DBC by 'EcoCAN' can't be used

A1. The name of the m file must match to the C Language variable naming requirement. And it can't be the name of the existing model or m file.

11.2 Q2. Model created by 'EcoCoder_Prj', emulation or code generation error

1. Check if your MATLAB has Fixed-Point Tool license. If not, the use of fixed-point blocks will trigger errors.
2. Make sure all support files are added to path.
3. Check whether necessary MATLAB components are installed.

11.3 Q3. 'CAN' module is blank after being configured

Please check whether the CAN definition .m file is added to Path.

11.4 Q4. EcoCoder Loader Pop-up error

You may have to register the ‘comdlg32.ocx’ to windows.

11.5 Q5. How to update application model to be compatible with updated EcoCoder

a) EcoCoder Target Definition

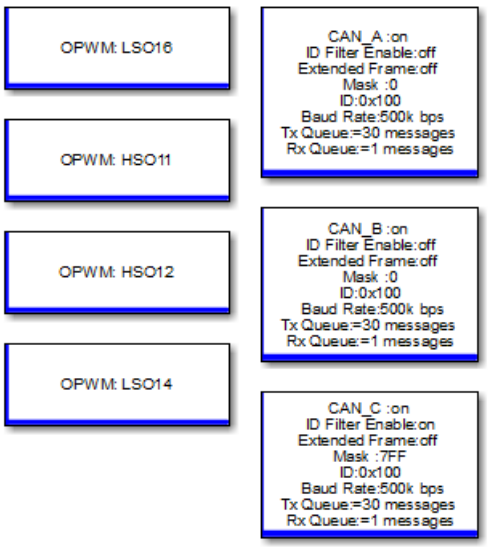
The model must include the EcoCoder Target Definition

EcoCoder Target Setting

Target :EV2106B01

b) Model configuration module

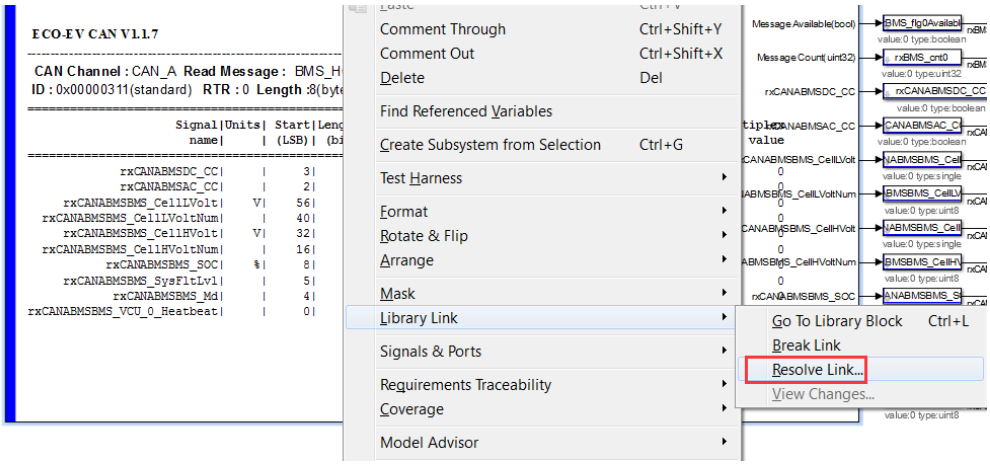
‘ECU_Setting’ module is divided into independent configuration blocks. Please add CAN, OPWM, CCP and other configuration blocks if needed.



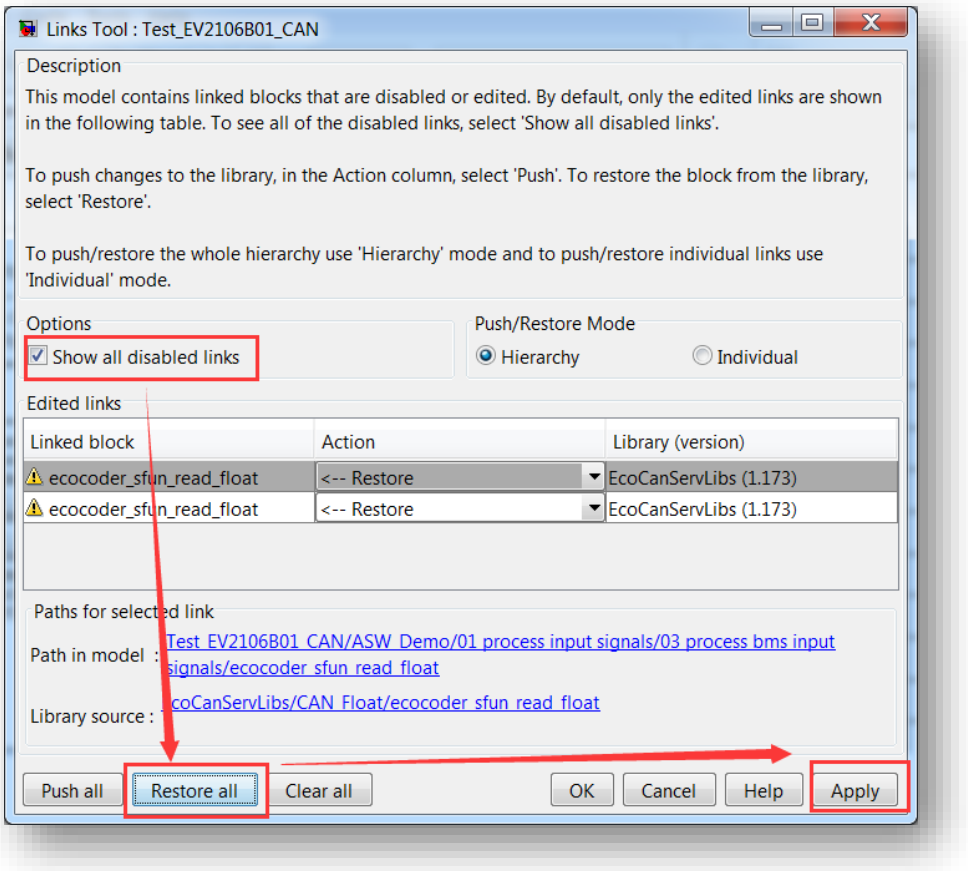
c) Resolve some disabled modules

If CAN library blocks used in the model are disabled, you need to firstly resolve all blocks and save the model before installing new EcoCoder, otherwise the original model will be stuck when using new EcoCoder.

1) Right-click on the disable block, and select Library link->Resolve Link



2) Restore all disable linked blocks.



11.6 Q6. Is there a way to get rid of popping up folder of generated file?

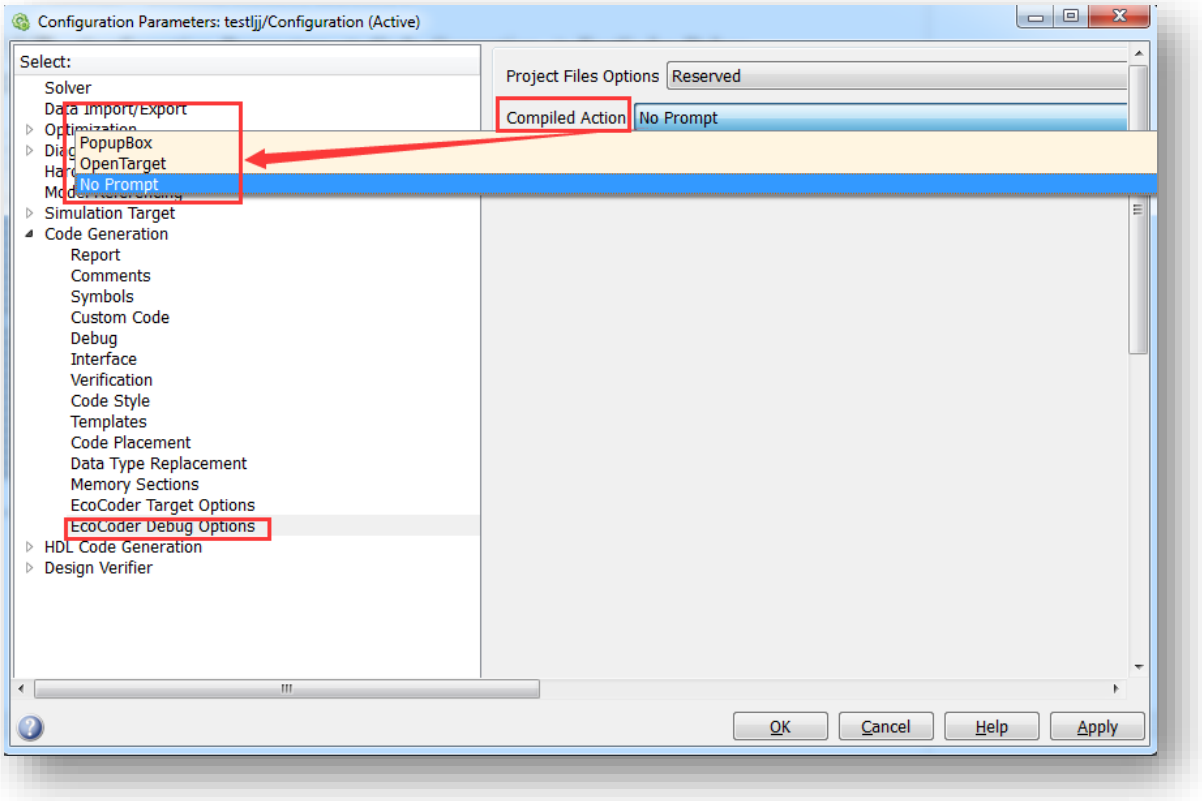
Yes. Please go through these steps, Configuration Parameters -> Code Generation -> EcoCoder Debug Options -> Compiled Action. Then you can configure.

Compiled Action includes: **No Prompt, OpenTarget, PopupBox**.

No Prompt: There is no any prompt when it finishes generating file.

OpenTarget: It will open folder which has generated files.

PopupBox: ‘Software has been compiled successfully!’ will pop up when it finishes generating files.



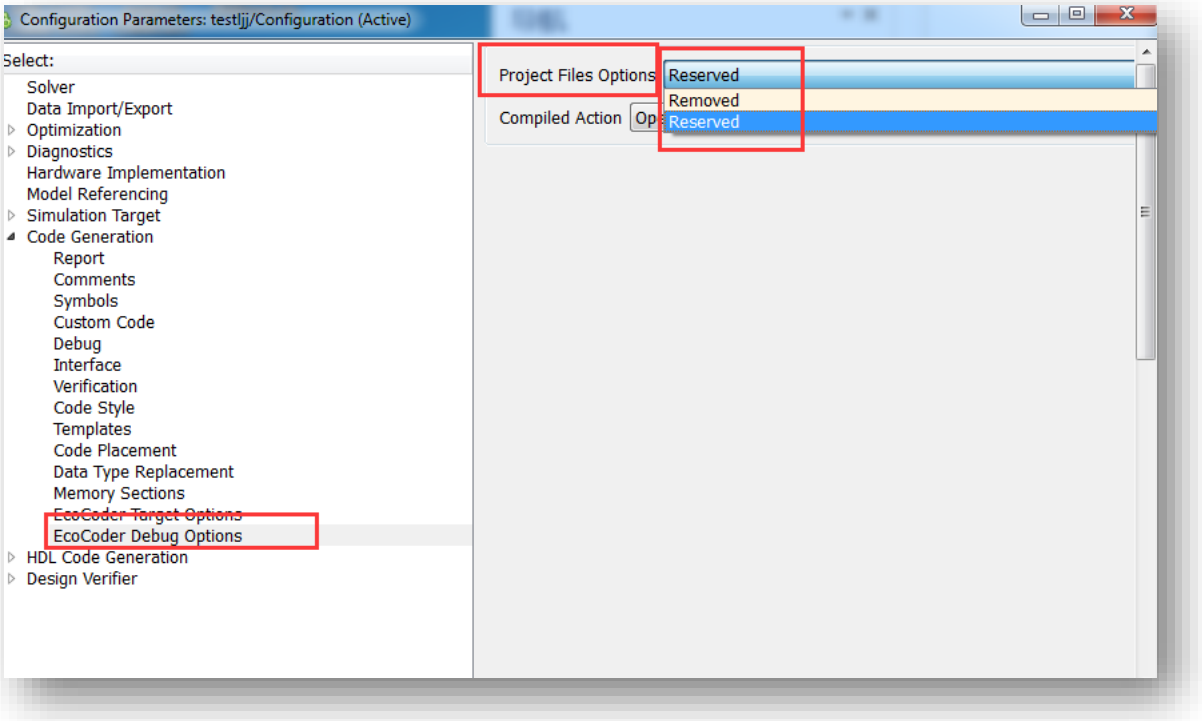
11.7 Q7. Is there a way to access project file in C code?

Yes, go same steps as Q6 then configure box of 'Project Files Options'.

Project Files Options includes **Reserved, Removed**.

Reserved: 'XX_CWprj' will be reserved when it finishes generating file, so user can access C code from 'XX_CWprj ' file.

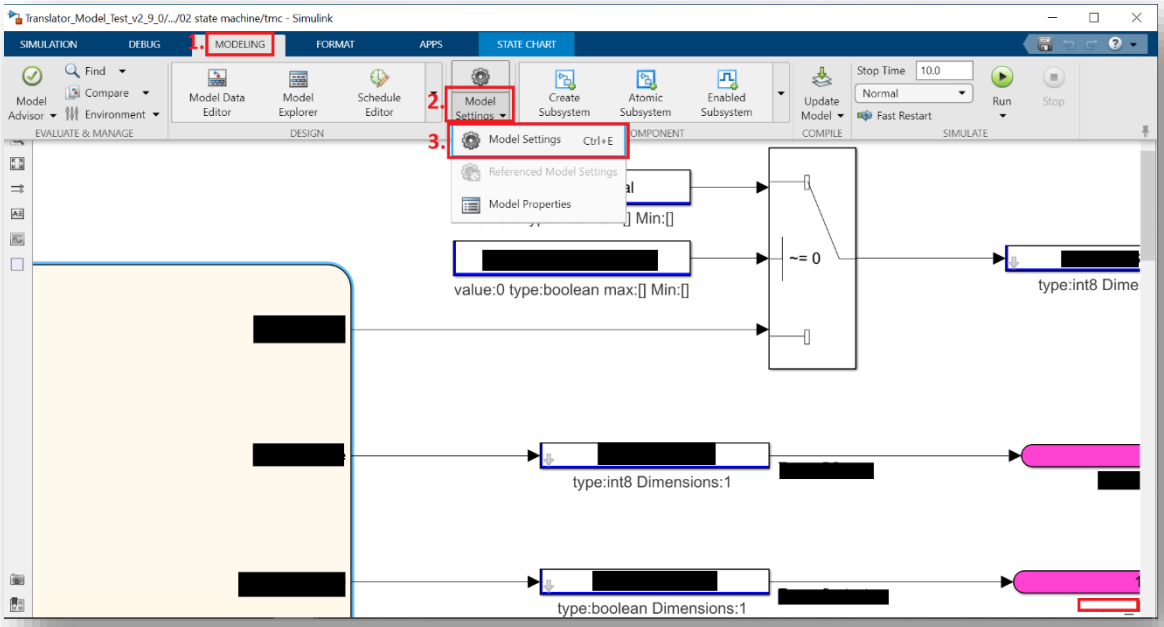
Removed: 'XX_CWprj' will be removed when it finishes generating files.

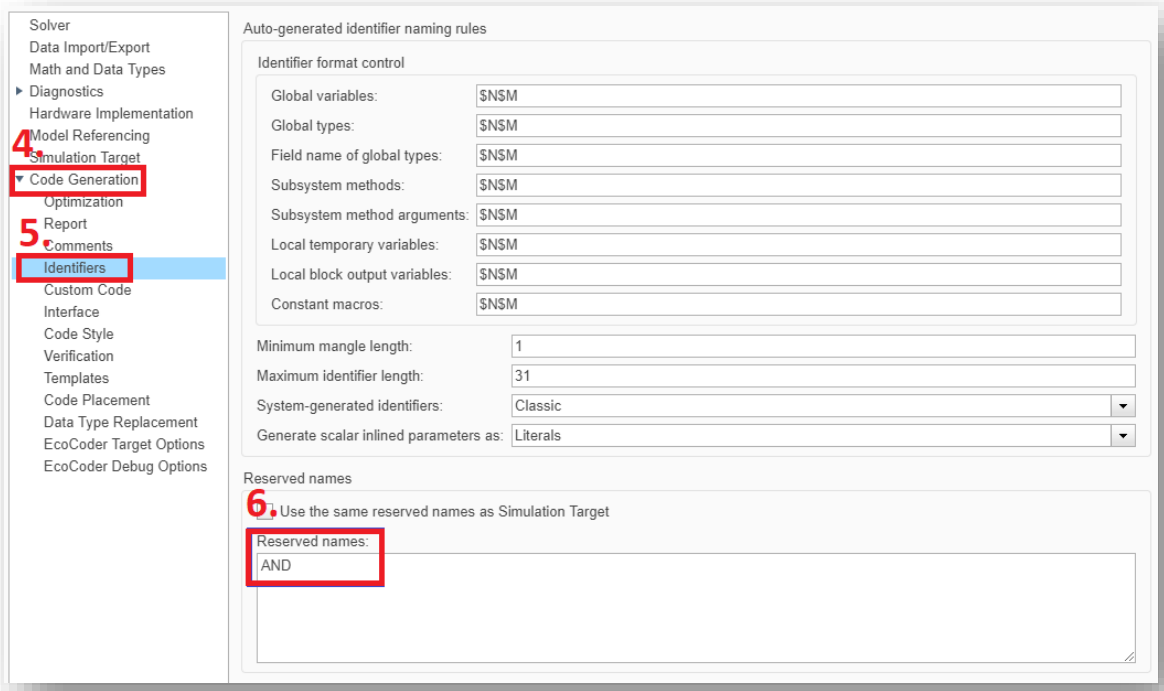


11.8 Q8. C code generation by Simulink is not accepted by compilers.

C code generation by Simulink will sometimes have the variable name not accepted by compilers, for example as 'AND'.

There is one simple way to solve this by reserving the variable Simulink is using. For example, the variable 'AND' is not accepted by Infineon High-Tec Tri-core compiler, but Simulink is generating code using 'AND' as one of the variable names since 'AND' is not restricted in Simulink code generation. As being said, please be aware that the Simulink generated code satisfies the coding rules for C code, but not necessarily satisfies the rules of compiler: *HighTec TriCore Tool Chain* and *S32DesignStudio.* Follow the steps below to reserve the 'AND' to prevent Simulink using it. Please note, the code is generated by Matlab/Simulink, not by EcoCoder. The machine generated code might not be friendly to read. Please follow the MAAB when designing your Simulink model. That way, you can reduce the error in the generated code.


Translator_Model_Test_v2_9_0/.../02 state machine/tmc - Simulink
SIMULATION
DEBUG
1.
MODELING
FORMAT
APPS
STATE CHART
Model Advisor
Find
Compare
Environment
EVALUATE & MANAGE
Model Data Editor
Model Explorer
Schedule Editor
DESIGN
2.
Model Settings
Create Subsystem
Atomic Subsystem
Enabled Subsystem
3.
Model Settings Ctrl+E
Referenced Model Settings
Model Properties
Update Model
COMPILE
Stop Time
10.0
Normal
Fast Restart
Run
Stop
SIMULATE
Min:[]
value:0 type:boolean max:[] Min:[]
~= 0
type:int8 Dime
type:int8 Dimensions:1
type:boolean Dimensions:1


Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
4.
Code Generation
Optimization
Report
5.
Comments
Identifiers
Custom Code
Interface
Code Style
Verification
Templates
Code Placement
Data Type Replacement
EcoCoder Target Options
EcoCoder Debug Options
Auto-generated identifier naming rules
Identifier format control
Global variables: NM
Global types: NM
Field name of global types: NM
Subsystem methods: NM
Subsystem method arguments: NM
Local temporary variables: NM
Local block output variables: NM
Constant macros: NM
Minimum mangle length: 1
Maximum identifier length: 31
System-generated identifiers: Classic
Generate scalar inlined parameters as: Literals
Reserved names
6.
Use the same reserved names as Simulation Target
Reserved names:
AND

Appendix A- Nonvolatile Variables

This is a description of assigning variable values to non-volatile memory and fixed non-volatile memory locations, how to change such values and notice for operation.

Non-volatile Variables

The NVM variables are stored in flash, which can maintain information even the VCU is powered off (unlike RAM, which would lose data after the VCU power off).

NVM data values are read from flash and written into RAM when VCU is powered on. The RAM variables can be read/written as many times as needed since RAM has high Program/Erase cycle. When key off signal is detected, power off logic would trigger the process of storing NVM variables from RAM to flash. An example power down block, which includes this NVM variable storage logic, *Power Management Example*, is provided in EcoCoder library.

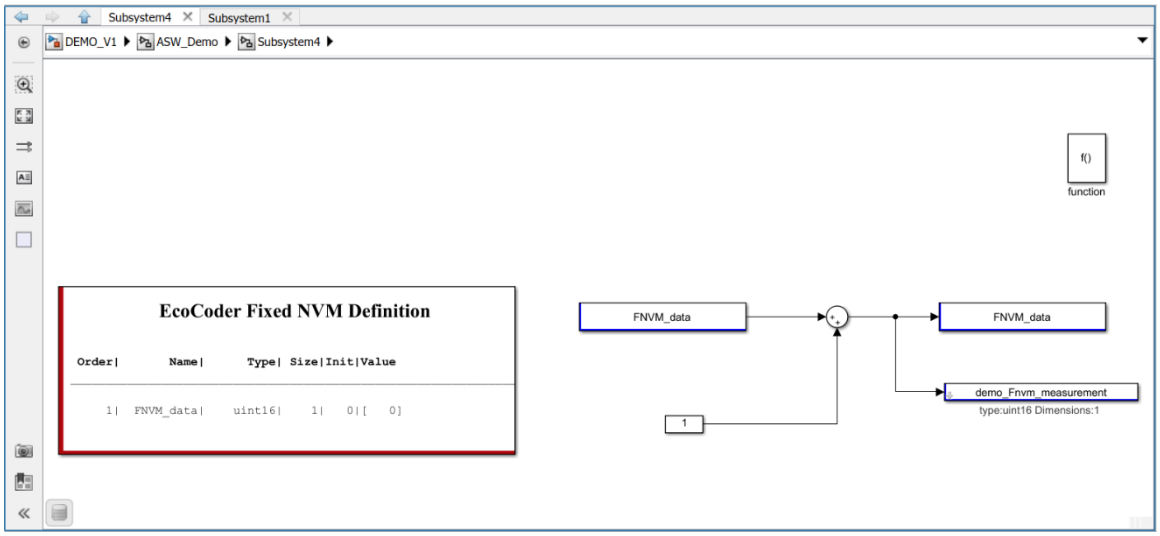
Note: The power input to VCU BATT is required to be uninterrupted to make sure the process of storing of NVM variables value is safe. If power is lost unexpectedly while application is running, the value of the NVM variables on the next key-on will be the same value as what had been saved into flash during the last controlled shutdown. If power is lost unexpectedly during the controlled shutdown procedure (when the process of saving NVM variables into flash is supposed to be happening), all NVM variables will revert to their default values (defined in the application software).

Fixed Non-volatile Variables

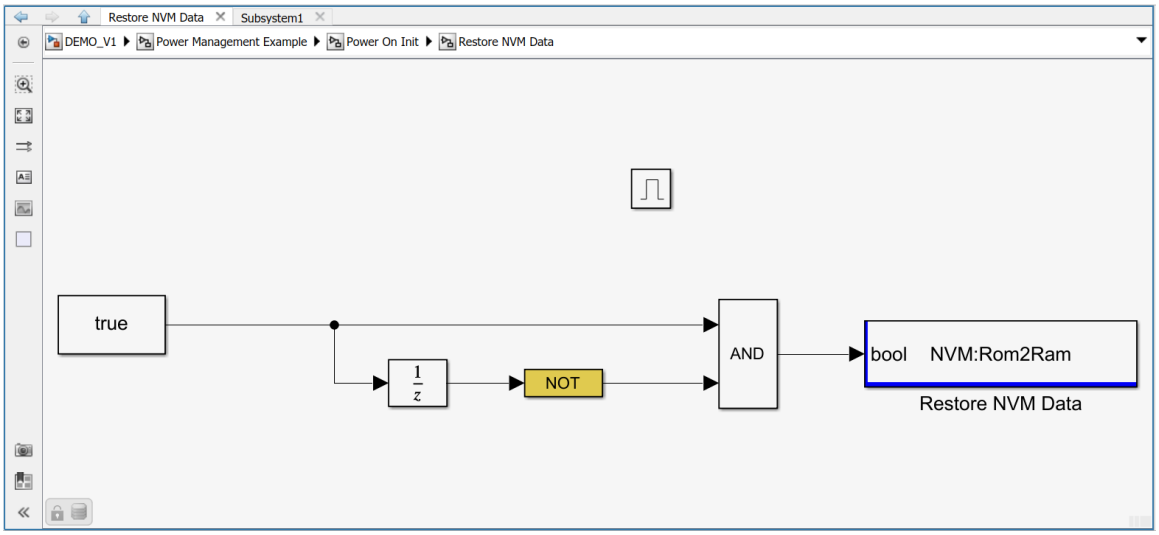
The Fixed NVM variables will be kept the same even the VCU is programmed (unless it is required to be changed by user configuration), so critical data such as odometer data will not be lost even the VCU software update is performed. The fixed NVM variables are stored in specific space of flash and arranged in the order defined block, which means specific addresses in flash are reserved for specific variables.

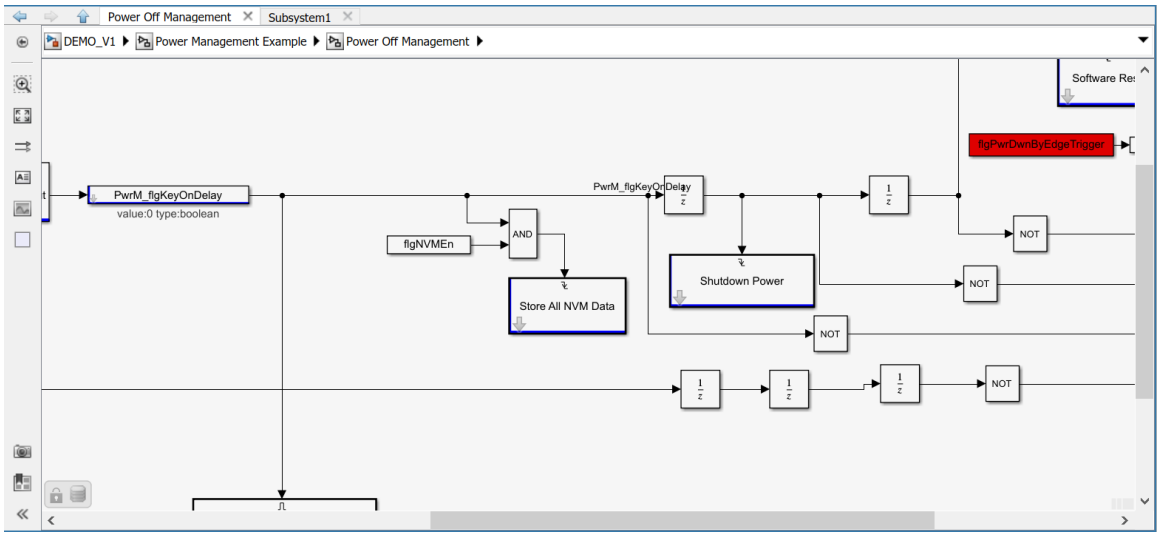
If new variables need to be added to fixed NVM space, it is necessary to re-initialize by calling definition block.

Fixed NVM Demo



To include Fixed NVM in the control strategy, Fixed NVM Definition, Read Fixed NVM, and Write Fixed NVM blocks are needed. For details about parameters in each block, please refer to section 4.8. The subsystem above will increase the value of the Fixed NVM variable by 1 every time this subsystem is executed. Other two blocks, Restore NVM Data and Store All NVM Data, are also necessary to fulfill the function of Fixed NVM. They are included in default power management block as shown below. Please include them in the control strategy if default power management block is not used.




Power Off Management
Subsystem1
DEMO_V1
Power Management Example
Power Off Management
Software Re
flgPwrDwnByEdgeTrigger
PwrM_flgKeyOnDelay
value:0 type:boolean
PwrM_flgKeyOnDelay
flgNVMEn
AND
Store All NVM Data
Shutdown Power
NOT
NOT
NOT
NOT

Appendix-B Battery Input

As mentioned previously, power supply has to be maintained at least for a short period after key-off, in order for the VCU to execute the shutdown process.

The shutdown process implemented by block 'Power Management Example' includes stopping the application and saving NVM variables that have been temporarily stored in RAM to flash, during the power-off delay, after key-off. This is the recommended way to save nonvolatile variables to flash. If the frequency of calling 'Store All NVM Data' block is too high, errors might occur.

Table 1.

Status Value	Supported MinGW version
0	Successful operation
1	Insufficient space, available active area block is less than set active block
2	Flash operation error
3	Block operation error
4	Block detection error
5	Insufficient space
6	Need to erase
7	Abnormal block status
8	Parameters error
9	Record not found
10	Record type does not match
11	Record deleted
12	Record replication succeeded
13	Writing a record
14	Executing a swap operation
15	Records need to be written to new activity area
16	Needs to be erased

17	Read length does not match

Table 2

Status Code	Description
0	Bus Normal
1	Parameter Error
2	Bus Busy

Appendix C-MAB(MathWorks Advisory Board)

For your information, below is from MathWorks' guideline for model construction.

MathWorks Advisory Board (MAB) guidelines stipulate important basic rules for modeling in Simulink and Stateflow. The overall purpose of these modeling guidelines is to allow for a simple, common understanding by modelers and consumers of control system models.

The main objectives of these guidelines are:

Readability

1. Improve graphical understandability
2. Improve readability of functional analysis
3. Prevent connection mistakes
4. Comments, etc.

Simulation and verification

1. Mechanism to enable simulation
2. Testability

Code Generation

1. Improve the efficiency of code generation (ROM, RAM efficiency)
2. Ensure the robustness of generated code

If more information is needed, please refer to the link here.