# EcoCoder Manual

V4.9.2

## 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 |
| 01/28/2021 | 4.9.2 | 1. Update target definition block<br>2. Update set up information | Jason Du |

# Contents

**CHAPTER 1 GENERAL INFORMATION** ..... 14

**1.1 About EcoCoder** ..... 14

**1.2 Operating System Requirements** ..... 14

**1.3 MATLAB Installation Requirements** ..... 15

1.3.1 MATLAB R2010b ..... 15

1.3.2 MATLAB R2011a-R2021a ..... 15

**1.4 Supported MATLAB Version** ..... 15

**CHAPTER 2 ECOCODER DEVELOPMENT ENVIRONMENT** ..... 17

**2.1 Software Installation List** ..... 17

**2.2 CodeWarrior Installation** ..... 19

**2.3 MinGW-GCC Compiler Installation** ..... 20

**2.4 C++ Compiler Installation** ..... 23

2.4.1 Installation of Compiler for MATLAB 32-Bit ..... 23

2.4.2 Compiler Installation for MATLAB 64-Bit ..... 24

**2.5 EcoCoder Installation** ..... 24

**2.6 Link S32DS_Power_Win32 to EcoCoder** ..... 28

**2.7 Link HighTec TriCore Tool Chain to EcoCoder** ..... 31

**2.8 Link CS+ to EcoCoder** ..... 33

**2.9 Link S32DS_ARM_Win32 to EcoCoder** ..... 33

**2.10 Activate EcoCoder** ..... 36

2.10.1 Active EcoCoder by USD Dongle ..... 36

2.10.2 Active EcoCoder by license file ..... 36

**2.11 Quick start for building application model ..... 40**

**CHAPTER 3 ECOCODER LIBRARY ..... 43**

**3.1 EcoCoder Target Definition ..... 43**

**3.2 ADC Analog Inputs ..... 45**

3.2.1 Read ADC Value ..... 45
3.2.2 Read Fixed-Point ADC Volt ..... 46
3.2.3 Read Float ADC Volt ..... 47
3.2.4 Read High Resolution ADC Value ..... 48

**3.3 CAN Communication ..... 49**

3.3.1 CAN Channel Definition ..... 49
3.3.2 CAN Wake-up Frame Definition ..... 51
3.3.3 Read Fixed-Point CAN Message ..... 52
3.3.4 Read CAN Message ..... 55
3.3.5 Send Fixed-Point CAN Message ..... 57
3.3.6 Send CAN Message ..... 59
3.3.7 Receive CAN Message ..... 61
3.3.8 Transmit CAN Message ..... 62
3.3.9 Send CAN Data ..... 63
3.3.10 CAN Receive Counter ..... 63
3.3.11 Set CAN Mode ..... 64
3.3.12 Unpack CAN Data to Signals ..... 65
3.3.13 Pack Signals to CAN Data ..... 65
3.3.14 Read CAN Error State ..... 66
3.3.15 Receive CAN Raw with Trigger ..... 66
3.3.16 Recover CAN Bus Off ..... 67
3.3.17 CAN MESSAGE2NormalType ..... 67
3.3.18 CAN NormalType2NormalType ..... 68
3.3.19 Receive CANFD Raw With Trigger ..... 69
3.3.20 Transmit CANFD Message ..... 70
3.3.21 Disable CAN transceiver and Wake-up ..... 71

**3.4 Serial Communication Interface (SCI) Block ..... 71**

3.4.1 SCI Definition....71
3.4.2 Read SCI Data....72
3.4.3 Send SCI Data....73

**3.5 Digital I/O....73**
3.5.1 Switch Input....73
3.5.2 KeyOn Input....74
3.5.3 Switch Output....75
3.5.4 IPM Read....76
3.5.5 PWM IO Frequency Range Definition....76
3.5.6 IPWM Read....77
3.5.7 IPWM Interrupt Handler Definition....77
3.5.8 IPWM Read with Interrupt Handler....78
3.5.9 PWM Definition....79
3.5.10 PWM Output....79
3.5.11 PPM Read....80
3.5.12 Switch Internal....81
3.5.13 WakeUp Input....81
3.5.14 H-bridge Definition....82
3.5.15 H-bridge Output....82
3.5.16 INA226 Definition....83
3.5.17 Get INA226 Current....84
3.5.18 Get INA226 Shunt Voltage....84
3.5.19 Quadrature Decoder Definition....85
3.5.20 Quadrature Decoder Input....85
3.5.21 Quadrature Decoder Set Counter....86
3.5.22 Get Switch Output Sense Current....87
3.5.23 PeakHold Definition....87
3.5.24 PeakHold Output....88

**3.6 LIN Communication....88**
3.6.1 LIN Channel Definition....89
3.6.2 LIN Get Status....89
3.6.3 LIN Receive Data....90
3.6.4 LIN Transmit Data....91
3.6.5 LIN Master Transmit ID And Data....92

3.6.6 LIN Master Transmit ID ..... 93
3.6.7 LIN Master Receive Data ..... 93

**3.7 Task Scheduler ..... 94**
3.7.1 Task Trigger ..... 94
3.7.2 Task Monitor ..... 95

**3.8 Non-Volatile Memory Blocks ..... 96**
3.8.1 Fixed NVM Definition ..... 97
3.8.2 Read Fixed NVM ..... 99
3.8.3 Write Fixed NVM ..... 99
3.8.4 NVM Definition ..... 100
3.8.5 NVM Variable Definition ..... 100
3.8.6 Read NVM ..... 101
3.8.7 Write NVM ..... 102
3.8.8 Store All NVM Data ..... 103
3.8.9 Restore All NVM Data ..... 103
3.8.10 Get Fixed NVM Variable Address ..... 104

**3.9 Diagnostic Blocks ..... 104**
3.9.1 Hardware Output DTC ..... 104
3.9.2 DTC Parser ..... 106
3.9.3 Software Core Diagnostic ..... 106
3.9.4 Clear H-bridge DTC ..... 107
3.9.5 Clear TLF35584 All DTC ..... 107
3.9.6 Get TLF35584 Summary DTC ..... 107
3.9.7 Get TLF35584Specific DTC Total Number ..... 108
3.9.8 Get TLF35584 Specific DTC ..... 108

**3.10 Calibration & Measurement ..... 109**
3.10.1 Calibration Definition ..... 109
3.10.2 Override Probe ..... 110
3.10.3 Read Calibration ..... 111
3.10.4 Write Measurement ..... 112
3.10.5 Write and Read Measurement ..... 113
3.10.6 1-D Lookup Table ..... 115

3.10.7 2-D Lookup Table ..... 115
3.10.8 Calibration Data Check ..... 116

**3.11 System Management Blocks ..... 117**
3.11.1 Power Management Example ..... 117
3.11.2 Shutdown Power ..... 119
3.11.3 Shutdown Power Hold ..... 119
3.11.4 Set ECU Mode ..... 120
3.11.5 ECU Master Chip Wake-Up Definition ..... 121
3.11.6 Watchdog Definition ..... 121
3.11.7 Service Software Watchdog ..... 122
3.11.8 Software Reset ..... 122
3.11.9 Software Reset Hold ..... 123
3.11.10 Read System Free Counter ..... 123
3.11.11 Power Control Output ..... 124
3.11.12 Stack Overflow Detection Definition ..... 125
3.11.13 Detect Stack Overflow ..... 125
3.11.14 Read BootID ..... 126
3.11.15 Hardware Reset ..... 126
3.11.16 Power on the ACU ..... 126
3.11.17 Power off the ACU ..... 127
3.11.18 Get RTC Current Data And Time ..... 127
3.11.19 Set RTC Current Data And Time ..... 128
3.11.20 Set RTC Fixed-cycle Timer ..... 129

**3.12 CCP ..... 131**
3.12.1 Fixed CCP Slave Definition ..... 131
3.12.2 CCP/CAL Seed&Key Security Definition ..... 133
3.12.3 CCP DAQ Seed&Key Security Definition ..... 134
3.12.4 CCP PGM Seed&Key Security Definition ..... 134
3.12.5 CCP Generate Seed Demo ..... 135
3.12.6 CCP Get Seed Trigger ..... 135
3.12.7 CCP Set Seed ..... 136

**3.13 Programming Blocks ..... 136**
3.13.1 Online Programming Definition ..... 136

3.13.2 Programming Seed&Key Definition ..... 137
3.13.3 Entry UDS Programming ..... 138

**3.14 Advanced Data Blocks ..... 139**
3.14.1 Read OTP ..... 139
3.14.2 Read OTP (Input port) ..... 139
3.14.3 Write OTP ..... 140
3.14.4 Write OTP (Input port) ..... 141
3.14.5 Read Data by Address ..... 141
3.14.6 Read Data by Address (Input port) ..... 142
3.14.7 Read String Value ..... 143
3.14.8 Read EEPROM ..... 143
3.14.9 Write EEPROM ..... 144
3.14.10 EEPROM Emulation Definition ..... 144
3.14.11 Clear ALL EEPROM Emulation Record ..... 145
3.14.12 Clear One EEPROM Emulation Record ..... 145
3.14.13 Read EEPROM Emulation Record ..... 146
3.14.14 Write EEPROM Emulation Record ..... 147
3.14.15 Read Signals from EEPROM Emulation Record ..... 147
3.14.16 Write Signals to EEPROM Emulation Record ..... 149
3.14.17 EEPROM Emulation Area Need to Erase ..... 150
3.14.18 Erase EEPROM Emulation Area ..... 150
3.14.19 Program First Run Flag ..... 151
3.14.20 Write RAM Data by Address ..... 151
3.14.21 Get General Variable Address ..... 152

**3.15 Application Base Blocks ..... 153**
3.15.1 Rising Edge ..... 153
3.15.2 Falling Edge ..... 153
3.15.3 Online Programming by SoftReset ..... 153
3.15.4 Online Programing By HardReset ..... 155
3.15.5 dt – time step length ..... 156
3.15.6 Rising Edge Debounce ..... 156
3.15.7 Falling Edge Debounce ..... 157
3.15.8 PT1 Filter ..... 157
3.15.9 Hysteresis ..... 158

3.15.10 SR Flip Flop ..... 158

**3.16 Volatile Variable ..... 159**

3.16.1 Read Const Volatile Variable ..... 159

**3.17 XCP Module ..... 160**

3.17.1 XCP Slave Definition ..... 160

**3.18 FlexRay Module ..... 160**

3.18.1 FlexRay Definition ..... 160

3.18.2 FlexRay Get Current State ..... 162

3.18.3 FlexRay Setup Network Control ..... 162

3.18.4 FlexRay Force to Halt ..... 163

3.18.5 FlexRay Restore to Default Configuration ..... 163

3.18.6 Read FlexRay Message ..... 164

3.18.7 Write FlexRay Message ..... 164

**3.19 Ethernet ..... 165**

3.19.1 Ethernet Definition ..... 165

3.19.2 Ethernet Handler ..... 166

3.19.3 Ethernet Init ..... 166

**3.20 TCP Protocol Blocks ..... 166**

3.20.1 TCP Server Definition ..... 166

3.20.2 TCP Server Init ..... 167

3.20.3 TCP Client Definition ..... 168

3.20.4 TCP Client Connect ..... 168

3.20.5 TCP Close ..... 169

3.20.6 TCP Abort ..... 169

3.20.7 TCP State ..... 170

3.20.8 TCP Receive ..... 171

3.20.9 TCP Transmit ..... 171

**3.21 UDP Blocks ..... 172**

3.21.1 UDP Definition ..... 172

3.21.2 UDP Init ..... 173

3.21.3 UDP Receive ..... 173

3.21.4 UDP Transmit ..... 174

**3.22 SPI Blocks ..... 175**

3.22.1 SPI Definition ..... 175

3.22.2 SPI Master Exchange Data ..... 176

3.22.3 SPI Slave Exchange Data ..... 177

**3.23 SPI2J2 ..... 178**

3.23.1 SPI2J2 Definition ..... 178

3.23.2 SPI2J2 Master Enable COM ..... 179

3.23.3 SPI2J2 Master Handler ..... 179

3.23.4 SPI2J2 Master Receive Data ..... 180

3.23.5 SPI2J2 Master Transmit Data ..... 180

3.23.6 SPI2J2 Message Bus Creator ..... 181

3.23.7 SPI2J2 Blank Message ..... 182

3.23.8 SPI2J2 Read Master Ready Counter ..... 182

3.23.9 SPI2J2 Read Master ACK Counter ..... 183

**CHAPTER 4 CAN THEORY OF ECOTRON ..... 184**

**4.1 Convert DBC to m File ..... 184**

**4.2 EcoCoder CAN Blocks ..... 187**

4.2.1 Select CAN library ..... 187

4.2.2 Select m file ..... 187

4.2.3 Select CAN Message ..... 188

4.2.4 Select Sample Time ..... 189

4.2.5 CAN demo model ..... 190

**CHAPTER 5 CUSTOM VARIABLE TYPE ..... 192**

**5.1 Customize Variable Types ..... 192**

**5.2 Add Variables to Workspace ..... 193**

**5.3 Customize Calibration Variables ..... 195**

**5.4 Customize measurement Variables ..... 196**

5.5 Customize NVM Variables ..... 197

5.6 Save the Variables to M file ..... 198

5.7 Load M file to Workspace ..... 199

5.8 Model Example ..... 200

CHAPTER 6 AUTOMATICALLY CODE GENERATION ..... 201

CHAPTER 7 PROGRAMMING VCU WITH ECOFLASH ..... 202

CHAPTER 8 MEASUREMENT AND CALIBRATION WITH ECOCAL ..... 203

CHAPTER 9 UNINSTALL ECOCODER ..... 204

9.1 Uninstall EcoCoder from MATLAB ..... 204

9.2 Uninstall EcoCoder from Windows System ..... 205

CHAPTER 10 MANUALLY ADD THE REGISTRY INFORMATION ..... 206

CHAPTER 11 FAQS ..... 209

11.1 Q1. The m file exported from DBC by 'EcoCAN' can't be used ..... 209

11.2 Q2. Model created by 'EcoCoder_Prj', emulation or code generation error ..... 209

11.3 Q3. 'CAN' module is blank after being configured ..... 210

11.4 Q4. Task scheduling priority ..... 210

11.5 Q5. EcoCoder Loader Pop-up error ..... 210

11.6 Q6. Cancel window pops up after the code is generated ..... 210

11.7 Q7. How to keep a C code project ..... 211

11.8 Q8. How to update application model to be compatible with updated EcoCoder .. 212

**11.9 Q9. Compilation error FAQ ..... 215**
11.9.1 Win10 system failed to add environment variable ..... 215
11.9.2 After adding the environment variable, need to restart MATLAB ..... 215
11.9.3 Added an environment variable with splitter error ..... 215
11.9.4 EcoCoder Loader selects the sysroot path incorrectly ..... 215
11.9.5 Current model path has none-English, spaces, or special characters ..... 216

**11.10 Q10-EcoFlash default DLL ..... 216**

**11.11 Q11-EcoCoder Loader indicates that COMDLG32.OCX is missing ..... 216**

**CHAPTER 12 APPENDIX ..... 218**

# Chapter 1 General Information

## 1.1 About EcoCoder

EcoCoder is an application development tool for control system, that makes it easier for users to develop embedded applications in the Simulink environment. It extends the resources of Simulink and Real-Time Workshop embedded encoders, when generating the necessary code modules, and can automatically configure and optimize code generation process. By encapsulating the basic software libraries into s-functions, developers can do parameters configuration in a graphical way.

EcoCoder helps developers to develop control systems entirely in MATLAB/Simulink environments, without learning hardware knowledge, C code programming or single chip micro-controller settings. Users can do the development in a graphic way, which can help application developers reduce their difficulty to use low level software. Developers can also develop their system applications through CAN bus flashing tool and calibration software tool.

In addition, unlike other similar automated code generation tools among the market, EcoCoder is an automated code generation library based on MATLAB/Simulink. It is very powerful, by simply adding the library modules, it will connect applications software to specific controllers, make the maximum use of Simulink general libraries, and transfer models to any platform which supports Simulink.

## 1.2 Operating System Requirements

Windows7, Windows10

## 1.3 MATLAB Installation Requirements

### 1.3.1 MATLAB R2010b

Mandatory Components:

- MATLAB
- Simulink
- Real-Time Workshop
- Real-Time Workshop Embedded Coder

Highly recommended components to be installed:

- Stateflow
- Stateflow Coder

### 1.3.2 MATLAB R2011a-R2021a

Mandatory Components:

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

Highly recommended components to be installed:

- Stateflow

## 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
- MATLAB R2020a 64-bit
- MATLAB R2020b 64-bit
- MATLAB R2021a 64-bit

Note: If customers are having problem about MATLAB configuration issue, please contact customer service support.

# Chapter 2 EcoCoder Development Environment

## 2.1 Software Installation List

Note: The software marked as "red" must be installed in strict numbering order.

1) Development environment for generating flash-able files

| Main Micro-chip | Development environment |
|---|---|
| Infineon TC27x/TC29xx/TC39xx | 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) |
| S32K146 | S32DS_ARM_Win32_v2018.R1.exe |

2) Compilers for generating DLL files

| Compiler name | Description |
|---|---|
| tdm64-gcc-4.9.2.exe | Includes a GCC compiler that can generate DLL files with calibration, measurement, and program flashing authorizations. Supports all versions of MATLAB to generate DLL files |

3) Program flashing tool: EcoFlash

4) Calibration tool: EcoCAL or INCA

5) CAN Driver

6) Stateflow Compiler

| Compiler | Support |
|---|---|
| C++ Compiler | Supports Stateflow for both 32-bit and 64-bit MATLAB |
| Lcc-win32 | Supports Stateflow for 32-bit MATLAB |
| MinGW-GCC | Support some versions of MATLAB stateflow,. For specific MATLAB support, please check the following table about MATLAB versions and their corresponding supported MinGW versions. |

MATLAB versions and their corresponding supported MinGW versions are as follows:

| MATLAB version | Supported MinGW version |
|---|---|
| MATLAB 2015a or below | Not support |
| MATLAB 2015b | MinGW 4.9.2 (Distributor：TDM-GCC) |
| MATLAB 2016a | MinGW 4.9.2 (Distributor：TDM-GCC) |
| MATLAB 2016b | MinGW 4.9.2(Distributor：TDM-GCC) |
| MATLAB 2017a | MinGW 4.9.2(Distributor：TDM-GCC) |
| MATLAB 2017b | MinGW 5.3(Distributor：TDM-GCC) |
| MATLAB 2018a | MinGW 5.3(Distributor：mingw-w64) |
| MATLAB 2018b | MinGW 6.3(Distributor：mingw-w64) |
| MATLAB 2019a | MinGW 6.3(Distributor：mingw-w64) |
| MATLAB 2019b | MinGW 6.3(Distributor：mingw-w64) |
| MATLAB 2020a | MinGW 6.3(Distributor：mingw-w64) |
| MATLAB 2020b | MinGW 6.3(Distributor：mingw-w64) |
| MATLAB 2021a | MinGW 6.3(Distributor：mingw-w64) |

Compiler options:

To install the compiler that supports MATLAB, you can look up the corresponding compiler for each MATLAB version by entering **Compilers** in the search box MATLAB website:

https://www.mathworks.com/support/requirements/previous-releases.html

If install Microsoft Visual C++, it is recommended to refer to the followings.

Also, when install Microsoft Visual Studio, just install the C++ environment is **enough**.

Don't need to activate the development environment after installation.

MATLAB R2010b-R2016a can use Microsoft Visual C++ 2010 Professional。

MATLAB R2015a-R2018a can use Microsoft Visual C++ 2013 Professional。

MATLAB R2016b-R2019b can use Microsoft Visual C++ 2015 Professional。

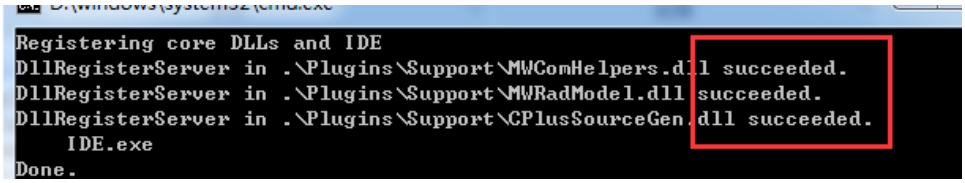
MATLAB R2018a-R2021a can use Microsoft Visual C++ 2017 Family。

7) EcoCoder installation package

## 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.

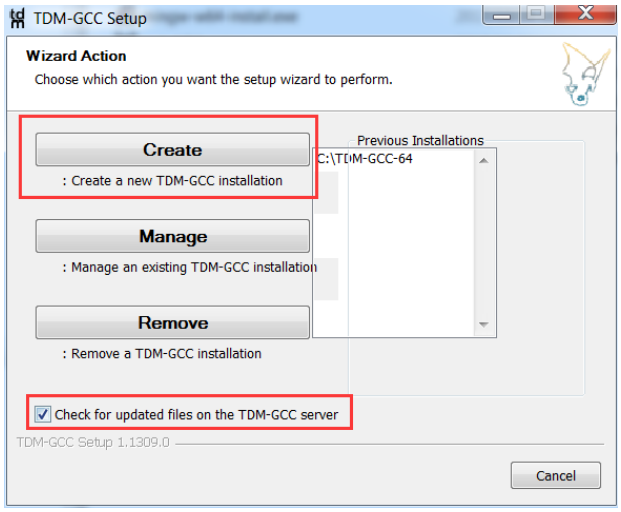



Note: If you follow the step 1 and other versions are called by default during compilation, you will need to uninstall other versions of CodeWarrior.

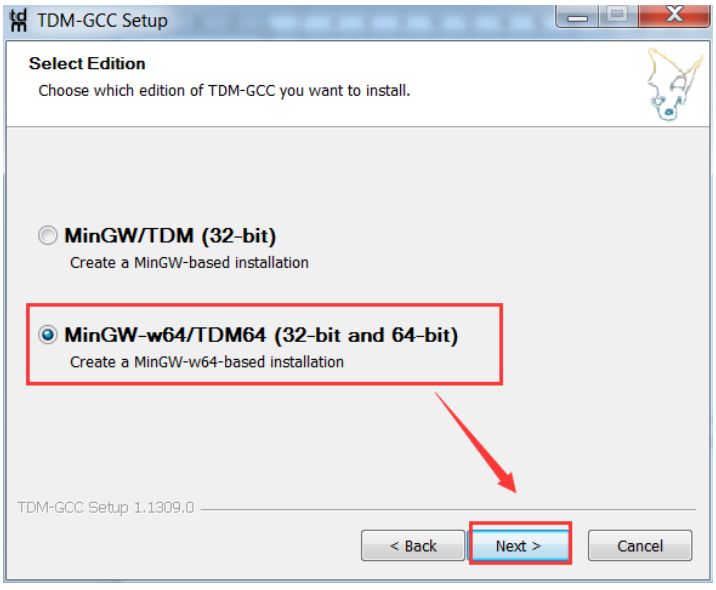
## 2.3 MinGW-GCC Compiler Installation

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:

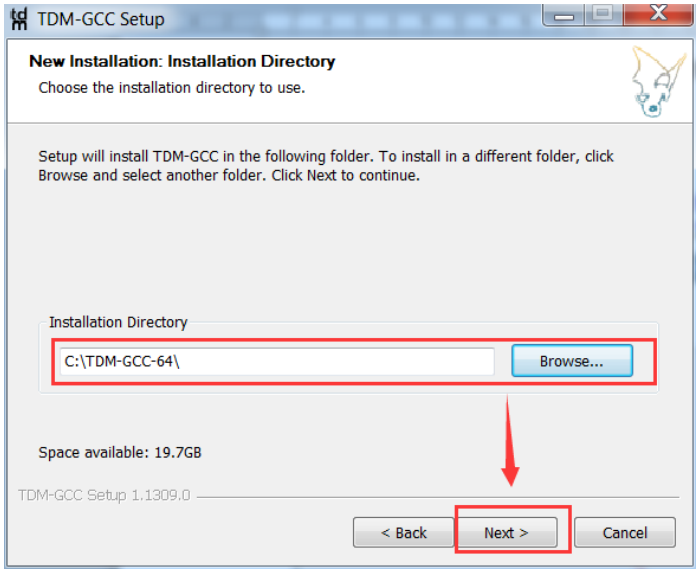
1. First check “Check for updated files on the TDM-GCC server”



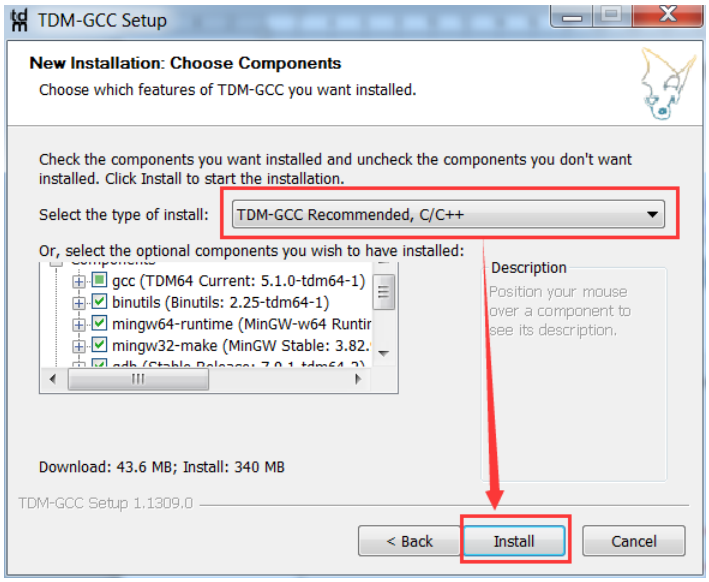

2. Choose MinGW-w64



3. Choose the installation directory

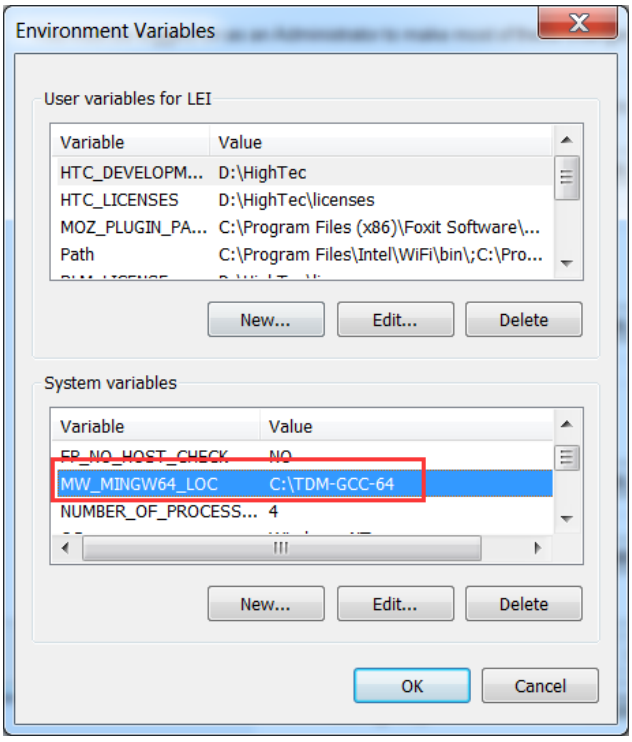



4. Choose TDM-GCC Recommended, C/C++



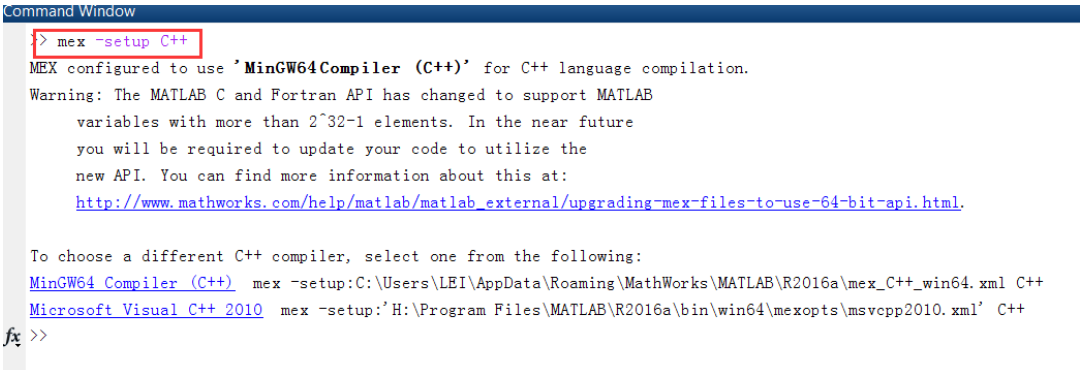
**Note: If user uses MinGW-GCC as the Stateflow complier, proceed to the next step.**

5. Add environment variables




6. Restart or open MATLAB.

7. Enter “mex -setup C++” in MATLAB command window




8. Choose MinGW64 Complier

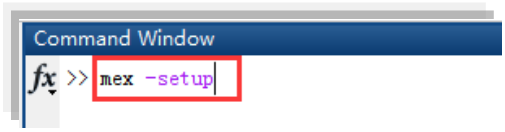
```
>> mex -setup:C:\Users\LEI\AppData\Roaming\MathWorks\MATLAB\R2016a\mex_C++_win64.xml C++
MEX configured to use 'MinGW64 Compiler (C++)' for C++ language compilation.
Warning: The MATLAB C and Fortran API has changed to support MATLAB
     variables with more than 2^32-1 elements. In the near future
     you will be required to update your code to utilize the
     new API. You can find more information about this at:
     http://www.mathworks.com/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.
fx >>
```

## 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 a 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' in MATLAB Command Window.




2. Type 'y' at Command Window, then press Enter

```
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, then press Enter

```
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, then press Enter

```
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 shown up, means installation is successful.

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

Done . . .

********************************************************************************
  Warning: The MATLAB C and Fortran API has changed to support MATLAB
           variables with more than 2^32-1 elements.  In the near future
           you will be required to update your code to utilize the new
           API.  You can find more information about this at:
           http://www.mathworks.com/support/solutions/en/data/1-5C27B9/?solution=1-5C2
           Building with the -largeArrayDims option enables the new API.
********************************************************************************
```

### 2.4.2 Compiler Installation for MATLAB 64-Bit

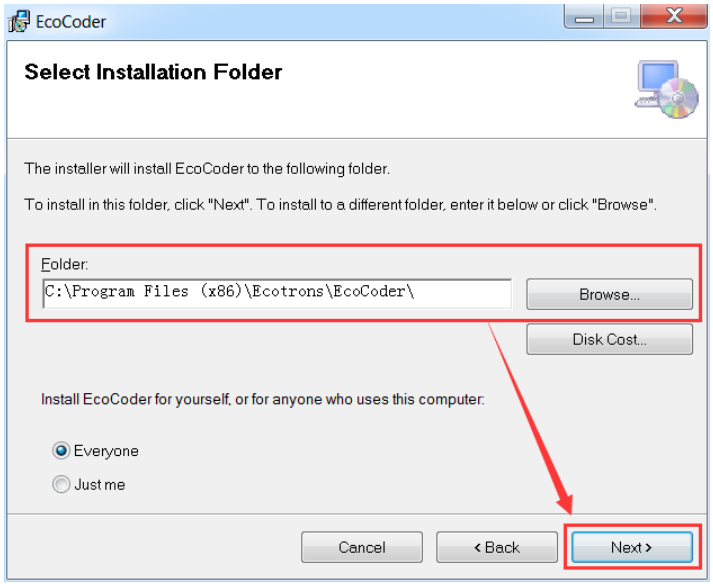
1. Go to the official website of MathWorks.
2. Find the compatible compilers for MATLAB 64-Bit, download and install.
3. Configure the compilers following by previous steps of MATLAB 32-Bit.

## 2.5 EcoCoder Installation

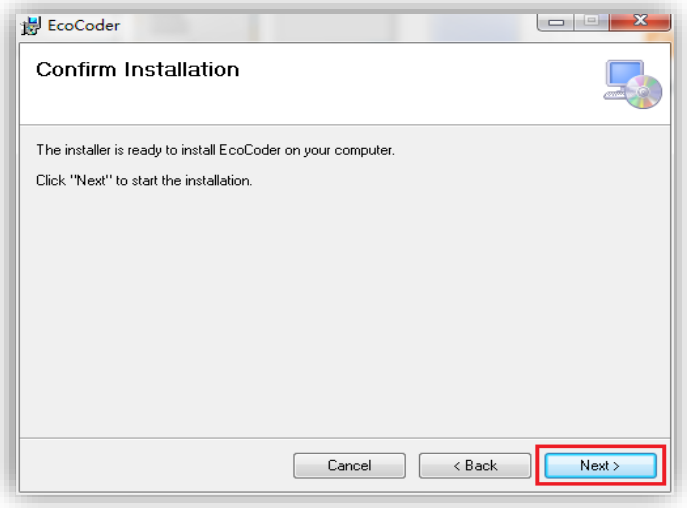
Note: Please keep MATLAB closed during the whole installation process.

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

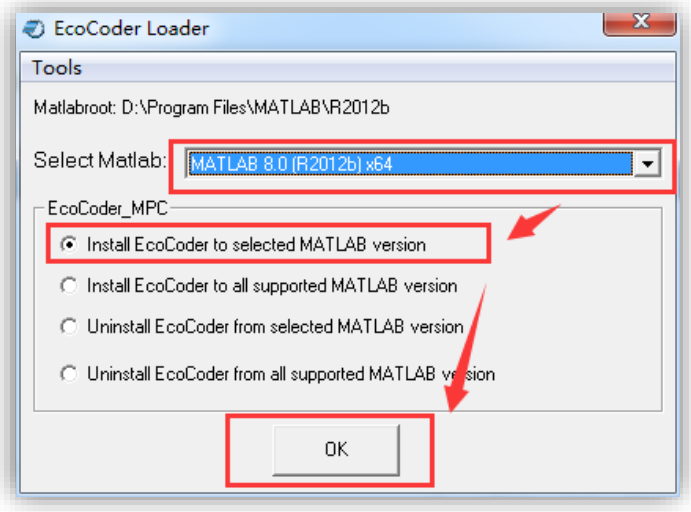

2. Choose installation path, click '*Next*'.



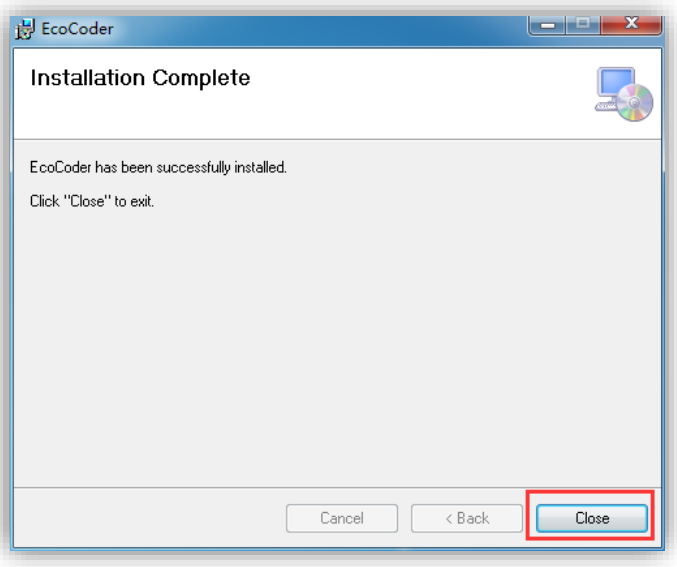

3. Click '*Next*'.




4. Choose the version of MATLAB you want to install EcoCoder to, then select '*Install EcoCoder to selected MATLAB version*', click *'OK'*.




5. When installation is complete, 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 show the message '*EcoCoder has been installed successfully'* as shown in following red box. It indicates that EcoCoder has been successfully installed to MATLAB.

```
=========================================================
    Loading EcoCoder to MATLAB...
    ..............
    Type "EcoCoder_Prj('prjname')" at Command Window to create application
    ..............
    EcoCoder Version: V2.8.4 R22
    EcoCoder Directory: C:\Program Files (x86)\Ecotrons\EcoCoder\V2.8.4R22\EcoCoder_MPC\
    Matlabroot: D:\Program Files\R2014b
    EcoCoder has been installed successfully!
=========================================================
    EcoCoder has been activated successfully!
```

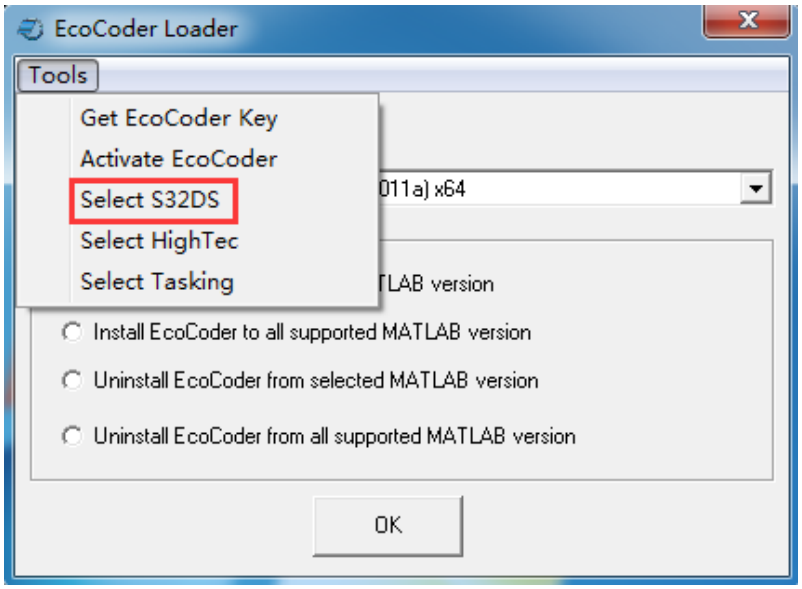
Note: After installation is complete, you need to activate EcoCoder to use it. Please refer to Section 2.10 about how to activate EcoCoder.

## 2.6 Link S32DS_Power_Win32 to EcoCoder

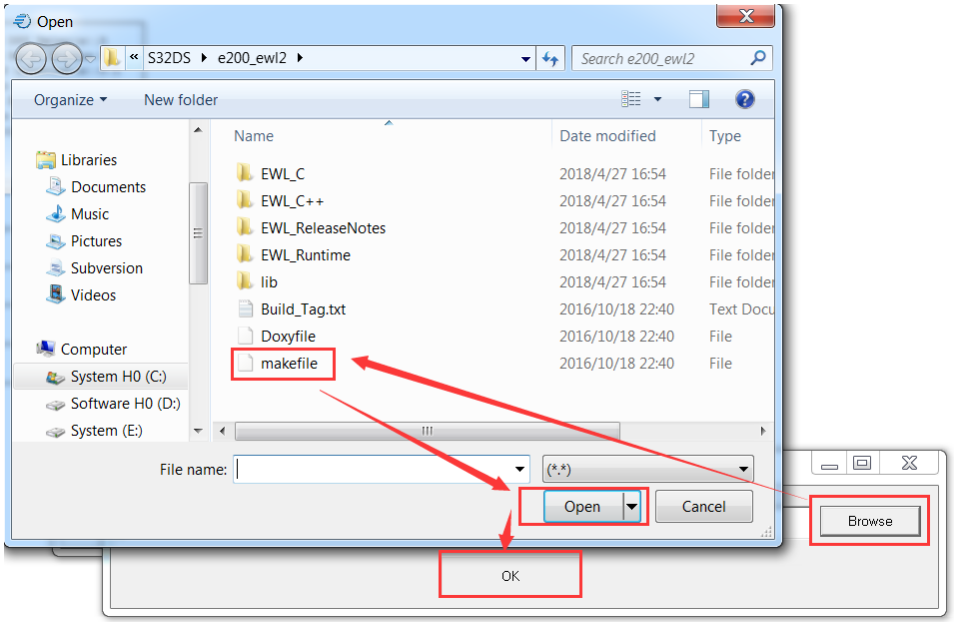
If you need to install S32DS_Power_Win32 development environment, such as S32DS_Power_Win32_v2017.R1_b171019.exe, you can download it from the NXP website.

After installing EcoCoder and S32DS_Power_Win32, you also need to use EcoCoder Loader to select the path of makefile, under e200_ewl2 directory, and add the path of powerpc-eabivle-gcc.exe and the path of make.exe into the environment variables. Shown as below:

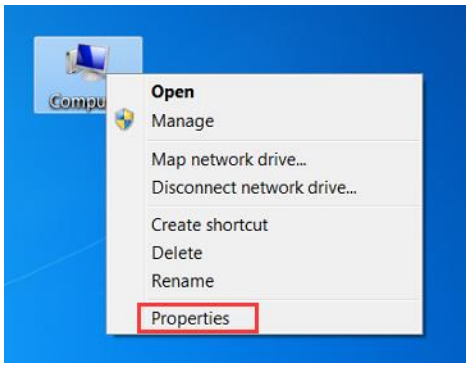
1. Open EcoCoder, choose Tools > Select S32DS



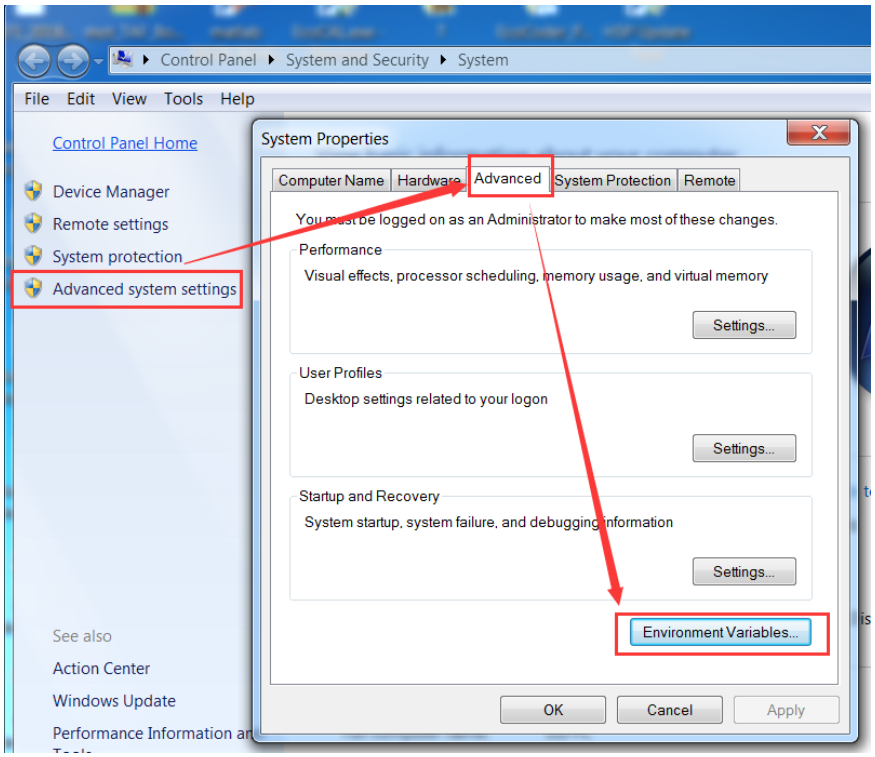

2. Click Browse, choose *makefile* under e200_ew12 directory, for example, the full path is: C：\NXP\S32DS_Power_v2017.R1\S32DS\e200_ewl2\makefile



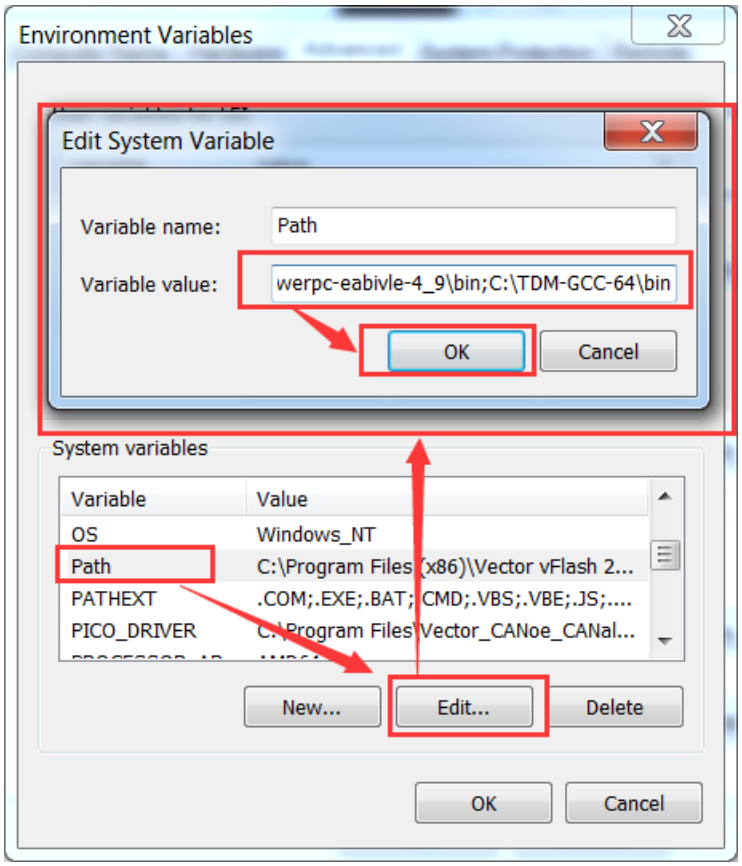
3. Right click on Computer, then click on properties




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. For example, at the end of the Variable value, add: C:\NXP\S32DS_Power_v2017.R1\utils\msys32\usr\bin;C:\NXP\S32DS_Power_v2017.R1\Cross_Tools\powerpc-eabivle-4_9\bin



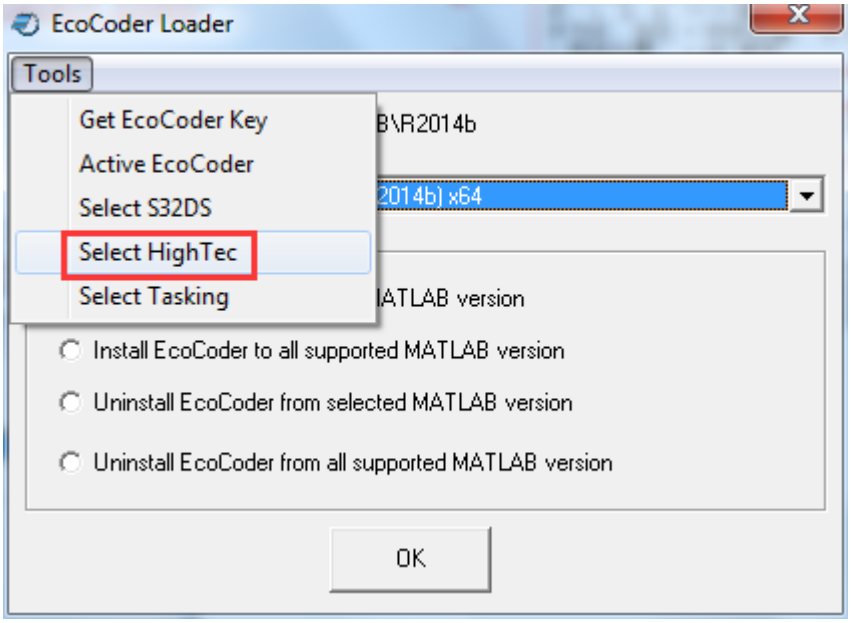

Note: After adding path to environment variables, need to restart MATLAB

## 2.7 Link HighTec TriCore Tool Chain to EcoCoder

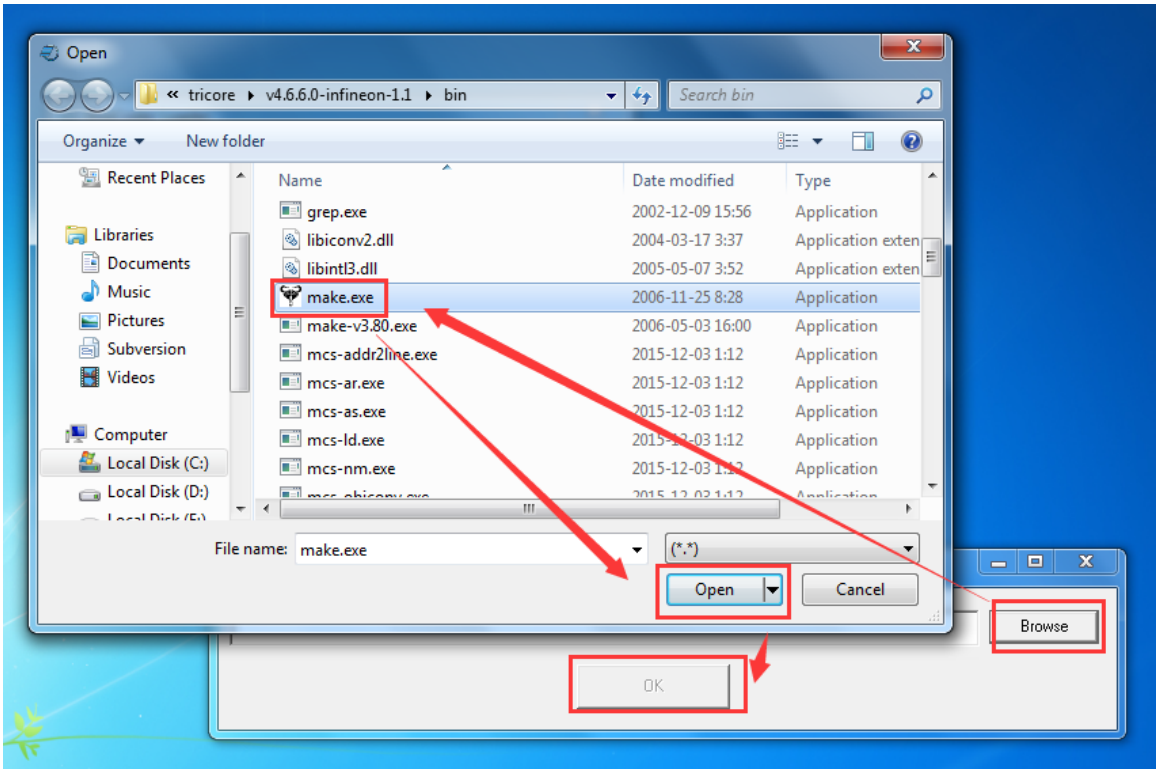
If you need to install the HighTec TriCore Tool Chain development environment, you can download it from the HighTec or Infiniton website.

After installing the HighTec TriCore Tool Chain, you also need to use the EcoCoder Loader to select the path of make .exe in the HighTec installation directory, as follows:

1) Open EcoCoder Loader and select Tools-> Select HighTec




2) Click Browne and select make .exe in the toolchains folder under the HEIGHTEC installation directory, for example, the path is C:\HIGHTEC\toolchains\tricore\v4.6.6.0-infineon-1.1\bin\make.exe



## 2.8 Link CS+ to EcoCoder

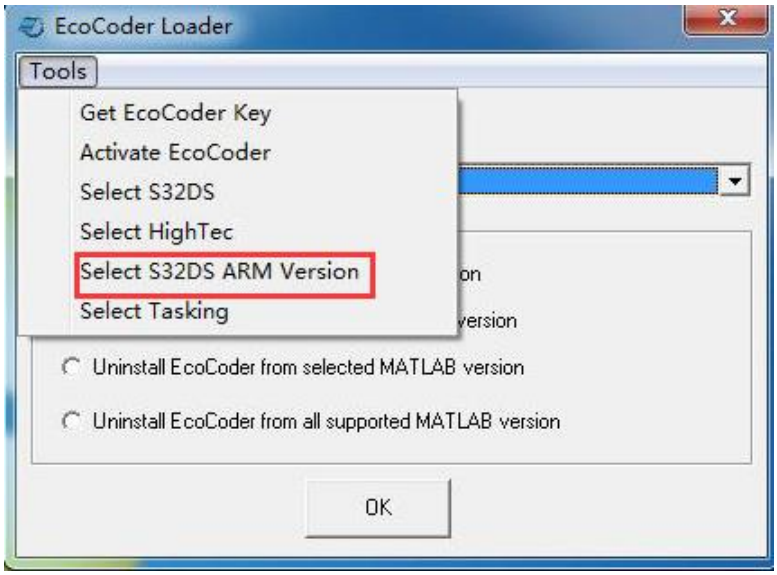
If you need to install CS+, such as CSPlus_CC_Package_V70000.exe. After installation, you need to add the directory where CubeSuite+.exe is located to the system environment variable Path, and you need to restart MATLAB after adding the environment variable. The method of adding environment variables can refer to the section for linking S32DS_Power_Win32 with EcoCoder.

## 2.9 Link S32DS_ARM_Win32 to EcoCoder

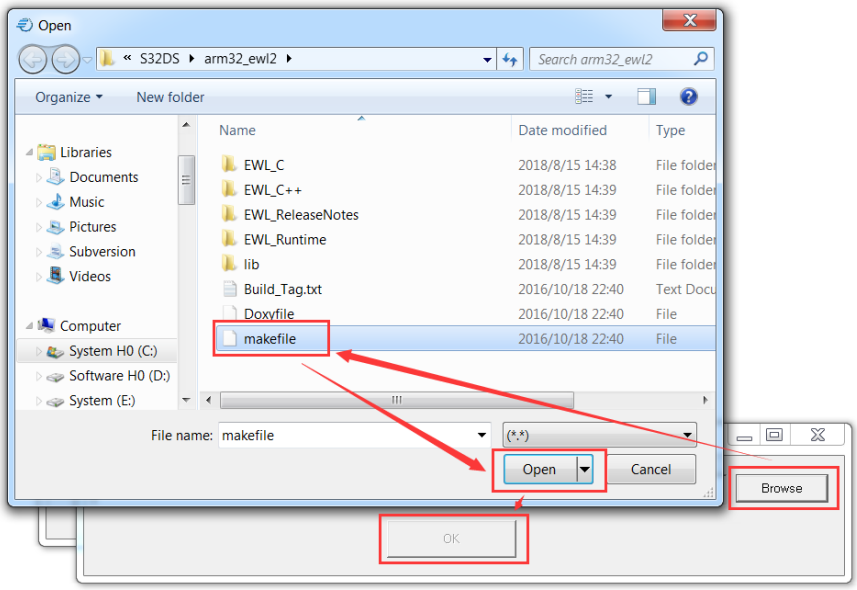
If you need to install S32DS_ARM_Win32 development environment, such as S32DS_ARM_Win32_v2018. R1 .exe, download it from the NXP official website.

After installing EcoCoder and S32DS_ARM_Win32, you also need to use the EcoCoder Loader to select the path of makefile under arm32_ewl2 directory, and add the path of arm-none-eabi-gcc .exe and the path of make.exe to environment variables, shown as below:

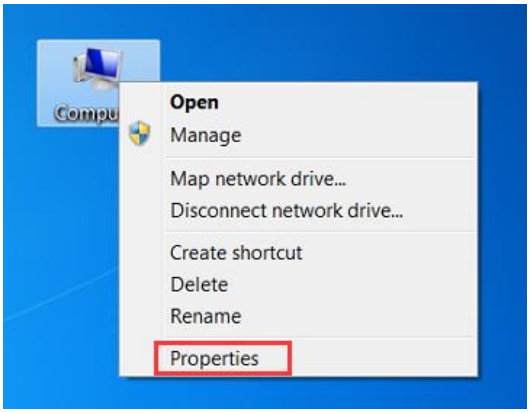
1) Open ecoCoder Loader and select Tools->Select S32DS ARM Version



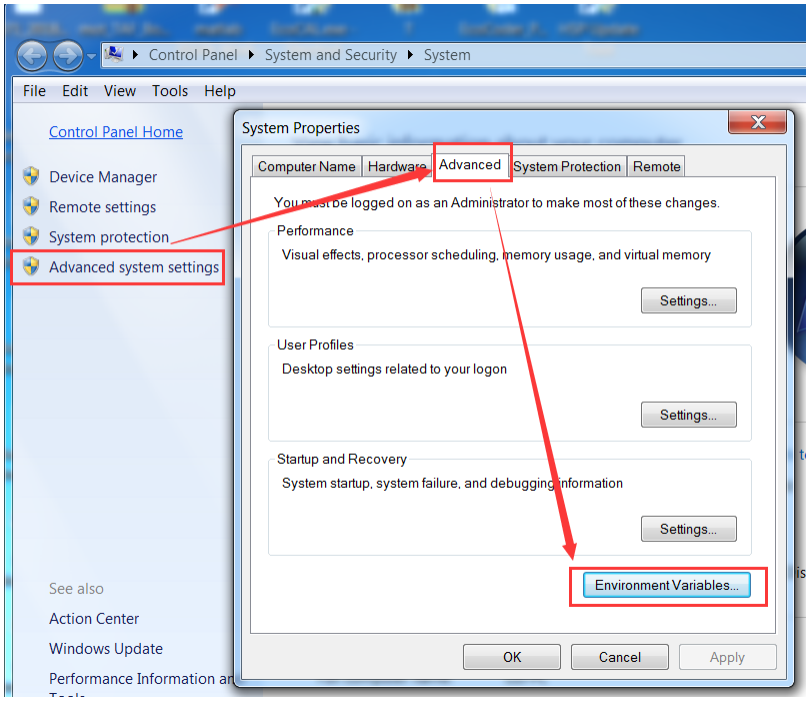

2) Click Browse and select makefile under arm32_ewl2 directory, e.g. the path is C:\NXP\S32DS_ARM_v2018.R1\S32DS\arm32_ewl2\makefile



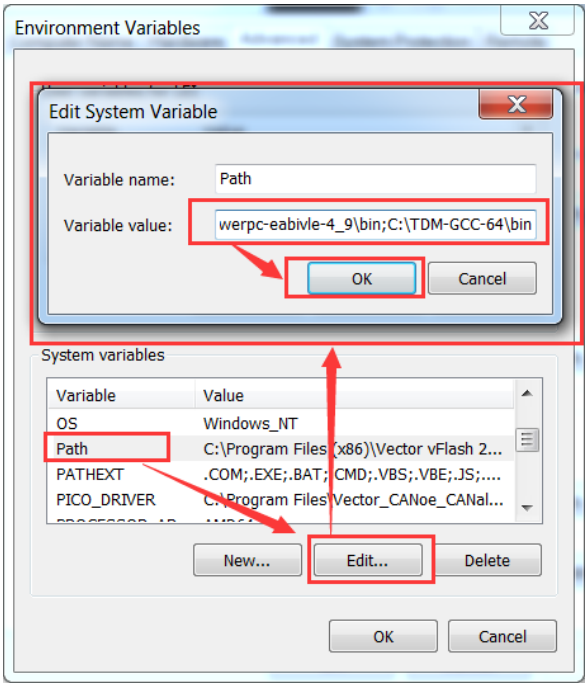

3) Right click Computer, select Properties




4) Select Advanced system settings->Advanced->Environment Variables




5) Add the path of arm-none-eabi-gcc.exe and the path of make.exe to environment variables, separated by semicolon, for example, at the end of Variable value, add: C:\NXP\S32DS_ARM_v2018.R1\Cross_Tools\gcc-6.3-arm32-eabi\bin; C:\NXP\S32DS_ARM_v2018.R1\utils\msys32\usr\bin;




Note: need to restart MATLAB after adding path to environment variables

## 2.10 Activate EcoCoder

EcoCoder requires a license file or USB dongle before using.

### 2.10.1 Active EcoCoder by USD Dongle

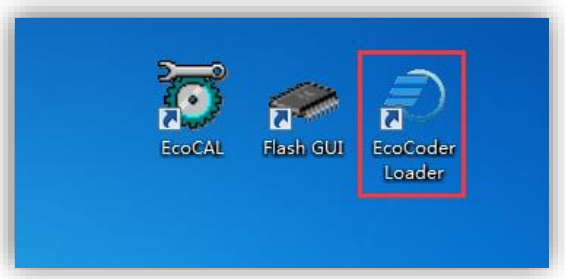
Connect the USB dongle device to the computer, you can automatically activate EcoCoder. After successfully activation, MATLAB will show the following red box prompt message:

```
============================================================
    Loading EcoCoder to MATLAB...
    ..............
    Type "EcoCoder_Prj('prjname')" at Command Window to create application
    ..............
    EcoCoder Version: V2.8.4 R22
    EcoCoder Directory: C:\Program Files (x86)\Ecotrons\EcoCoder\V2.8.4R22\EcoCoder_MPC\
    Matlabroot: D:\Program Files\R2014b
    EcoCoder has been installed successfully!
============================================================
   EcoCoder has been activated successfully!
```

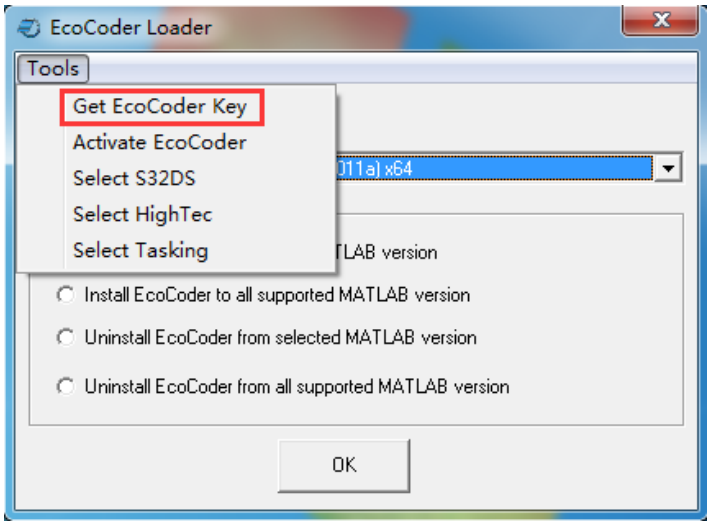
### 2.10.2 Active EcoCoder by license file

1. Get Key file

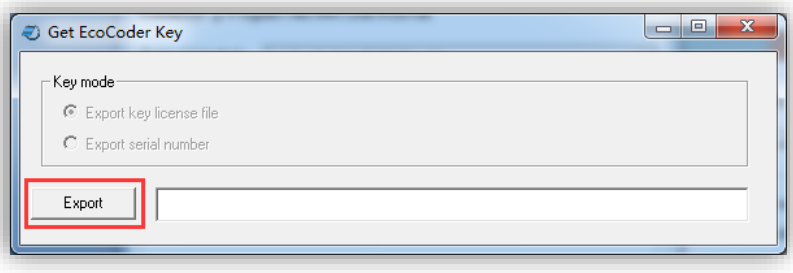
    1) Double-click “EcoCoder Loader” on the desktop.



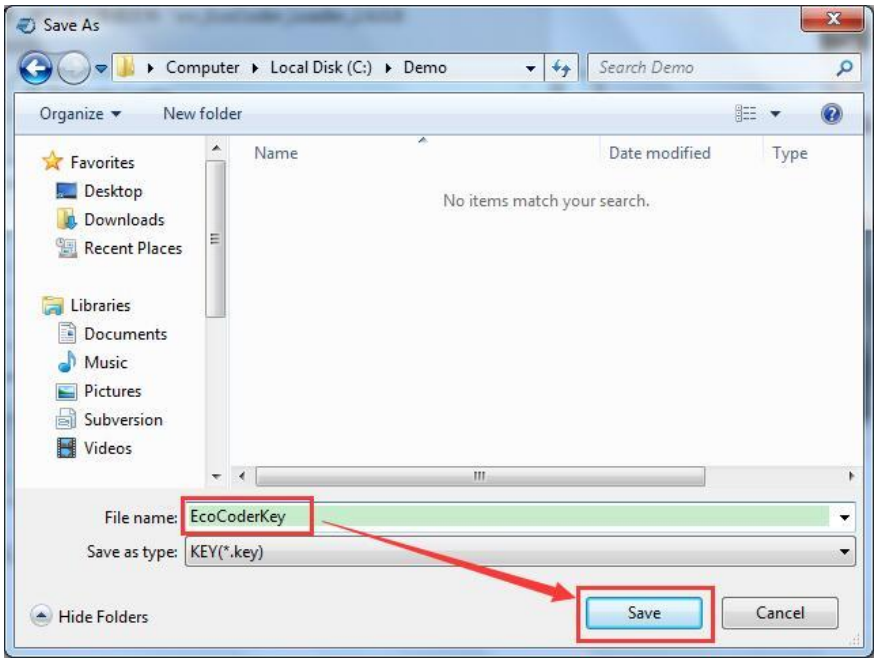

    2) Select Tools → Get EcoCoder Key.




3) Click ‘Export’.




4) The pop-up window is as follows, name the file and save it. For example, EcoCoderKey




5) Send this “EcoCoderKey.key” to Ecotron customer support or info@ecotron.ai to get a license file later.

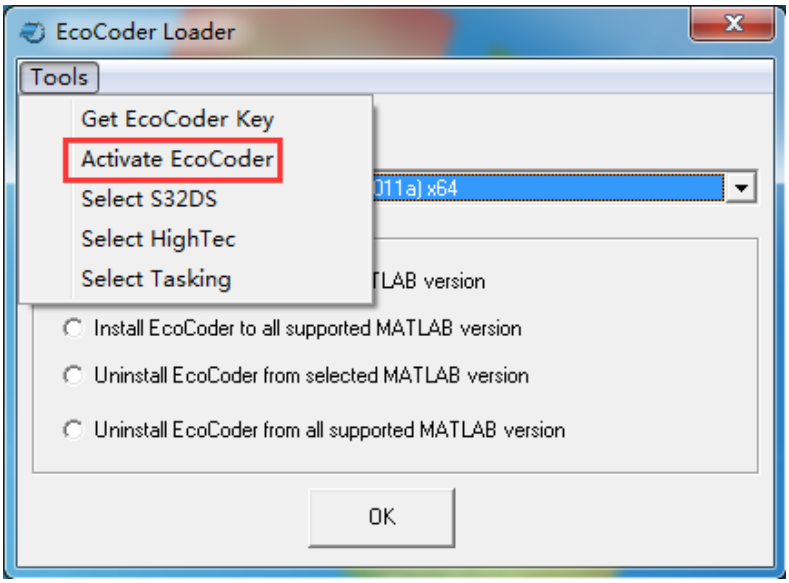
2. Active EcoCoder using license file

Note: please close all the MATLAB windows when active EcoCoder

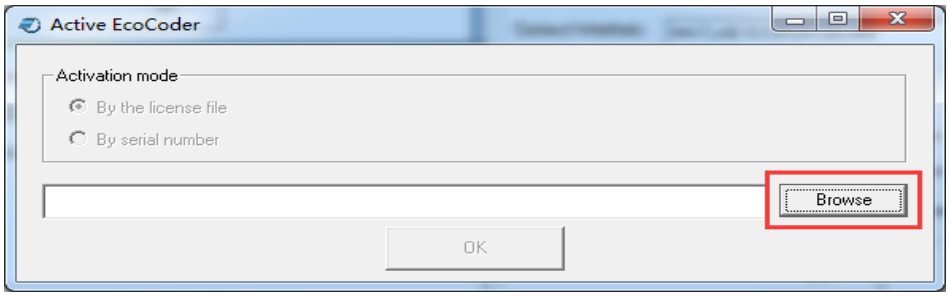
1) Double click “EcoCoder Loader” on desktop


2) If customer only want to activate a specific version of MATLAB, please select the appropriate Matlab version in the drop-down menu.

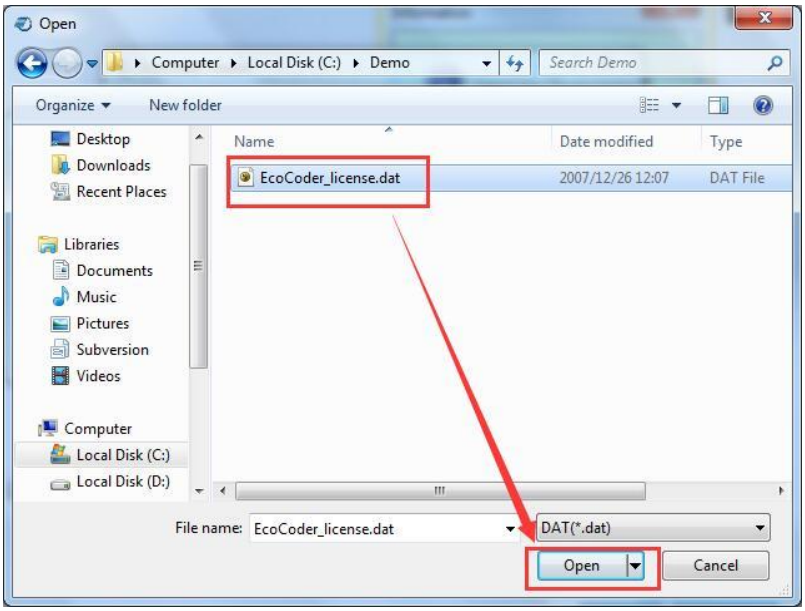
3) Select Tools->Activate EcoCoder



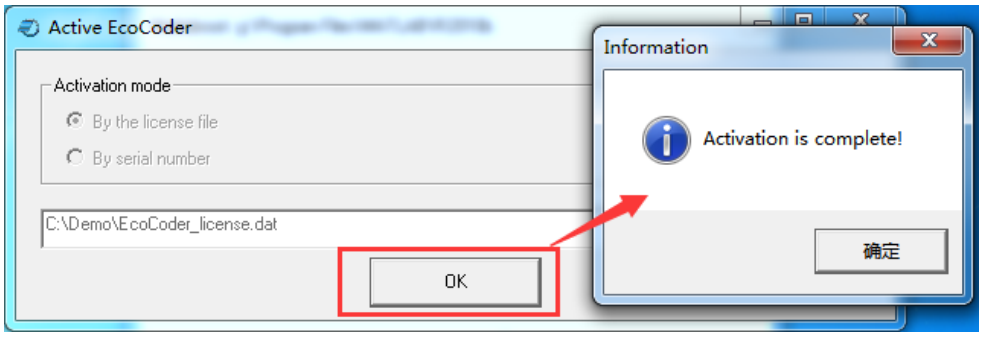

4) Click “Browse”




5) Select the license file you got from Ecotron support, such as "EcoCoder_license.dat", then click Open



6) Click "OK". If the following window pops up, it means activation is successful.




7) After open MATLAB, if it shows the red box as below, means customer can use EcoCoder now.

```
============================================================
    Loading EcoCoder to MATLAB...
    ..............
    Type "EcoCoder_Prj('prjname')" at Command Window to create application
    ..............
    EcoCoder Version: V2.8.4 R22
    EcoCoder Directory: C:\Program Files (x86)\Ecotrons\EcoCoder\V2.8.4R22\EcoCoder_MPC\
    Matlabroot: D:\Program Files\R2014b
    EcoCoder has been installed successfully!
============================================================
    EcoCoder has been activated successfully!
```

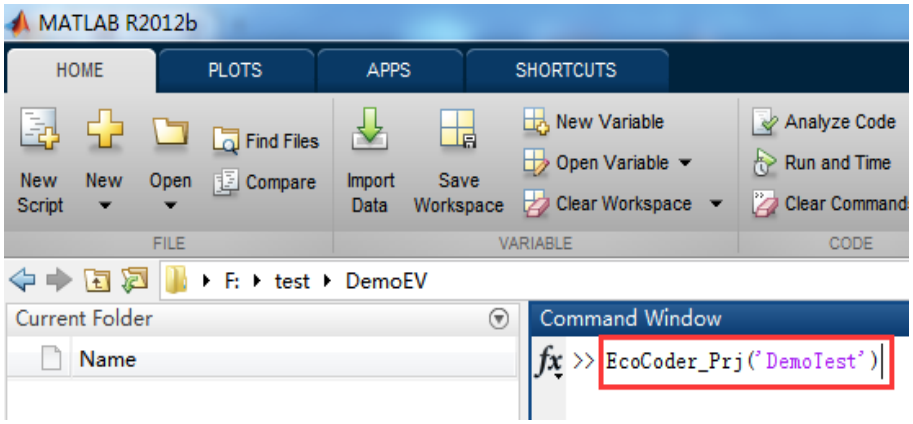
## 2.11 Quick start for building application model

If customer doesn't have an existing Simulink model, then they can use EcoCoder to create a quick model, then develop their control logic based on it.

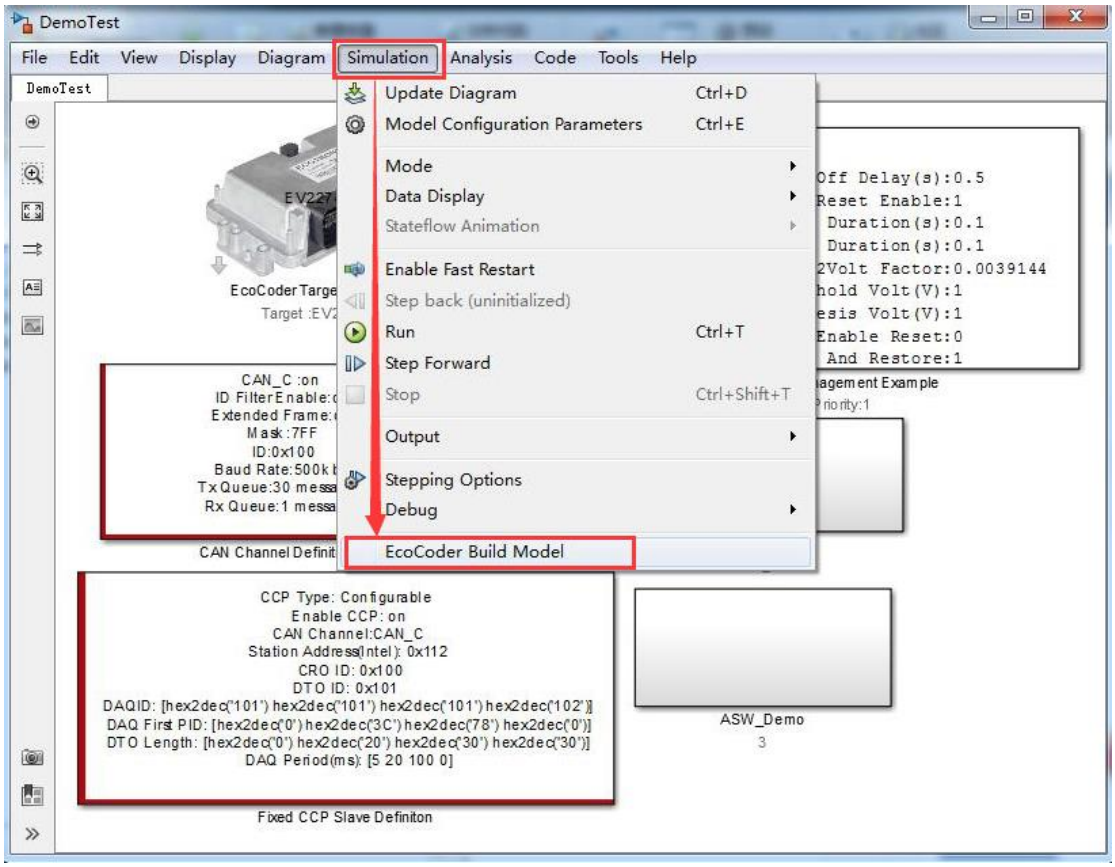
There're two ways to create this quick model, first way is by typing commands in the command window, the other way is to create a new model.

1. Create quick model by commands:

1) Change the path to a folder except for MATLAB installation directory.
2) Enter the command "EcoCoder_Prj ('DemoTest')" in the command window and press Enter, then a model named "DemoTest" and its related m files will be generated.




3) Select "Simulation->EcoCoder Build Model" in menu bar to generate A2L files and executables (mot files or hex files).




2. Create quick model by build a new model

Quick model can be created by dragging EcoCoder Target Definition block into a new model.

Note:

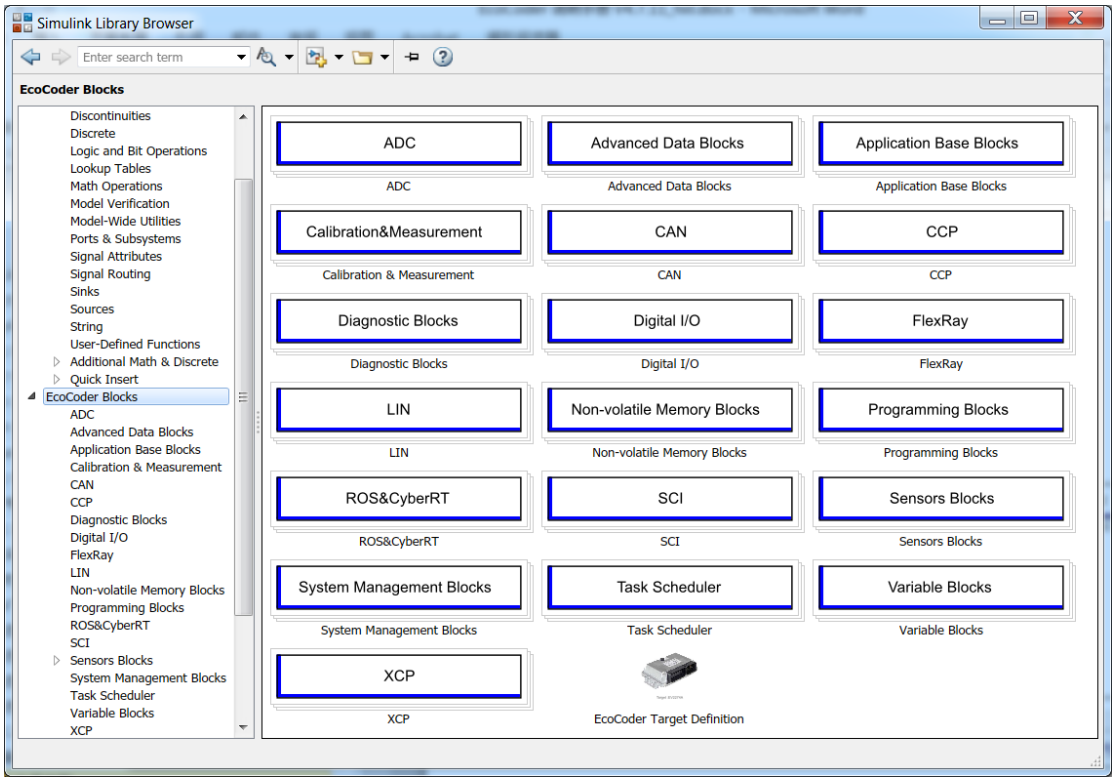
1. If some window pops up during compilation, don't click them, they will be

automatically closed after the compilation is completed.

2. The quick model directory cannot be MATLAB installation directory. Otherwise, an error will pop up and the compilation process will fail.

# Chapter 3 EcoCoder Library

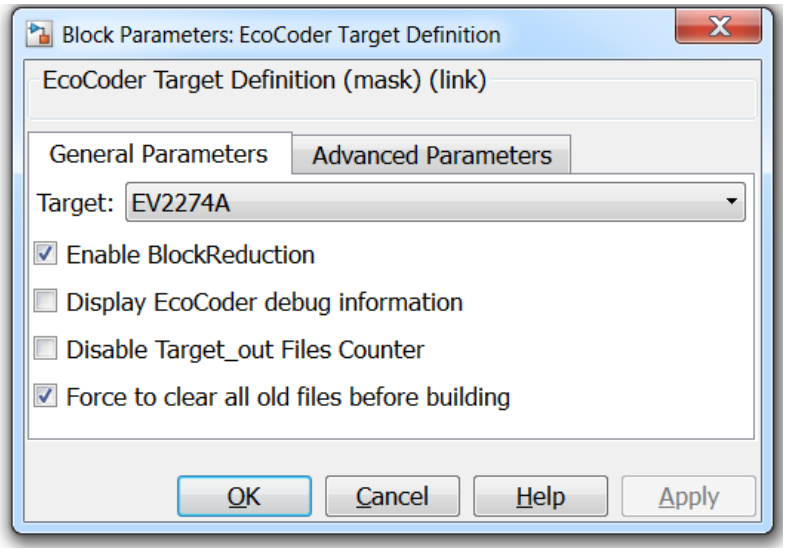
The EcoCoder library is an add-on library in Simulink. EcoCoder library mainly provides interfaces for application software to handle I/Os, VCU power, communication, and calibration / measurement setup, etc.




## 3.1 EcoCoder Target Definition

EcoCoder Target Definition is used to select the target controller, and this block must be added, otherwise an error will occur during compilation. If drag this block to a new model, the model configuration will be performed automatically.


EcoCoder Target Definition
Target :EV2274A




Parameters:

**General Parameters:**

1) Target: Select the target hardware.

2) Enable BlockReduction: Enables the module de-redundancy, if enabled, the C code for related redundant model will not be generated.

3) Display EcoCoder debug information: Displays debugging information, if enabled, some debugging information will be generated when compiling.

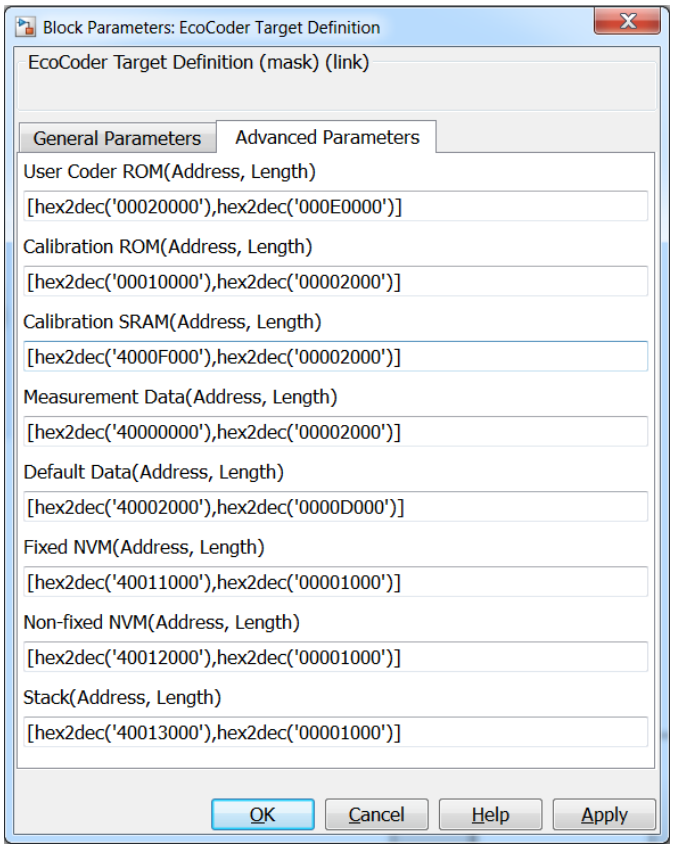
4) Disable Target_out Files Counter: When unchecked, multiple builds of the same model will cause the A2L and program files generated in the Target_out folder to have a count suffix without overwriting the old file; when checked, these files have no count suffix and always overwrite the old file.

5) Force to clear all old files before building: When checked, each time Build, EcoCoder will first intervene to directly delete the XXX_xxx_trw (where XXX is the model's name and xxx is the system target file name) and slprj folders under the generated directory; when not checked, each time Build these folders are controlled by Simulink and EcoCoder for partial intervention. If you have a reference model and do not want the

Rebuild option to be Always, you need to uncheck this option.

**Advanced Parameters:**

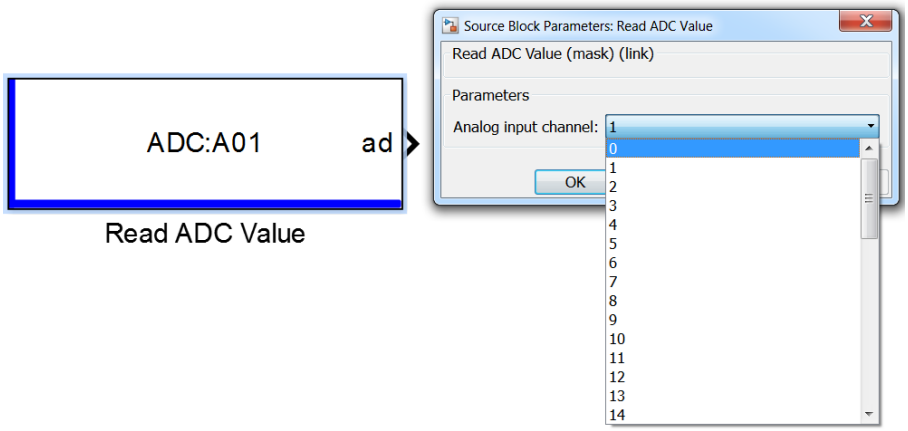
1) The tab can be configured for ROM and RAM part of the microcontroller, if there is a need to adjust, it is recommended to contact technical support.




## 3.2 ADC Analog Inputs

### 3.2.1 Read ADC Value

The module is used to get the original value (AD value) of the analog voltage input.




Parameters:

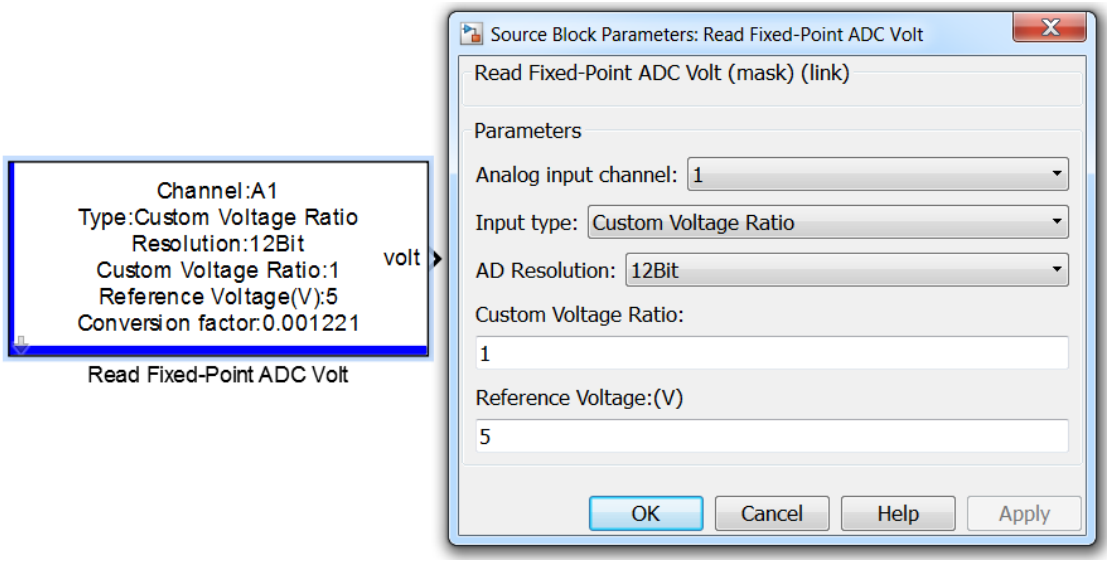
1) Analog input channel: Analog channel selection.

Output:

1) ad: ad value of specific channel.

### 3.2.2 Read Fixed-Point ADC Volt

This module is used to read the voltage value of the analog input channel signal. Signals are fixed-point types and require Fixed-point tool pre-installed.




**Parameters:**

1) Analog input channel: Analog channel selection.

2) Input type: Channel type, is a voltage type input, support "Volt_0_5V", "Volt_0_12V", "Volt_0_24V" and "Customer Voltage Ratio" four types.

3) AD Resolution: Analog input channel accuracy, support 10-bit accuracy or 12-bit accuracy.

4) Custom Voltage Ratio: Custom voltage ratio. Changes are supported only if "Customer Voltage Ratio" is selected for Input type. Its values are determined by the parameters in the controller technical manual "Table 2.2.1" and are calculated as follows:

Custom Voltage Ratio = (Divided Voltage Resistor + Serial Resistor) / Divided Voltage Resistor.

5) Reference Voltage: Reference voltage value, the default is 5V.

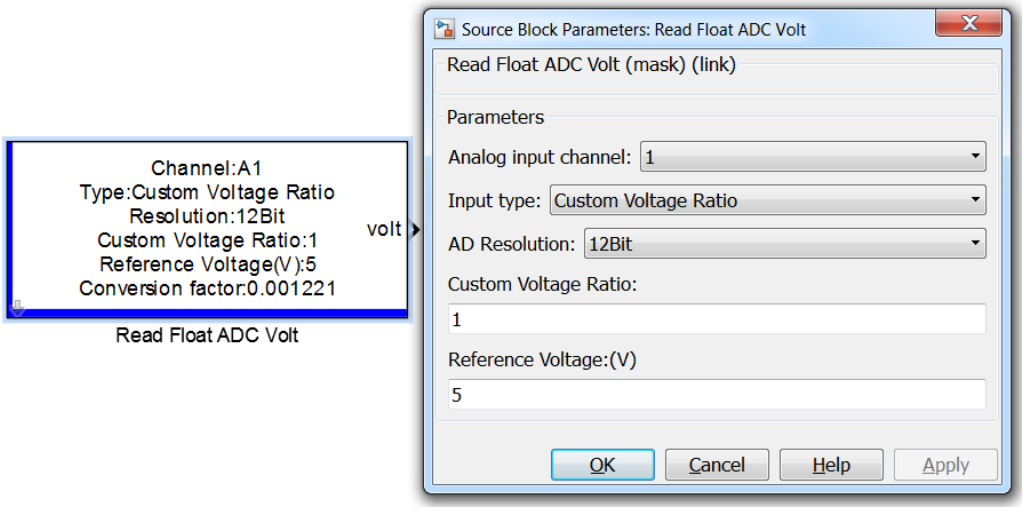
**Output:**

1) volt: Channel voltage value, unit of V, fixed-point data type.

Note: Each channel has a dedicated configuration, please refer to VCU datasheet and select correct channel settings.

### 3.2.3 Read Float ADC Volt

This module is used to read the voltage value of the analog input channel signal.




**Parameters:**

1) Analog input channel: Analog channel selection.

2) Input type: Channel type, voltage type input, support "Volt_0_5V", "Volt_0_12V", "Volt_0_24V" and "Customer Voltage Ratio" four types.

3) AD Resolution: Analog input channel accuracy, support 10-bit accuracy or 12-bit accuracy.

4) Custom Voltage Ratio: Custom voltage ratio. Changes are supported only if "Customer Voltage Ratio" is selected for Input type. Its values are determined by the parameters in the controller technical manual "Table 2.2.1" and are calculated as follows:

Custom Voltage Ratio= (Divided Voltage Resistor + Serial Resistor) / Divided Voltage Resistor.

5) Reference Voltage: Reference voltage value, the default is 5V.

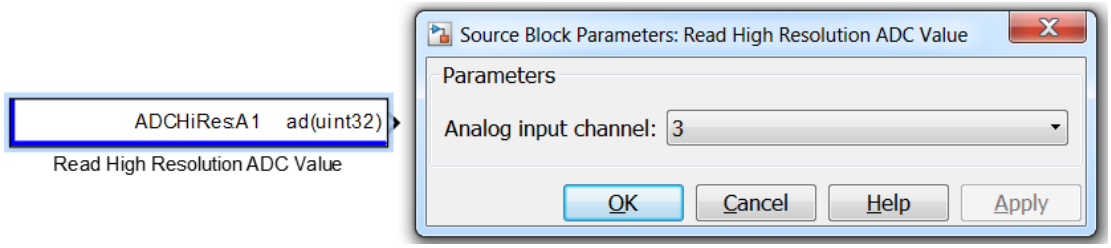
**Output:**

1) volt: Channel voltage value in V, data type is single.

Note: Each channel has a dedicated configuration, please refer to the VCU datasheet and select the correct channel settings.

### 3.2.4 Read High Resolution ADC Value

The module is used to read high-precision AD values that is over 2^16




Parameters:

1) Analog input channel: Channel selection

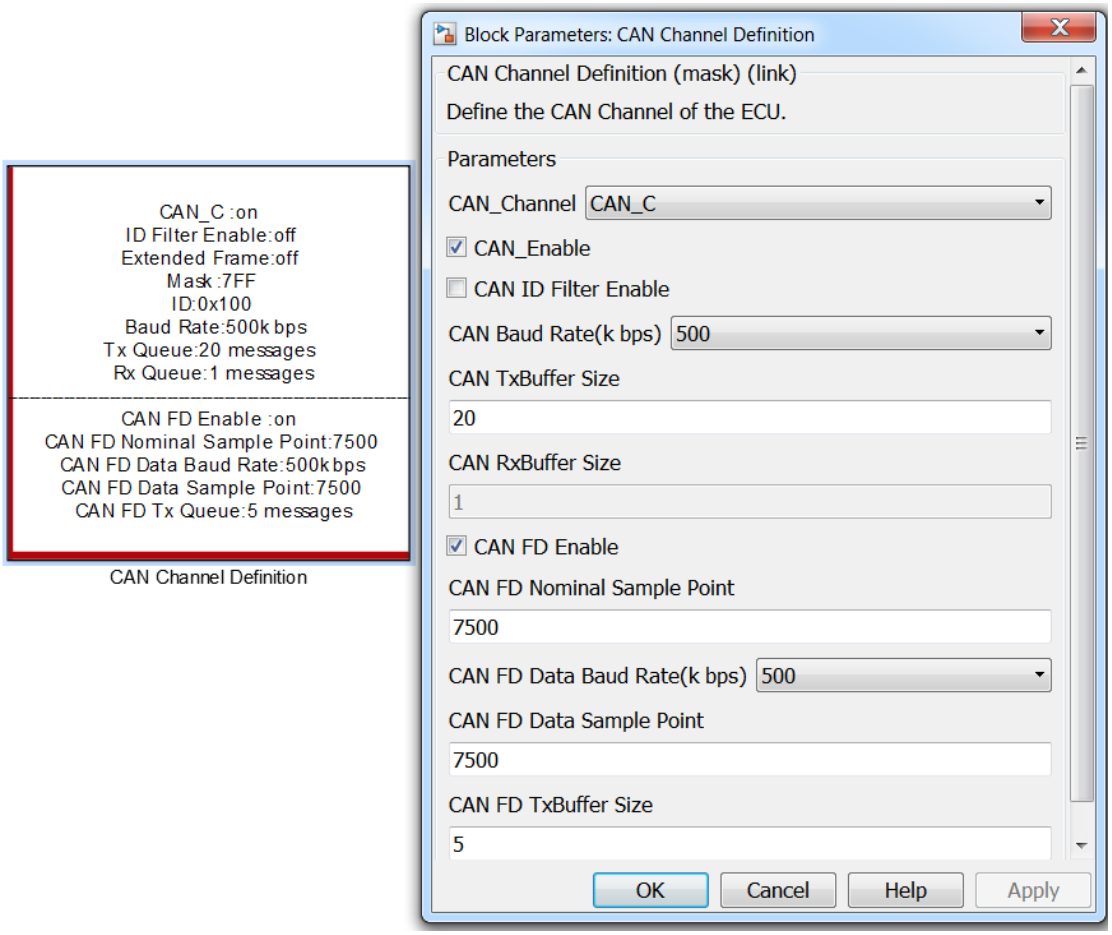
Output:

1) ad(uint32): ad value of specific channel, type uint32.

## 3.3 CAN Communication

### 3.3.1 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: Receive extended frame enable. Receive extended frame if enable, receive standard frame if not enable.

   Note: If CAN ID Filter Enable is not enabled, the setting can be ignored and all types will be received.

5) CAN ID Mask (uint32 Hex): The CAN ID filtering mask, whose value is a hexadecimal number, ignores the bit set by the ID if a bit is 0. If it is 1, the bit must be the same as the bit of the ID to be received. To better understand this feature, refer to Example 1.

   Note: This option needs to be used with the CAN ID Filter (uint32 Hex).

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: total 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%

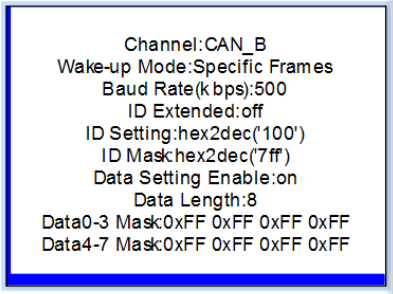
14) CAN FD TxBuffer Size: CAN FD transmit buffer; minimum value is 2

**Example 1:**

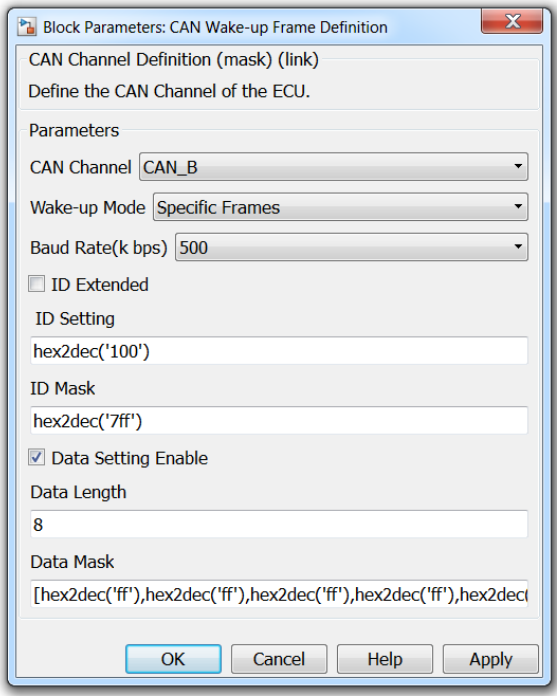
If a bit of CANID Mask is 0, the CAN BUS can receive both ID with this bit is 0 and the ID with this bit is 1. If a bit of CAN ID Mask is 1, then CAN BUS can only receive IDs with the same bit value as the CAN ID Filter. Assuming that the filtering mask is 0x700(0b0000011100000000), filter ID is 0x111(0b0000000100010001), then VCU will receive the ID of 0bxxxxx001xxxxxxxx, here x can be either 1 or 0.

### 3.3.2 CAN Wake-up Frame Definition

This module is used to wake up the controller via CAN message.




CAN Wake-up Frame Definition




**Parameters:**

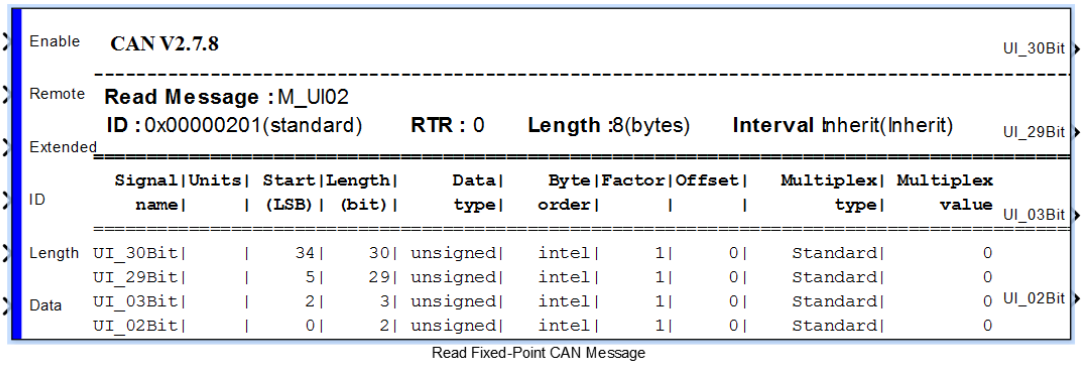
1) CAN Channel: Selecting CAN channel.
2) Wake-up Mode: Setting the wake-up mode, including Disable (disable CAN wake up function), All Frames (any frame on the specified bus can wake VCU up), and Specific Frames (User specify the frame that can wake up the VCU).
3) Baud Rate: CAN baud rate set up.
4) ID Extended: Receives extended frame if enabled. Receive standard frame if not enabled.
5) ID Setting: ID setting for specific frame.
6) ID Mask: Specific frame ID mask settings, need to be used with ID Settings. Please refer to previous section for CAN ID Mask.
7) Data Setting Enable: If enabled, requires the setting data for wake-up frame, if not enabled all data can wake up.

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.

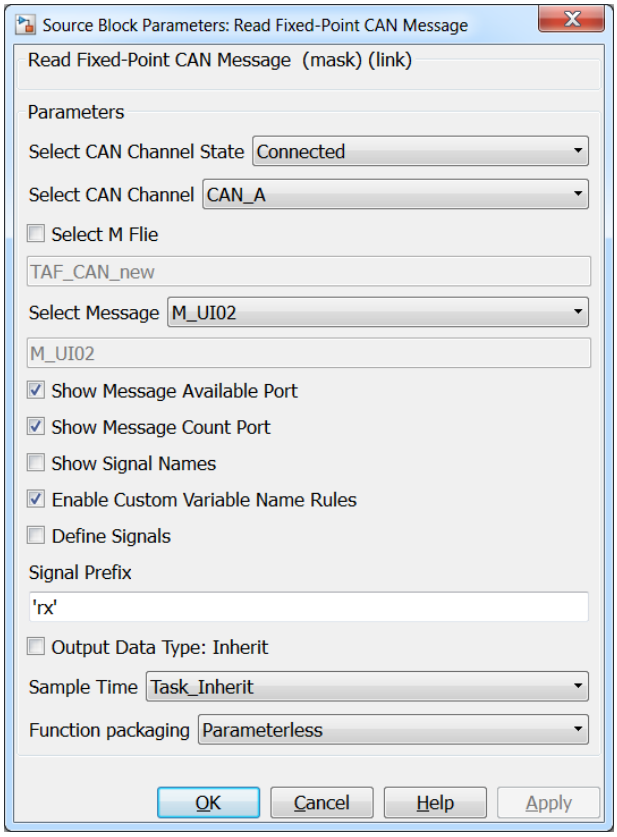
### 3.3.3 Read Fixed-Point CAN Message

This module is used to receive CAN messages. Translate CAN messages into signals, in fixed-point types, needs Fixed-point tool pre-installed. This module is not directly compatible with DBC, but can load m files that is converted by DBC.




| Signal name | Units | Start (LSB) | Length (bit) | Data type | Byte order | Factor | Offset | Multiplex type | Multiplex value |
|---|---|---|---|---|---|---|---|---|---|
| UI_30Bit | | 34 | 30 | unsigned | intel | 1 | 0 | Standard | 0 |
| UI_29Bit | | 5 | 29 | unsigned | intel | 1 | 0 | Standard | 0 |
| UI_03Bit | | 2 | 3 | unsigned | intel | 1 | 0 | Standard | 0 |
| UI_02Bit | | 0 | 2 | unsigned | intel | 1 | 0 | Standard | 0 |

Read Fixed-Point CAN Message




**Parameters:**

1) Select CAN Channel State: Select the channel state, including Connected, Disconnected, and DisconnectedOnlyData options. When selected connected, there is no input port, and the message information is read directly from the selected channel; when Disconnected is selected, the module will appear input ports, and the message information will be obtained from these ports; when selecting DisconnectedOnlyData, it is only used for data conversion which is not related to CAN.

2) Select CAN Channel: CAN channel selection.

3) Select M File: Select the m file converted by dbc conversion tool, need to add the m file to the MATLAB path. After each selection, when clicking "OK" and "Apply", you need to double-click the module again to make the Message selection.

4) Select Message: Select a CAN message.

5) Show Message Available Port: Shows whether message data is received, and 1

represents received data.

6) Show Message Count Port: Message counter, each time a new message is received, the counter is incremented by 1.

7) Show Signal Names: When enabled, the name of the signal is displayed on the output signal line.

8) Enable Custom Variable Name Rules: Enables a custom variable naming convention that removes the second underscore from the variable name and the first letter after the second underscore is uppercase.

9) Define Signals: When enabled, the variables on the signal line are defined as measurements. When enabled, Show Signal Names must be enabled.

10) Signal prefix: The variable prefix on the signal line.

11) Output data type: When checked, the data type of the signal is inherited backwards. When not checked, the signal type is automatically defined as the fixed-point data type with calibration.

12) Sample time: Module simulation time selection.

13) Function packaging: C code function settings, divided into two types, Parameterless and Parameterized, in which Parameterless is a parameterless type function, code execution efficiency is high, while Parameterized is a function with parameter type, supporting the case that the C code corresponding to the module input and output signals is a local variable, preventing compilation errors.

**Input:**

When the channel status is Disconnected, the input ports are as follows. No input port when Connected is selected.

1) Enable：

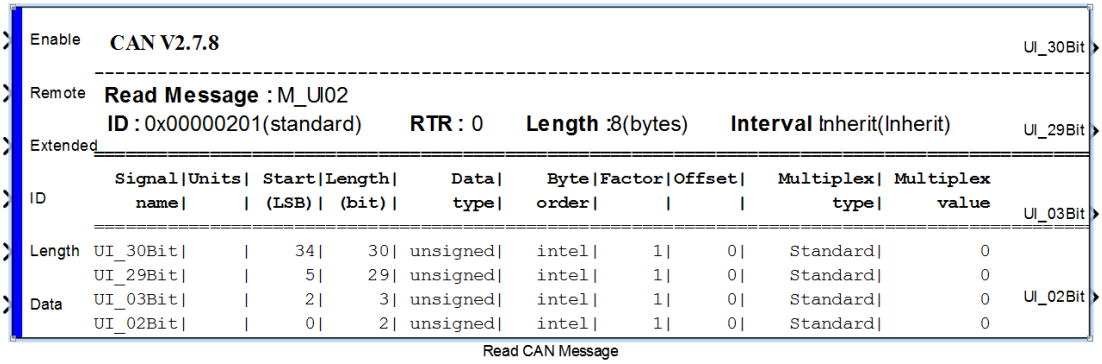
**Output:**

1) For each signal after unpacking, the value of the signal is the physical value of the actual meaning.

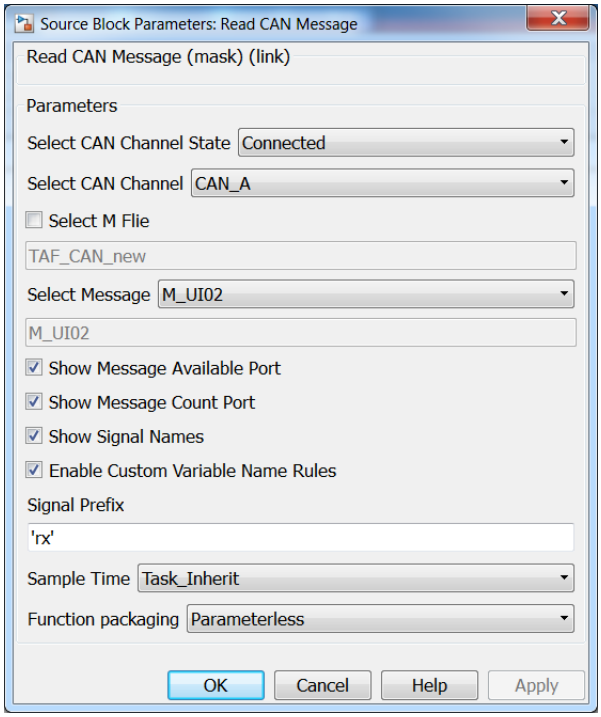
Note: The signal initial value 0 means hexadecimal value, not a physical value in real.

### 3.3.4 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



**Parameters:**

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.

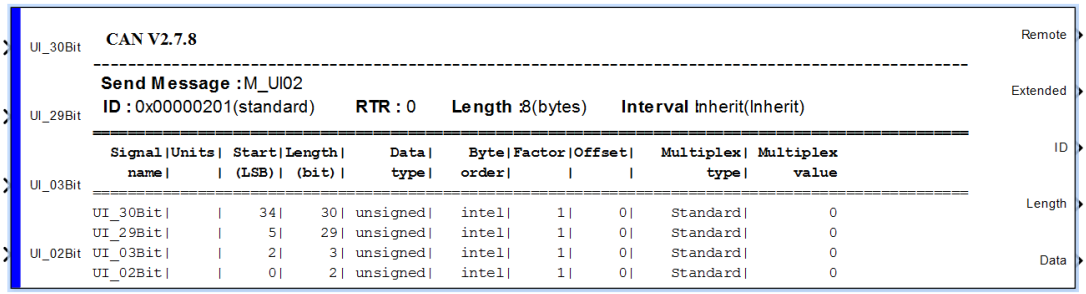
b. Parameterized: use C functions with parameters. This option offers more stable compilation.

**Output:**

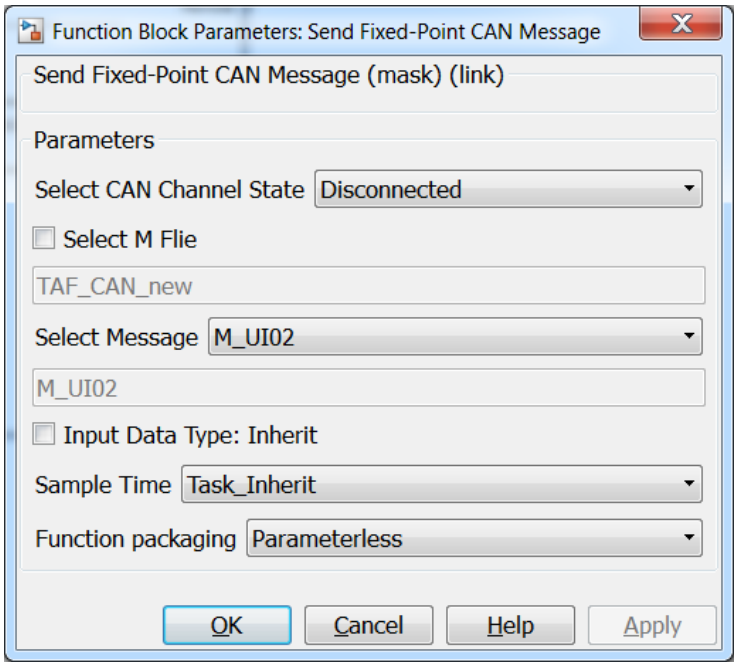
1) The signal needs to be transmitted. The unit should be the actual physical unit.

## 3.3.5 Send Fixed-Point CAN Message

This module is used to send CAN messages. The CAN signal is packaged and sent, in fixed-point type, requires the Fixed-point tool pre-installed. The module is not directly compatible with DBC, but can load the m file converted by DBC.




Send Fixed-Point CAN Message




**Parameters:**

1) Select CAN Channel State: Select the channel state, including Connected, Disconnected, and DisconnectedOnlyData options. When Connected is selected, there is no output port, message information is sent directly to the selected channel; when Disconnected is selected, the module will have an output port, and the message information will be output from these ports; when selecting DisconnectedOnlyData, it is only used for data conversion and CAN-independent.

2) Select CAN Channel: CAN channel selection.

3) Select M File: Select the m file converted by DBC conversion tool, need to add m file to the MATLAB path, after each selection, when clicking "OK" and "Apply", need to double-click the module again to select Message.

4) Select Message: Select a CAN message.

5) Input data type: When checked, the data type of the signal is inherited backwards; when not checked, the signal type is automatically defined as the fixed-point data type with calibration.

6) Sample time: Module simulation time selection.

7) Function packaging: Module C code function settings, divided into two types, Parameterless and Parameterized. Parameterless is a parameterless type function, code

execution efficiency is high, while Parameterized is a function with a parameter type, which supports the case that the C code corresponding to the module input and output signal is a local variable, preventing compilation errors.

**Input:**

1) For each signal to be packaged, the value of the signal is the physical value of the actual meaning.

**Output:**

When the channel status is Disconnected, the output port is as follows, there is no output port when Connected is selected.

1) Remote: Message frame type, 1 is remote frame, 0 is data frame.

2) Extended: Message frame type, 1 is an extended frame, 0 is a standard frame.

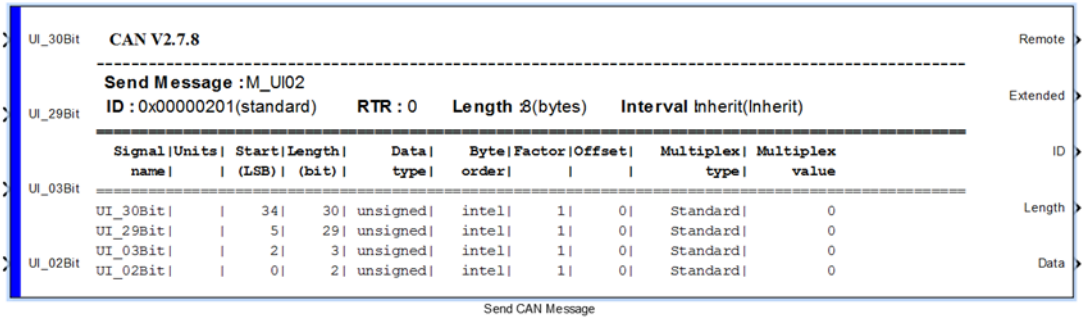
3) ID: Message ID.

4) Length: The length of the message data.

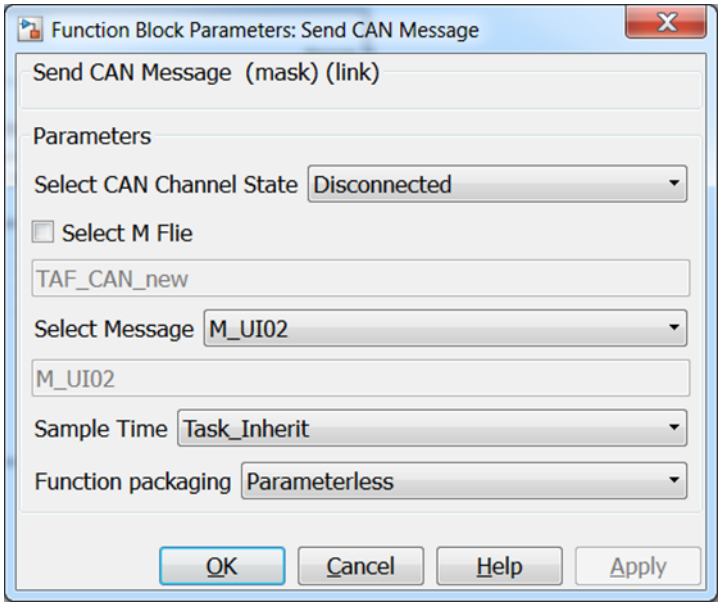
5) Data: Message data.

### 3.3.6 Send CAN Message

This module is used to send CAN messages. The CAN signal is packaged and sent, the signal type is inherited, the module is not directly compatible with DBC, but can load the m file converted by DBC.




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:**

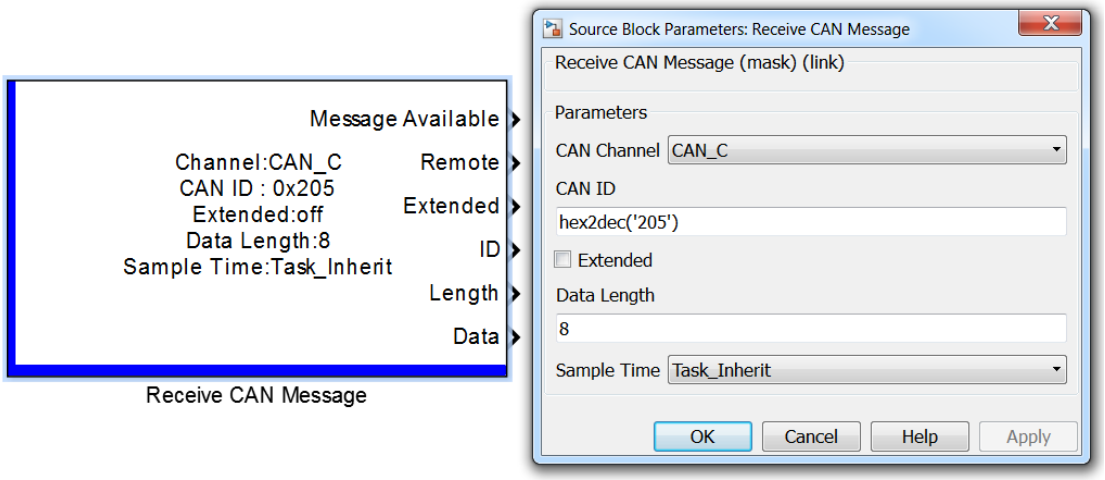
1) The signal needs to be transmitted. The unit should be the actual physical unit.

**Output:**

1) 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

## 3.3.7 Receive CAN Message

This module is used to receive CAN messages. Can be used with the Read CAN Message module, when the channel status is selected as Disconnected, it can re-translate the date with certain ID.




**Parameters:**

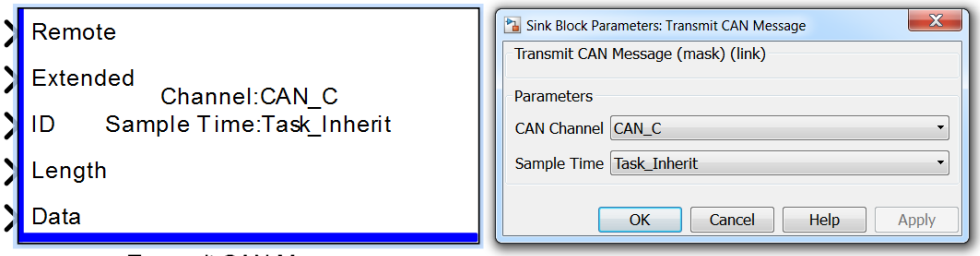
1) CAN channel: CAN channel selection.
2) CAN ID: The ID of the message to be received.
3) Extended: Message type to be received. If checked: extended frame. Otherwise, standard frame.
4) Data Length: The data length of the to-be-received message.
5) Sample Time: Define the task scheduling time of this block being triggered.

**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.

### 3.3.8 Transmit CAN Message

This module is used to send CAN messages. Can be used with the Send CAN Message module, and when the channel status is selected as Disconnected, certain signal values can be sent out.




Transmit CAN Message

**Parameters:**

1) CAN Channel: Channel selection
2) Sample Time: Define the task scheduling time of this block.

**Inputs:**

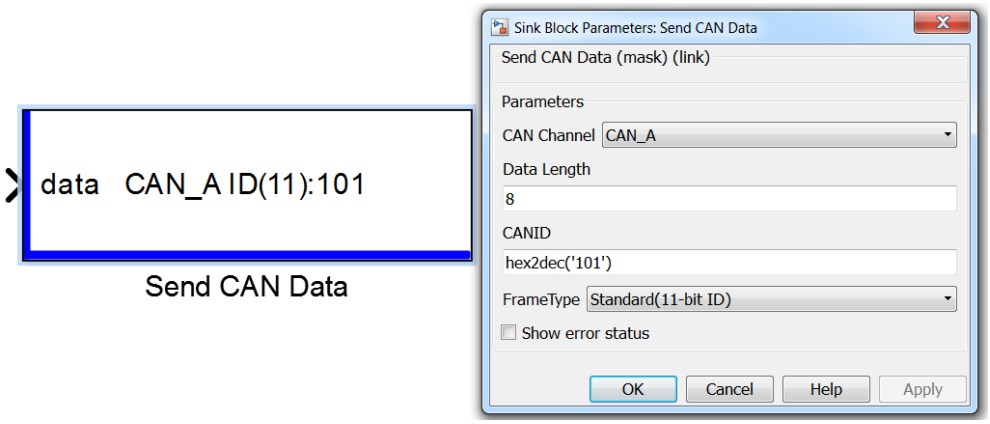
1) Remote: Flag for frame type, 1 stands for remote frame. 0 stands for data frame.
2) Extended: Flag for frame type, 1 stands for extended frame. 0 stands for standard frame.
3) ID: Message ID.

4) Length: Message data length.

5) Data: Message data.

### 3.3.9 Send CAN Data

This module is used to send CAN messages.




**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.

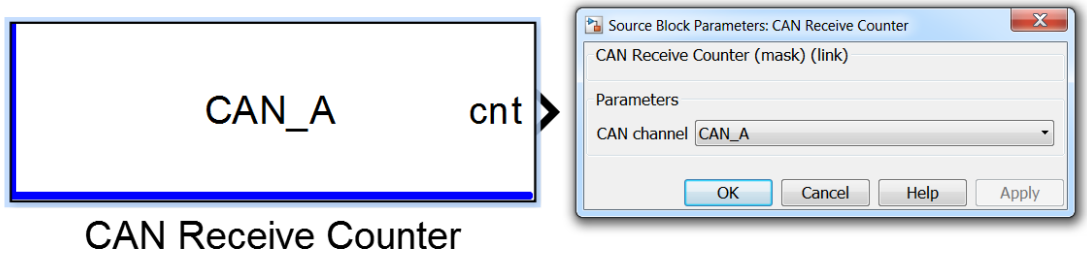
4) Frame Type: Drop-down list for frame type selection.

**Inputs:**

1) data: The message data to be sent.

### 3.3.10 CAN Receive Counter

The module is used to monitor whether the CAN BUS receives data. When receive each frame of message, the counter is incremented by 1.




**Parameters**

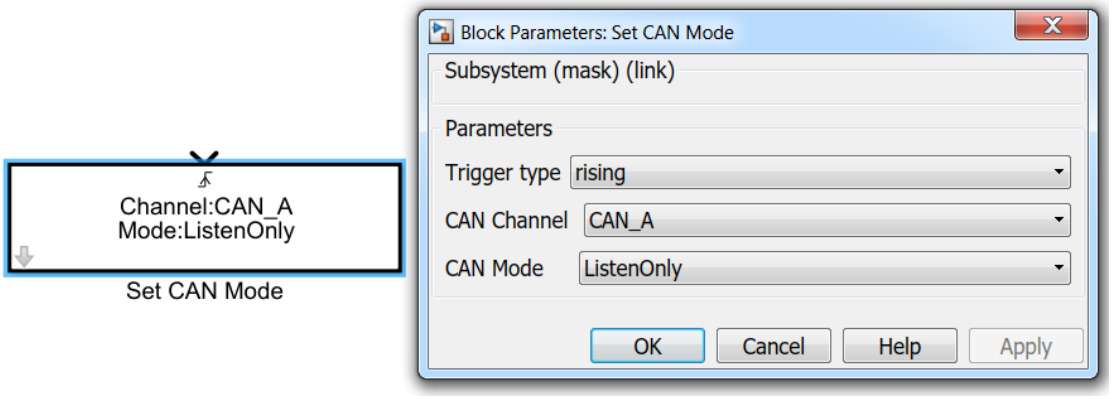
1) Select CAN Channel: CAN channel selection.

**Output:**

1) cnt: If the selected channel receives a data frame, increase by 1.

## 3.3.11 Set CAN Mode

This module is used to select the operating mode of the CAN channel.




**Parameter:**

1) Trigger type: Select the trigger type.

2) CAN Channel: CAN channel selection.

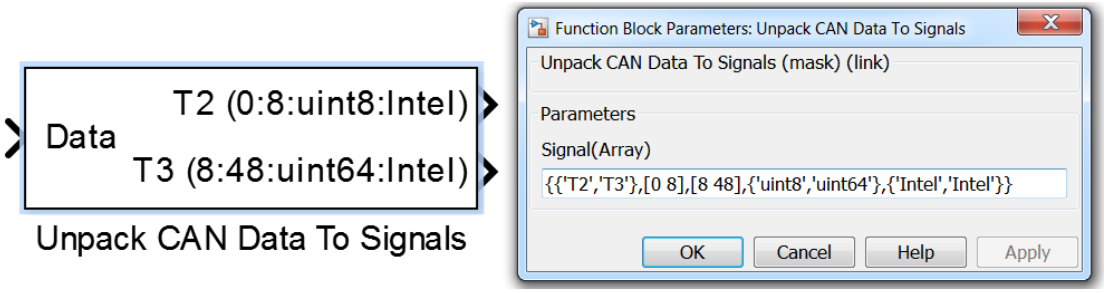
3) CAN Mode: Work mode selection, normal mode or listen mode.

**Input:**

1) Trigger signal: Trigger signal.

### 3.3.12 Unpack CAN Data to Signals

This module is used for signal unpack of CAN data




**Parameter:**

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

**Inputs:**

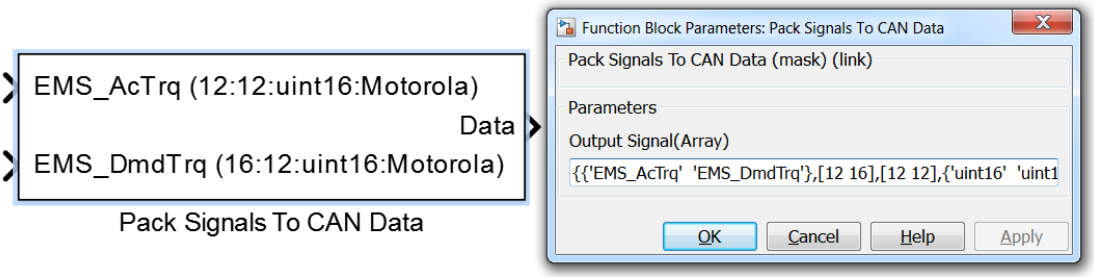
1) Data: the message data to be unpacked.

**Outputs:**

1) Unpacked signals from the CAN data, in Hex value

### 3.3.13 Pack Signals to CAN Data

This module is used to pack signals into CAN.




**Parameters:**

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

**Inputs:**

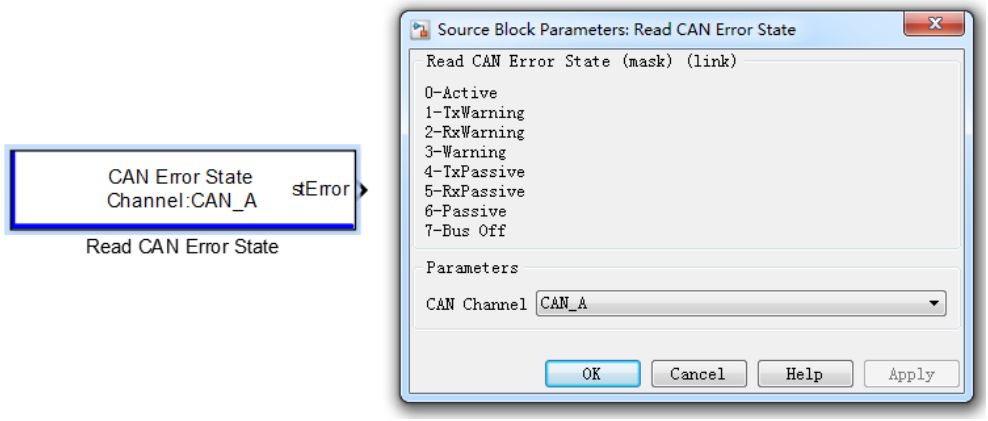
1) Signals to be packed, values are in HEX.

**Output:**

1) Data: the packed CAN message data.

## 3.3.14 Read CAN Error State

This module is used to diagnose if the CAN channel has error.

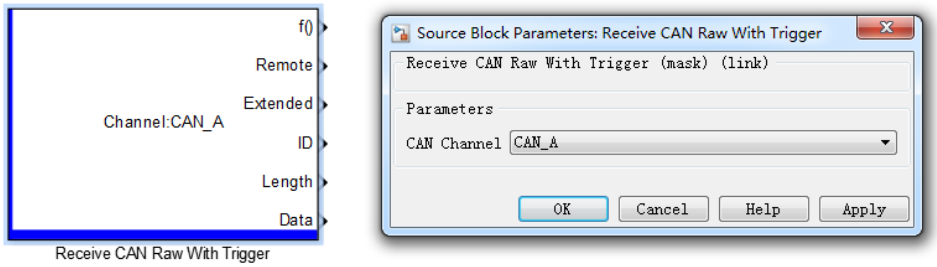



**Parameters:**

1) CAN Channel: Channel selection

## 3.3.15 Receive CAN Raw with Trigger

This module receives CAN messages using interrupts.




**Parameter:**

1) CAN Channel: channel selection

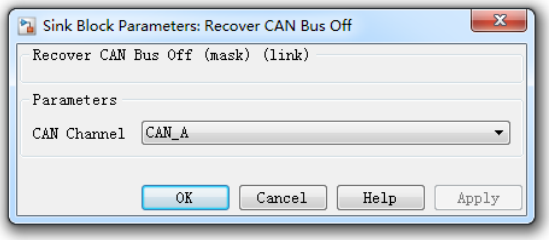
**Output:**

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

### 3.3.16 Recover CAN Bus Off

This module is used to recover from CAN Bus Off faults.


**Parameters:**

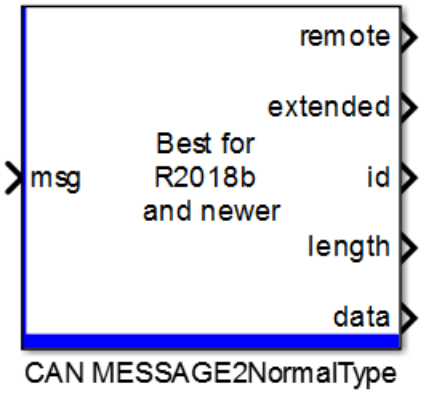
1) CAN Channel

**Input:**

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

### 3.3.17 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.




**Input：**

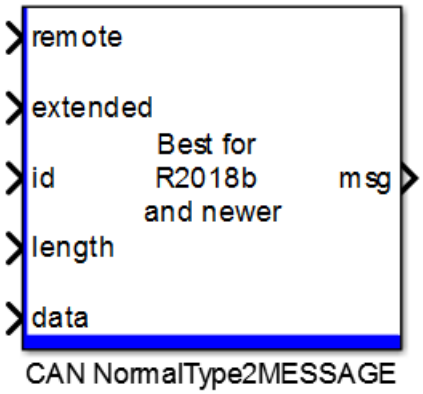
1) msg：Simulink default CAN_MESSAGE type.

**Output：**

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

### 3.3.18 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.




**Input：**

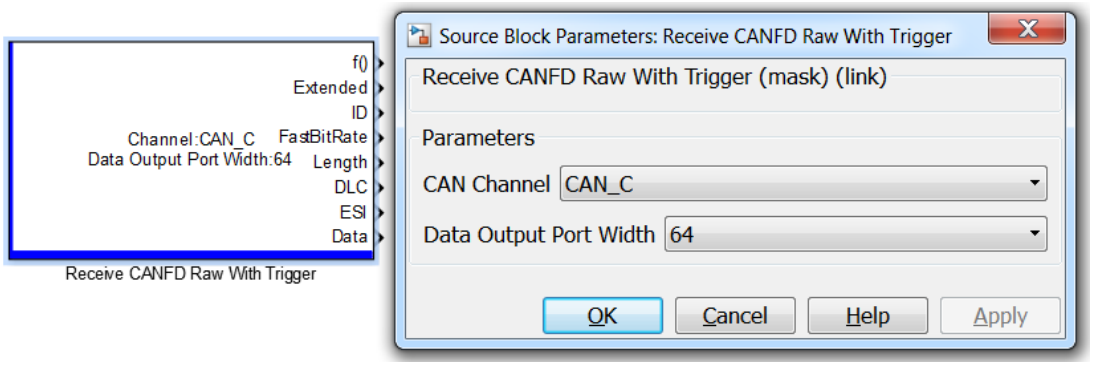
1) remote：Remote frame:1 Data frame: 0。
2) extended：Extended frame:1 Standard frame: 0
3) id：Message ID
4) length: Message length
5) data: Message data

**Output：**

1) msg：Simulink default CAN_MESSAGE type.

## 3.3.19 Receive CANFD Raw With Trigger

This block receives CANFD by interruption.




**Parameter：**

1) CAN Channel: CAN channel selection

2) Data Output Port Width: Data output port width

**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

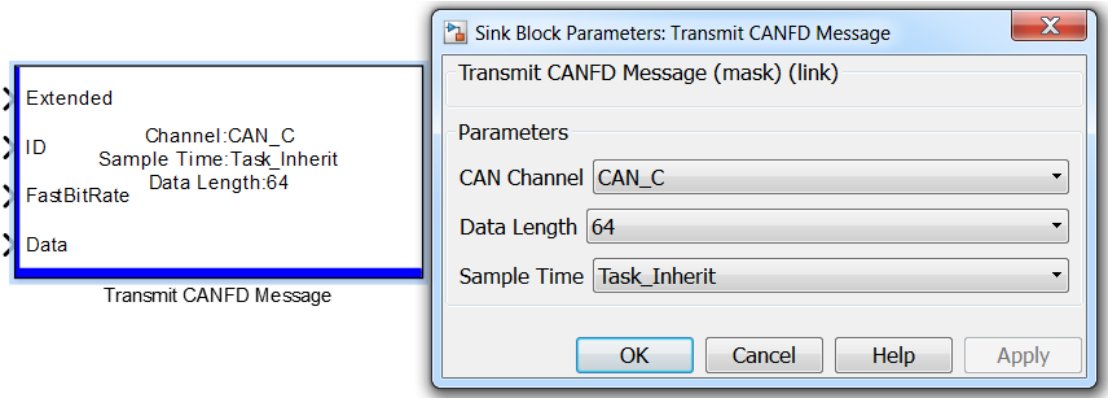
6) DLC： Data Length Code.

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.

### 3.3.20 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.




**Parameter:**

1) CAN Channel: CAN channel selection

2) Data Length: Data length

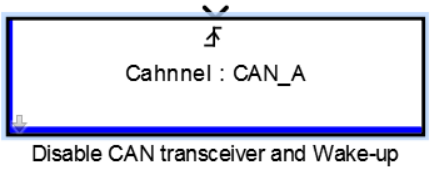
3) Sample Time: Task running time

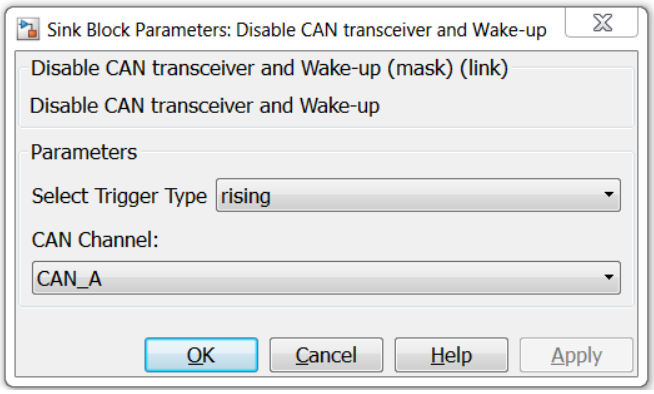
**Input:**

1) Extended： Extended frame:1 Standard frame: 0
2) ID: Message 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.

### 3.3.21 Disable CAN transceiver and Wake-up

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




**Parameter:**

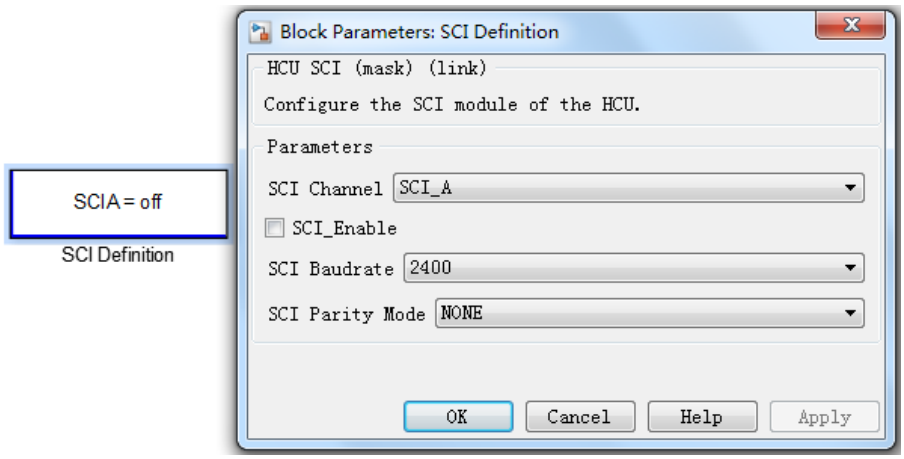
1) Select Trigger Type: Trigger type selection
2) CAN Channel: CAN channel selection

## 3.4 Serial Communication Interface (SCI) Block

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

### 3.4.1 SCI Definition

This module is used for SPI configuration.

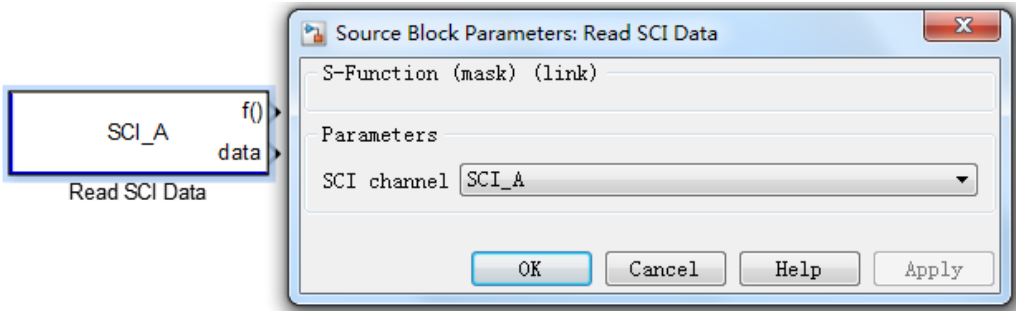



**Parameters:**

1. SCI Channel: SPI channel selection.
2. SCI_Enable: Enable selected channel.
3. SCI Baud rate: Channel baud rate setup.
4. SCI Parity Mode: Parity check mode setup.

### 3.4.2 Read SCI Data

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




**Parameter:**

1) SCI_Channel: SCI communication channel selection.

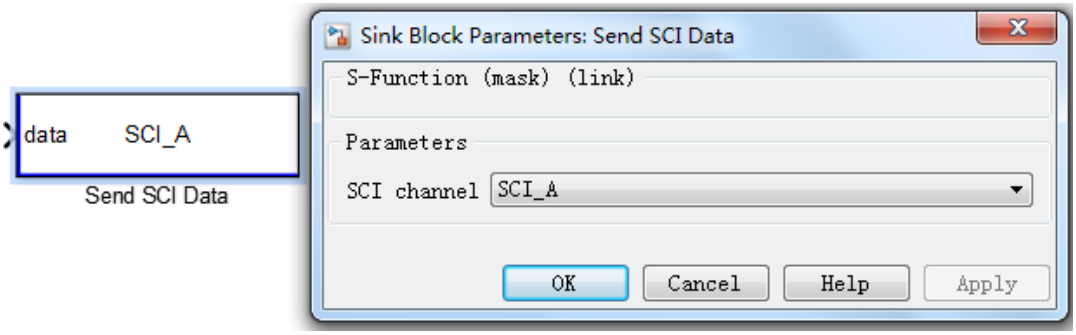
**Outputs:**

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.

### 3.4.3 Send SCI Data

This block will send SCI data to assigned channel.




**Parameter:**

1) SCI_Channel: SCI channel selection.

**Input:**

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

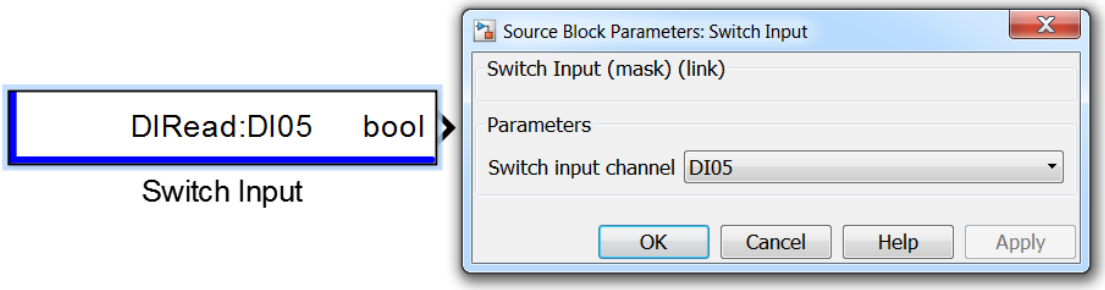
## 3.5 Digital I/O

The module is input measurement and output control of digital IO, of which the switching type includes Switch Input, Switch Output, etc.

PWM type includes IPM Read and PWM Output.

### 3.5.1 Switch Input

The module is used to read the digital input channel voltage level.




**Block Parameters:**

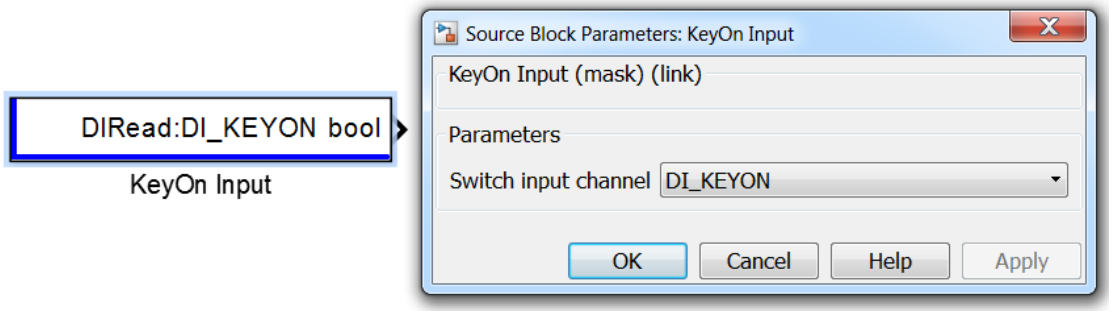
1) Switch input channel: Digital channel selection

**Output:**

1) bool: 1 means high, 0 means low

### 3.5.2 KeyOn Input

This module is used to collect Keyon signals.




**Block Parameters:**

1) Switch input channel: Channel selection, KEYON is designed as the only channel for this module.

Note: Consult the VCU technical manual to determine whether the key switch is a digital or analog signal. If it is an analog signal, the following options need to be configured; if it is a digital signal, the parameter settings can be ignored.

2) Key AD2Volt Factor: The ratio factor to multiply the AD sample value into the input

voltage value is determined by the parameters in Table 2.2.1 of the controller technical manual, and the calculation formula is as follows:

Key AD2Volt Factor = (Divided Voltage Resistor + Serial Resistor) / (819 * Divided Voltage Resistor)

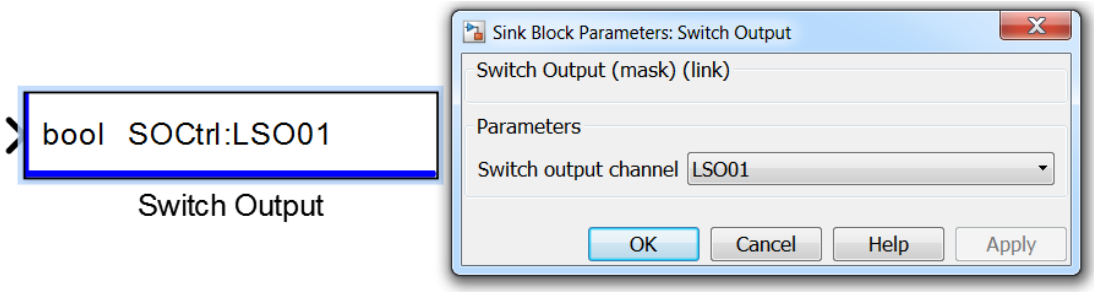
3) Key Off Threshold Volt: If the voltage is lower than this value, the output is 0 (e.g. less than 7 is 0).
4) Key On Hysteresis Volt: If the voltage is greater than the Key Off Threshold Volt plus this value, the output is 1 (e.g. greater than 7+2 is 1).

**Output:**

1) bool: 1 means high, 0 means low

### 3.5.3 Switch Output

The module is used to configure the outputs of high-side, low-side, and H-bridge channels in a switching output.




**Block Parameters**:

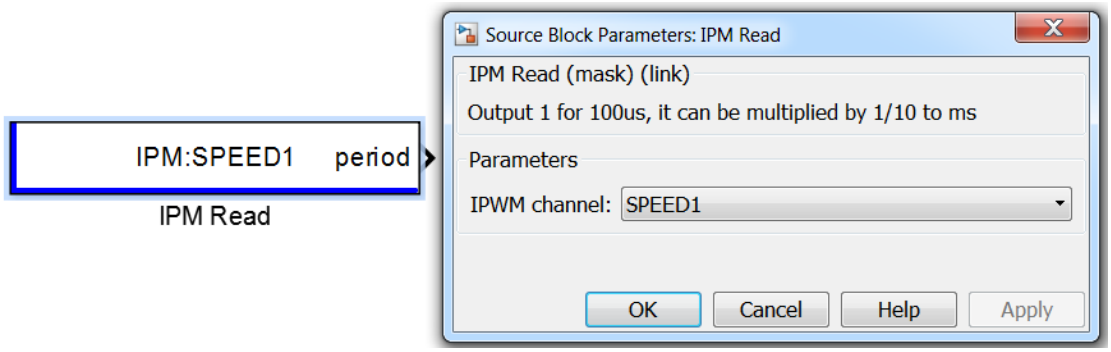
1) Switch output channel: Digital channel selection.

**Output:**

1) Switch output channel: Digital channel selection.

### 3.5.4 IPM Read

The module is used to acquire the frequency of a specified channel, and the output value is a period, which can be converted to a frequency by calculation.




**Parameter:**

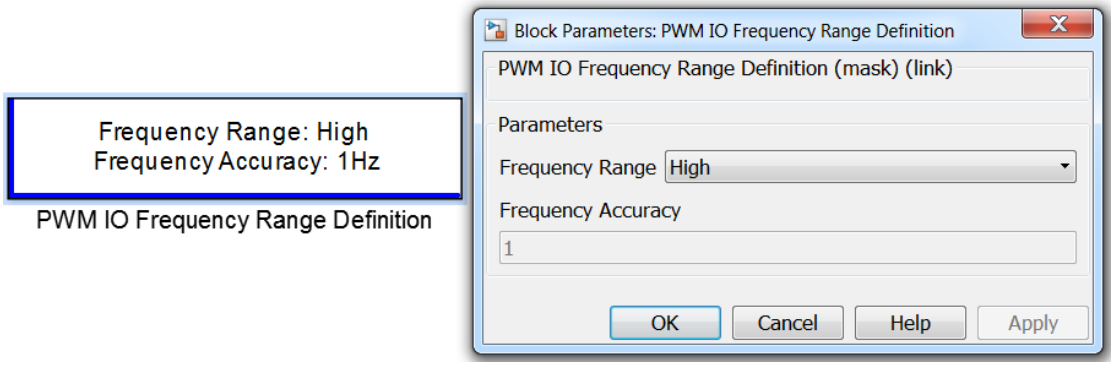
1) IPM Channel: digital channel selection

**Output:**

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

### 3.5.5 PWM IO Frequency Range Definition

This module is used to define the frequency accuracy, which is 1Hz by default.

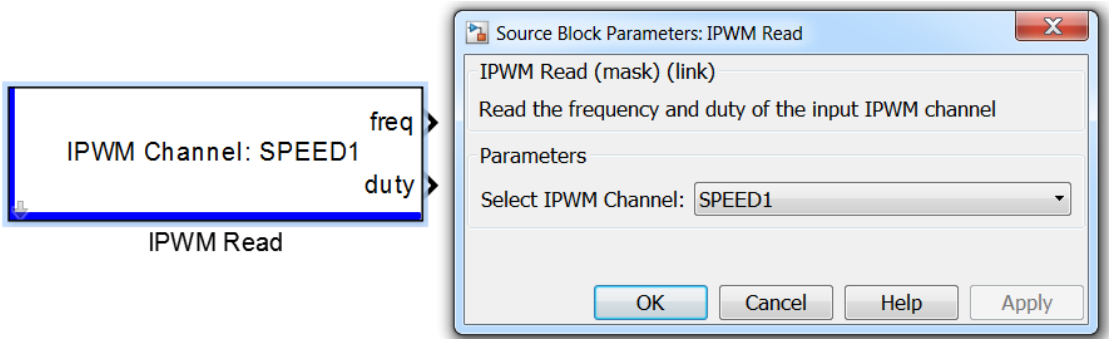



**Block Parameters:**

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: Frequency accuracy. For the range of accuracy, please refer

to the hardware technical manual.

### 3.5.6 IPWM Read




**Block Parameter:**

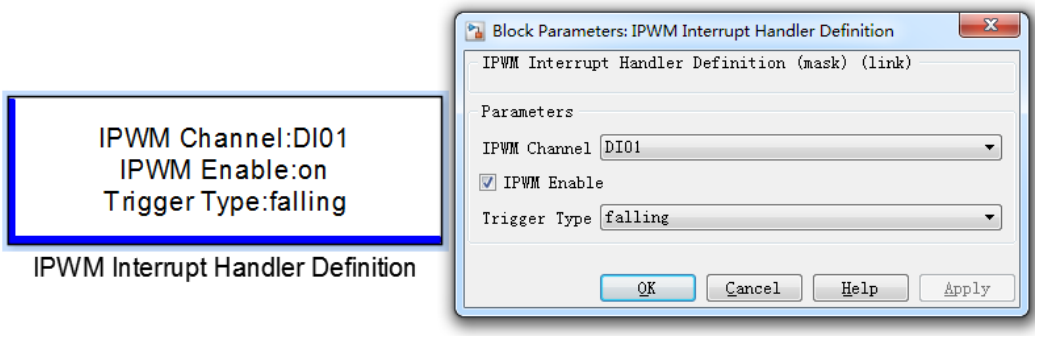
1) Select IPWM Channel: PWM inputs channel selection

**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%

### 3.5.7 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:**

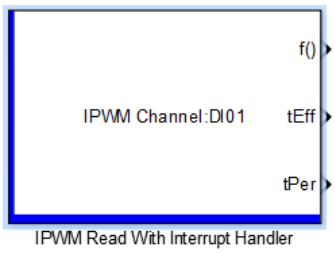
1) IPWM Channel: Channel selection.

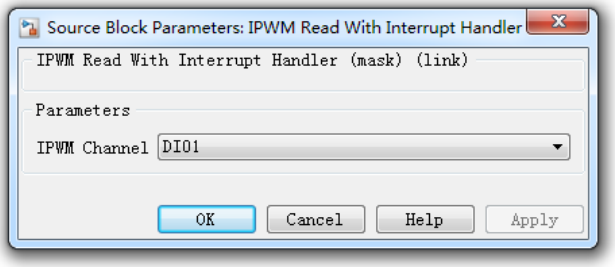
2) IPWM Enable: Channel enabled.

3) Trigger Type: Select the trigger mode. There are two ways: rising edge and falling edge.

### 3.5.8 IPWM Read with Interrupt Handler

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



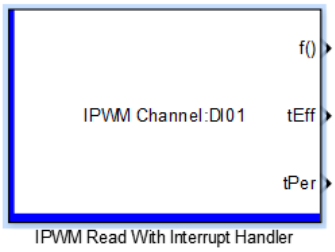

Block Parameter:

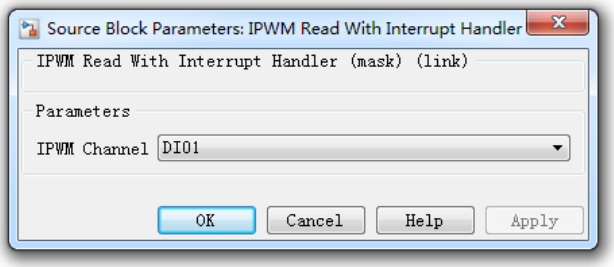
1) IPWM Channel: Channel selection

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).

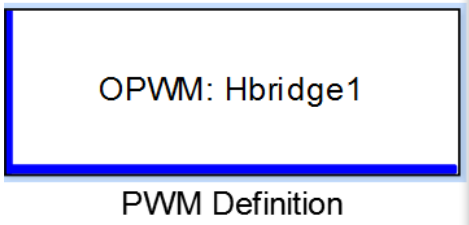
Note: When the trigger mode is rising edge trigger, the effective level time refers to high level time; when the trigger mode is falling edge trigger, the effective level time refers to low level time.



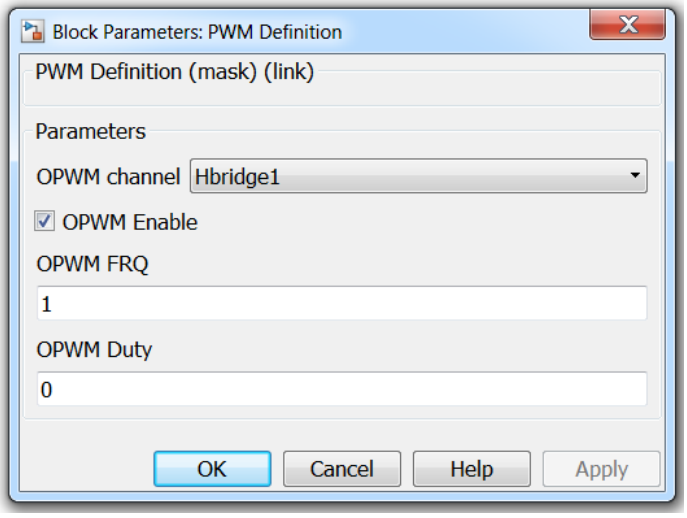



### 3.5.9 PWM Definition

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.




PWM Definition




Block Parameter:

1) OPWM channel: Channel selection.

2) OPWM Enable: Channel enable.

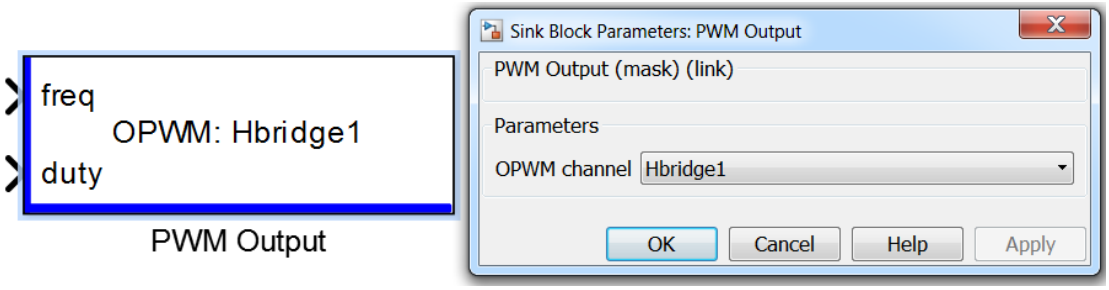
3) OPWM FRQ: Channel initialization frequency, unit is related to PWM IO Frequency Range Definition settings.

4) OPWM Duty: 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.

### 3.5.10 PWM Output

The module is used for PWM output control of high-side, low-side, and H-bridge channels.




**Block Parameters:**

1) OPWM channel: Digital channel selection.

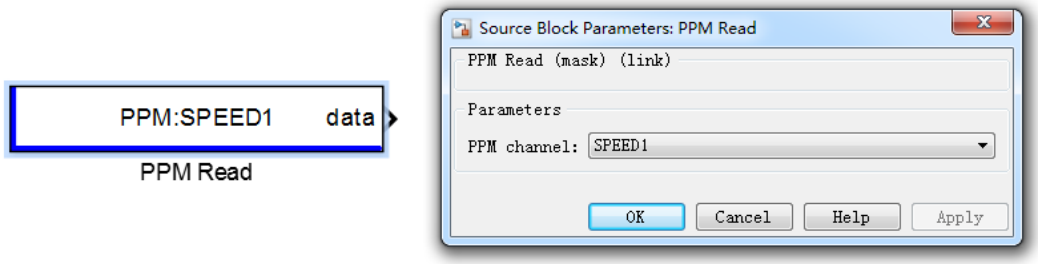
**Inputs:**

1) freq: Controls the frequency of the specified channel signal, unit is related to the PWM IO Frequency Range Definition setting.

2) duty: Controls the duty cycle of the specified channel signal, where 0-10000 corresponds to 0-100%.

### 3.5.11 PPM Read

This module is used to read PPM signal.




Block Parameter:

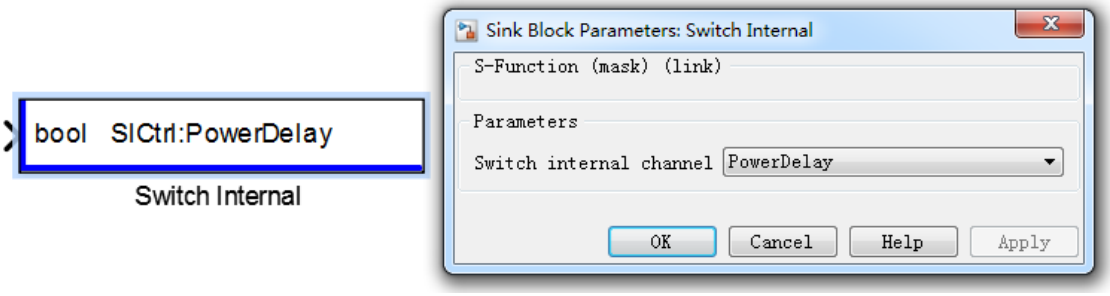
1) PPM channel: Channel selection.

Block Output:

1) data: Specifies the data for the channel.

### 3.5.12 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.



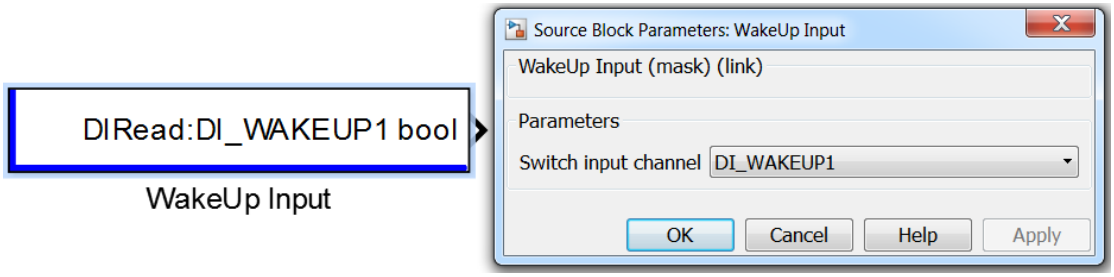

Block Parameter:

1) Switch internal channel

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

### 3.5.13 WakeUp Input

This module is used to detect if there is a wake-up signal.




**Block Parameters:**

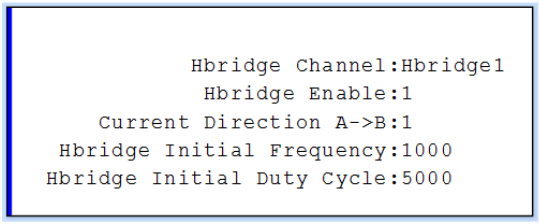
1) Switch input channel: Digital channel selection.

**Output:**

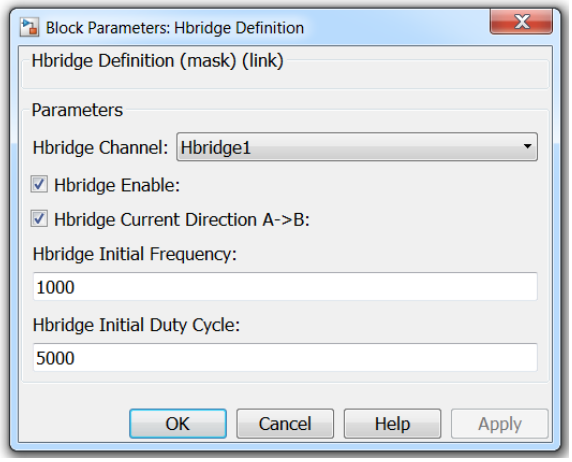
1) bool: 1 for high and 0 for low.

## 3.5.14 H-bridge Definition

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




Hbridge Definition




**Block Parameters**:

1) Hbridge Channel: Channel selection.

2) Hbridge Enable: Channel enabled. Checking indicates enable; unchecking indicates not enabling.

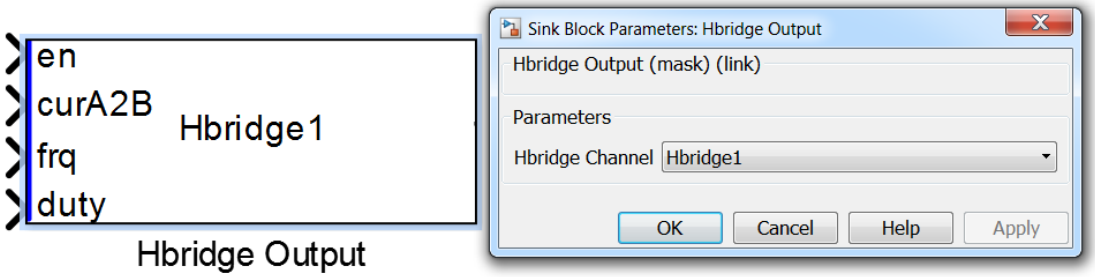
3) Hbridge Current Direction A->B: Current direction setting. Checking indicates flow from A to B, and checking uncheck indicates flowing from B to A. A and B are the two ports of the H bridge.

4) Hbridge Initial Frequency: Initialization frequency, unit is related to PWM IO Frequency Range Definition settings.

5) Hbridge Initial Duty Cycle: Initializes the duty cycle, where 0-10000 corresponds to 0-100%.

## 3.5.15 H-bridge Output

This block controls H-bridge output.




**Block Parameters**

1) Hbridge Channel: Channel selection.

**Input:**

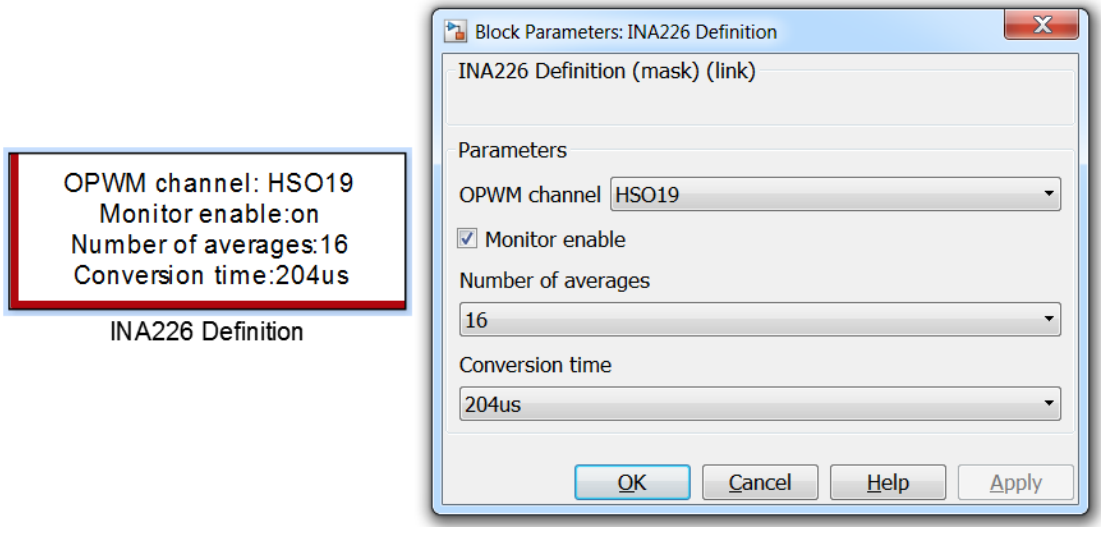
1) en: enable control, 1 is enabled, 0 is not enabled.

2) curA2B: Current direction, 1 is A flow to B, 0 is B to A.

3) frq: Frequency control, unit is related to PWM IO Frequency Range Definition settings.

4) duty: duty ratio control, where 0-10000 corresponds to 0-100%.

## 3.5.16 INA226 Definition



**Block Parameter:**

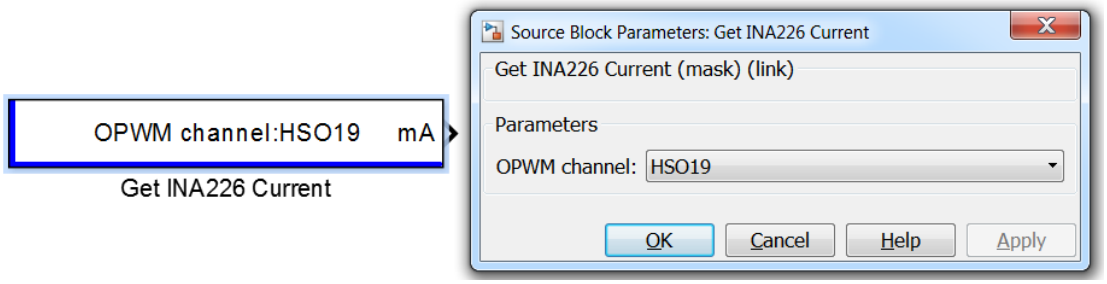
1) OPWM Channel: Channel selection.

2) Monitor enable: Enables the IN226 monitoring function.

3) Number of averages: Average the number of samples.

4) Conversion time: Bus and shunt voltage conversion time.

## 3.5.17 Get INA226 Current



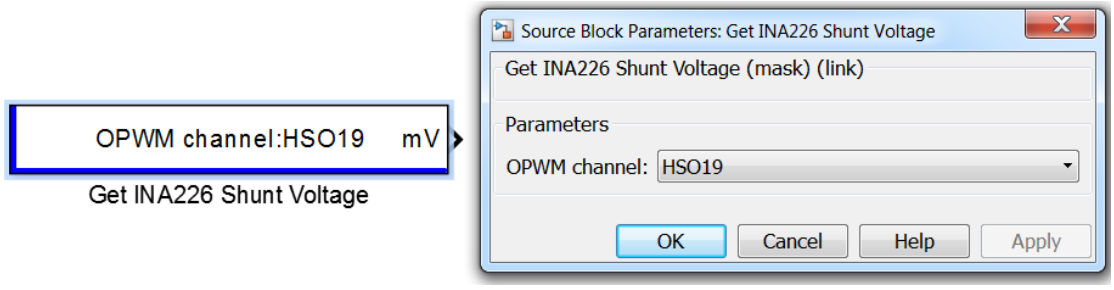

Block Parameter:

1) OPWM Channel: Channel selection

Block Output:

1) mA：Read the INA226 current, in mA.

## 3.5.18 Get INA226 Shunt Voltage



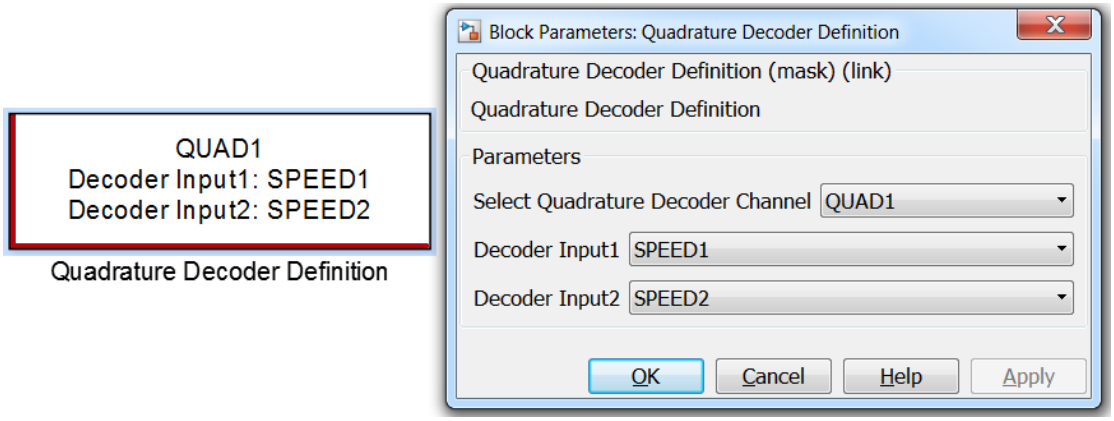

Block Parameter:

1) OPWM Channel: Channel selection

Block Output:

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

## 3.5.19 Quadrature Decoder Definition



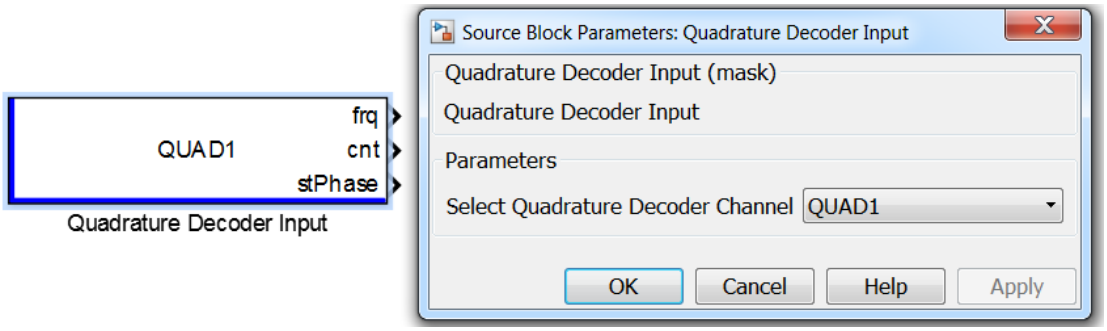

Block Parameter

1) Select Quadrature Decoder Channel: Channel selection.

2) Decoder Input1: Decode input 1

3) Decoder Input2: Decode input 2

## 3.5.20 Quadrature Decoder Input




Block Parameter

1) Select Quadrature Decoder Channel: Channel selection

Block Output:

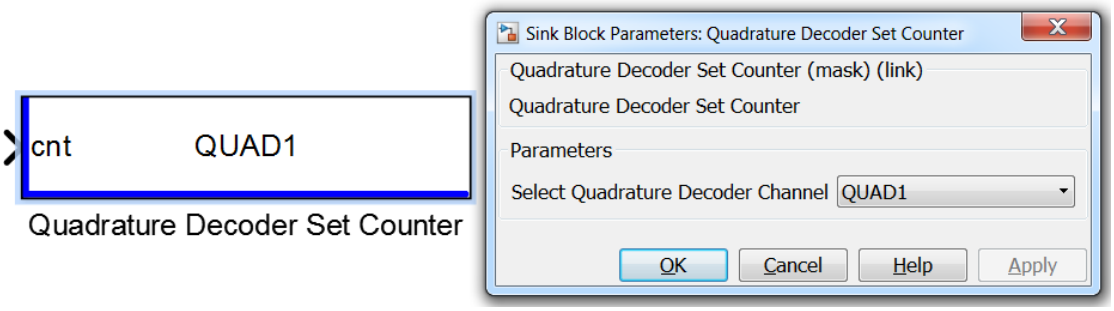
1) frq: Frequency, in Hz.

2) cnt: square wave counter, add 1 if phase state stPhase is 1, subtracted by 1 if -1.

3) stPhase: Phase state, 1 represents Channel 1 leading Channel 2 phase 90°, -1 represents Channel 1 lag Channel 2 phase 90°.

### 3.5.21 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.



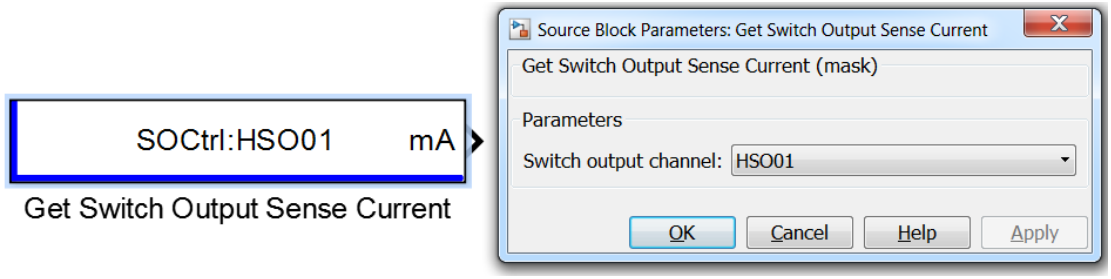

Block Parameter:

1) Select Quadrature Decoder Channel: Channel selection

Block Input:

1) cnt：Initial square wave count.

### 3.5.22 Get Switch Output Sense Current



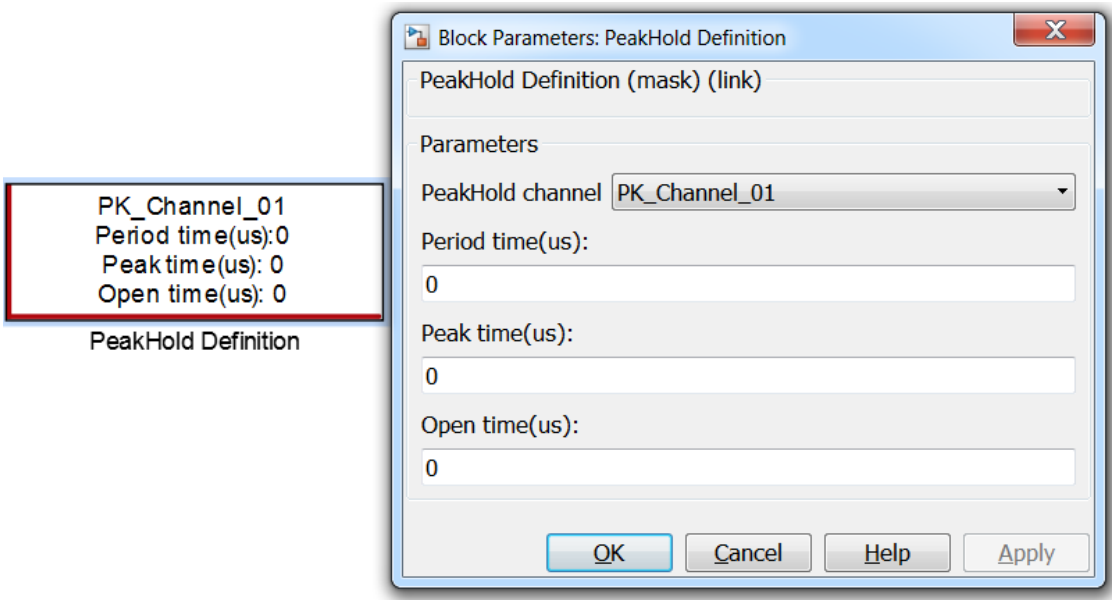

Block Parameter

1) Switch output channel: Channel slection

Block Output:

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

### 3.5.23 PeakHold Definition




Parameter:

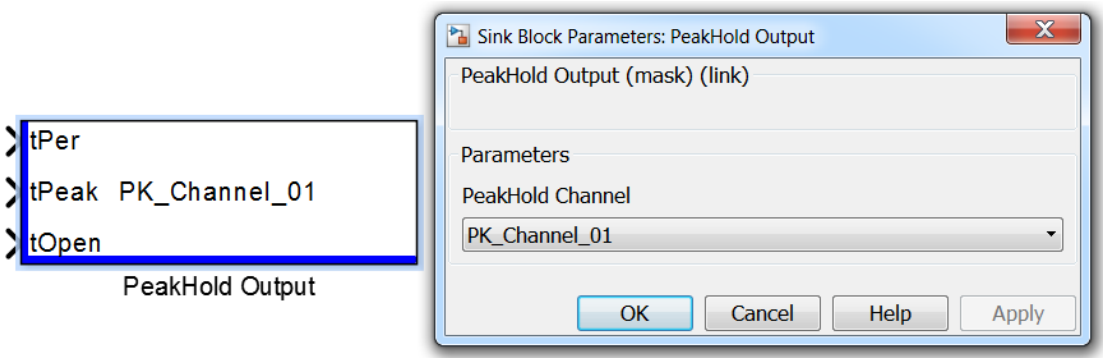
1) PeakHold channel: Channel selection.

2) Period time(us): The initial value of the period time.

3) Peak time(us): The initial value of peak time.

4) Open time(us): The open time is the sum of the time of Peak plus Hold.

### 3.5.24 PeakHold Output




Parameter:

1) PeakHold channel: Channel selection.

Input:

1) tPer: Cycle time, in us.

2) tPeak: peak time, in us.

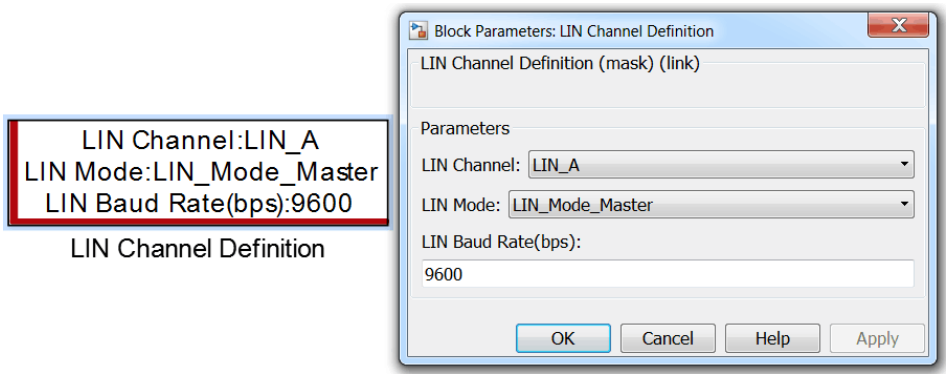
3) tOpen: The open time is the sum of the time of Peak plus Hold, in us.

## 3.6 LIN Communication

For LIN communication module, LIN Channel Definition is supported for current controllers, while other modules are suitable for different microcontroller types. The controller with the main chip of Infineon currently only supports master mode, and the supported modules are LIN Master Transmit ID And Data, LIN Master Transmit ID and LIN Master Receive Data. The main chip of NXP supports master mode and slave mode, the supported modules are LIN Transmit Data, LIN Receive Data and LIN Get Status. In addition, it is recommended that if customer is not familiar with LIN module development, it's better to firstly try the Demo model in the installation package, before actually developing the LIN bus by their own.

### 3.6.1 LIN Channel Definition

This module is used for LIN bus initialization parameter settings. It mainly includes LIN channel enable, LIN master-slave mode setting, baud rate setting, etc.




**Block Parameters:**

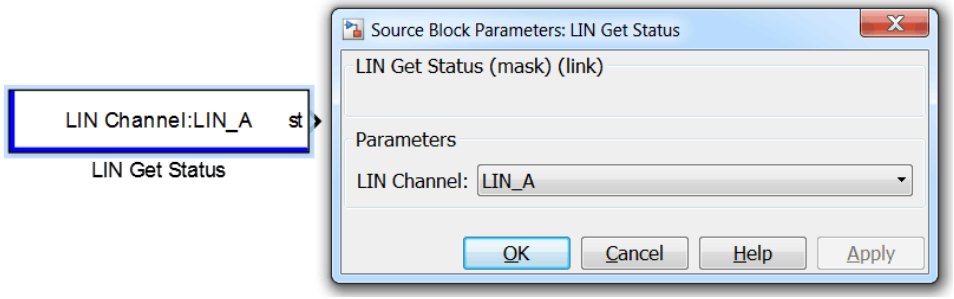
1) LIN_Channel: LIN channel selection.

2) LIN Mode: LIN mode selection, including master mode and slave mode.

3) LIN Baud Rate(bps): LIN baud rate setting, in bps.

### 3.6.2 LIN Get Status

This module is used to obtain the status of a LIN channel.




**Block Parameters:**

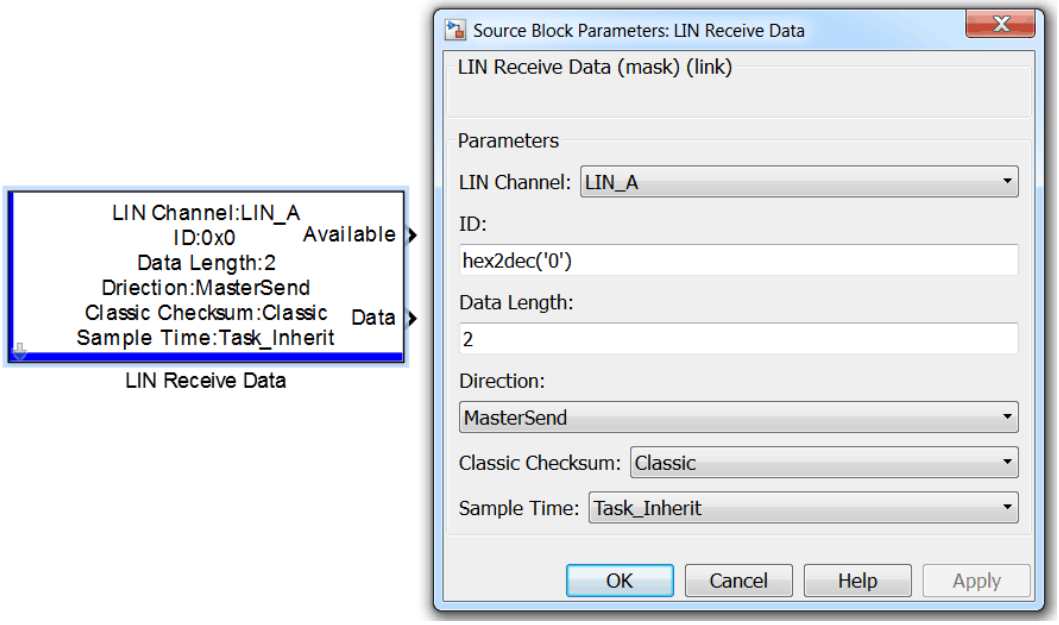
1) LIN_Channel: LIN channel selection.

**Block Output:**

1) st: Output status, see Appendix 2.

### 3.6.3 LIN Receive Data

The module is used to receive data from the LIN bus, when the LIN Channel Definition is configured as the master mode, this module needs to set the parameter Direction to MasterReceive to receive data from the slave, and at the same time, it needs to use the LIN Transmit Data module that sets the Direction to MasterReceive, to firstly send the id of the data to be received. When LIN Channel Definition is configured for slave mode, the module needs to set the parameter Direction to MasterSend to receive the ID and data sent by the master.




**Block Parameters**

1) LIN_Channel: LIN channel selection.

2) ID: The address of the data to be received.

3) Data Length: The length of the data to be received.

4) Direction: Data transmission direction, including master sending and slave sending.

5) Classic Checksum: Checksum options.

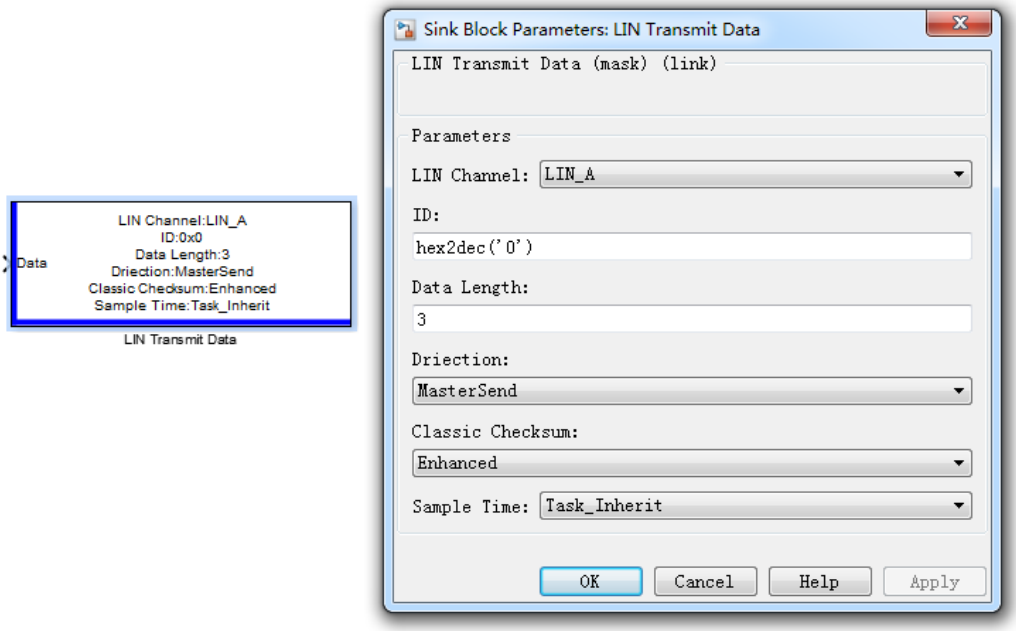
6) Sample Time: Module funning time.

**Block Outputs:**

1) Available: output of 1 means data is valid, otherwise invalid.
2) Data: data to be received

## 3.6.4 LIN Transmit Data

The module is used to send data from the LIN bus, note that when the LIN Channel Definition is configured as the master mode, when set Direction to MasterReceive, only send ID and the data input can be ignored. If set Direction to MasterSend, will send both data and ID at the same time. When the LIN Channel Definition is configured as slave mode, it must set the parameter Direction to MasterReceive and send the data when specific ID is received.




**Block Parameters:**

1) LIN_Channel: LIN channel selection.
2) ID: The address of the data to be sent.

3) Data Length: The length of the data to send.

4) Direction: Data transmission direction, including master sending and slave sending.

5) Classic Checksum: Checksum options.

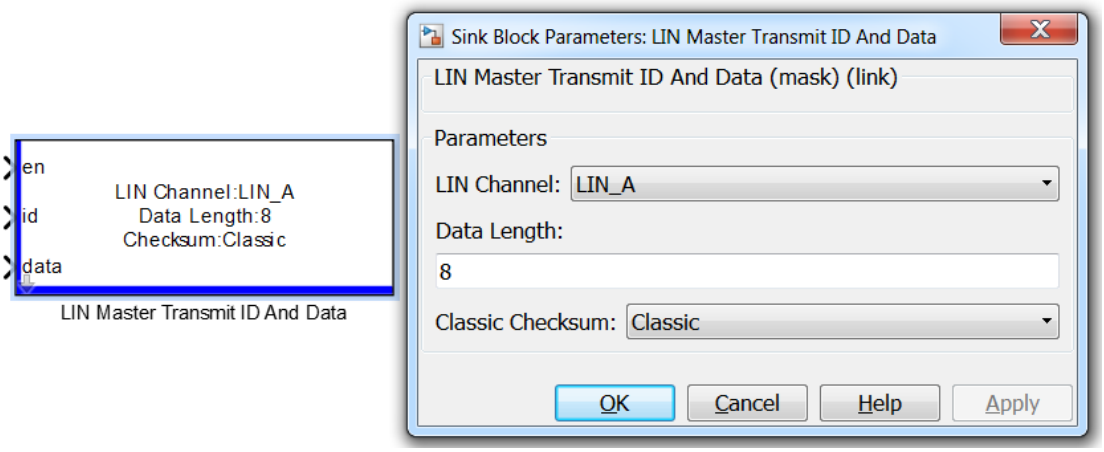
6) Sample Time: Module running time.

**Block Input:**

1) Data: data to be sent

### 3.6.5 LIN Master Transmit ID And Data

This module can only be used when LIN Channel Definition is configured as master mode and is used to send IDs and data.




**Parameter:**

1) LIN_Channel: LIN channel selection.

2) Data Length: The length of the data to send.

3) Classic Checksum: Checksum options.

**Input:**

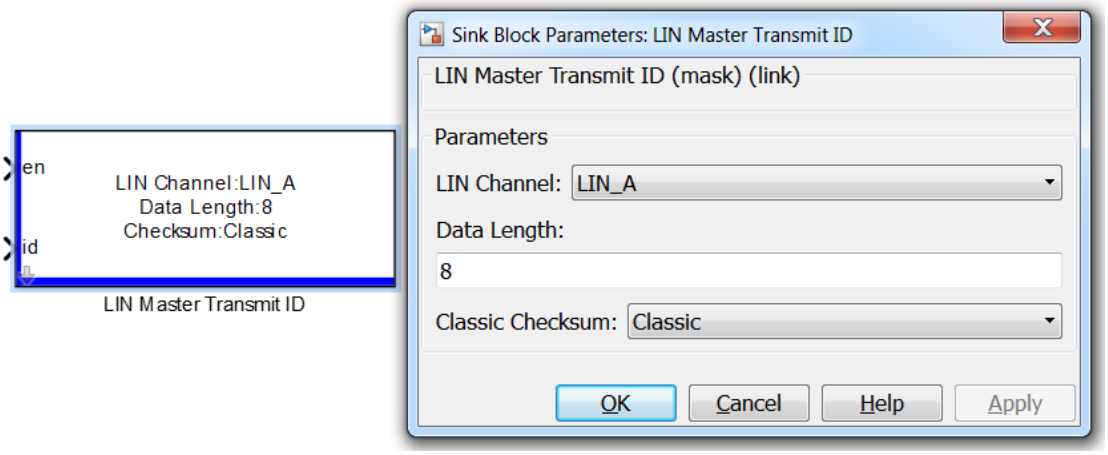
1) en: enable to send. 1 is enabled, 0 is not enabled.

2) id: ID to be sent.

3) Data: data to be sent.

### 3.6.6 LIN Master Transmit ID

This module can only be used when LIN Channel Definition is configured as master mode and is used to send IDs.




**Parameter:**

1) LIN_Channel: LIN channel selection.

2) Data Length: The length of the data to send.

3) Classic Checksum: Checksum options.

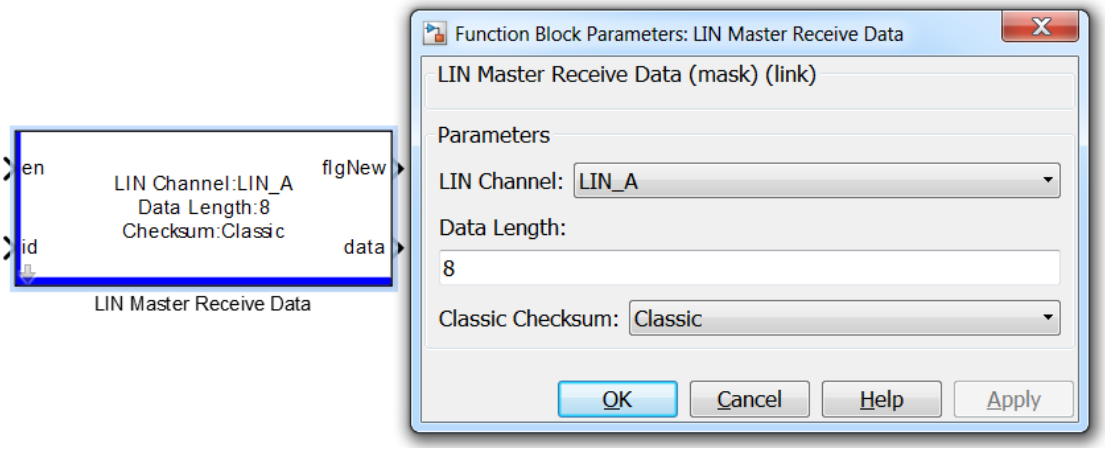
**Input:**

1) en: enables to send, 1 is enabled, 0 is not enabled.

2) id: The ID to be sent.

### 3.6.7 LIN Master Receive Data

The module can only be used when LIN Channel Definition is configured as master mode to receive slave data.




**Parameter:**

1) LIN_Channel: LIN channel selection.

2) Data Length: The length of the data to send.

3) Classic Checksum: Checksum options.

**Input:**

1) en: enables reception, 1 is enabled, 0 is not enabled.

2) id: The ID of the data to be received.

**Output:**

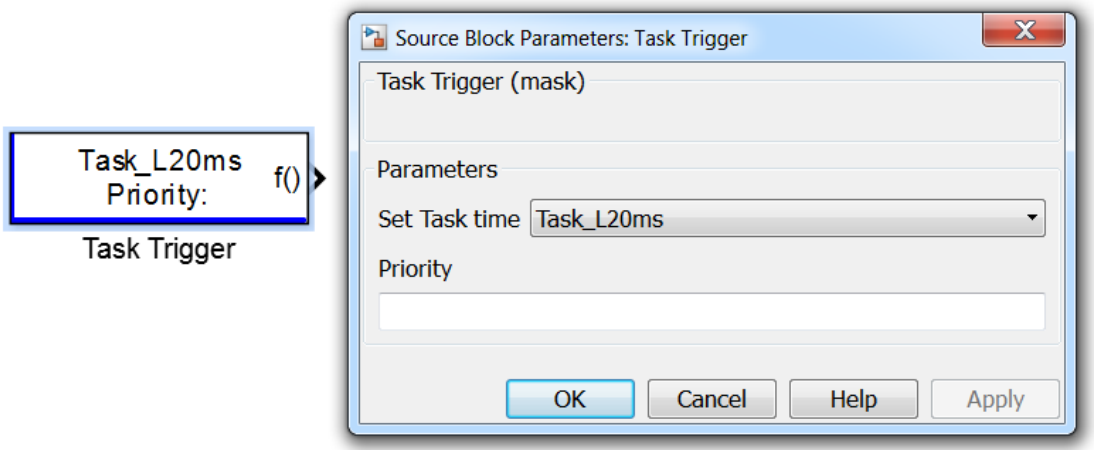
1) flgNew: whether the data is received, 1 is the data received, 0 is not received.

2) Data: Received data.

## 3.7 Task Scheduler

### 3.7.1 Task Trigger

This module is used to trigger tasks periodically.




**Parameter:**

1) Set Task Time: Select the task type.
2) Priority: Priority setting, the same as Priority in Simulink's Block Properties. It can control the execution order of the same type of tasks. The smaller priority value of the task, the earlier to be executed. If it is empty, Simulink will control the execution order by default.

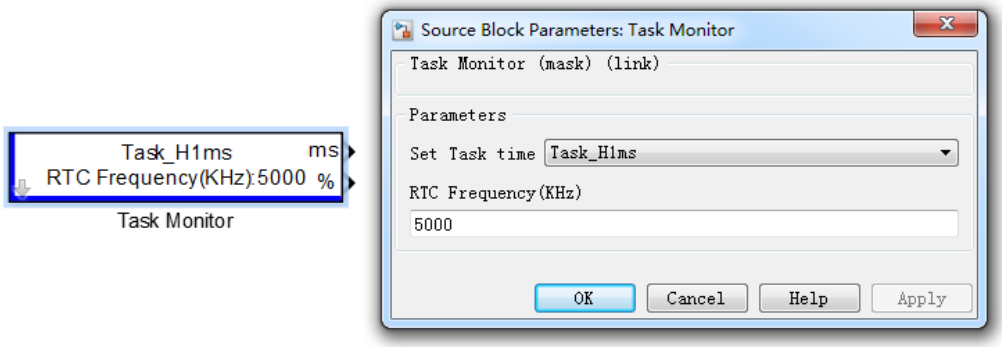
**Output:**

1) f(): Task with fixed-period

Note: H tasks have higher priority than L tasks. For all H tasks, the shorter the period, the higher the priority. For all L tasks, they have the same priority. If two tasks are assigned as the same task type, it needs to specify the priority of these two tasks to determine the order of execution.

### 3.7.2 Task Monitor

This module is used to monitor task scheduling time.




**Block Parameters:**

1) Set Task Time: Select the task type.

2) RTC Frequency (KHz): The task monitor clock frequency, in KHz. Different main chip types of controllers have different coefficients, as shown in the following table:

| Main chip | RTC Frequency(KHz) |
|---|---|
| Infineon TC27x/TC29xx | 100000 |
| NXP SPC57xx | 5000 |
| NXP SPC56xx | 150000 |

**Block Output:**

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

2. %: Load rate of chosen task type.

## 3.8 Non-Volatile Memory Blocks

NVM is power-down non-volatile data. The default data management process is to load data from ROM to RAM when power on, and save data from RAM to ROM when power off. After data is loaded into RAM after power on, each variable will have a mapping in RAM. The read and write of variables is happening in RAM, and it can read and written as many times as possible. When abnormal power down happened, it will use the data from

the last normally power down. If abnormal power failure during power-down when saving data, the data will have error and got lost when next power on.

NVM variables are divided into fixed NVM variables and non-fixed NVM variables. The difference between them is whether the address is fixed, that is, when model changes, the fixed NVM can fix the physical address of the variable. When model changes, the variable physical address of the non-fixed NVM is not fixed, and support to use .m file.

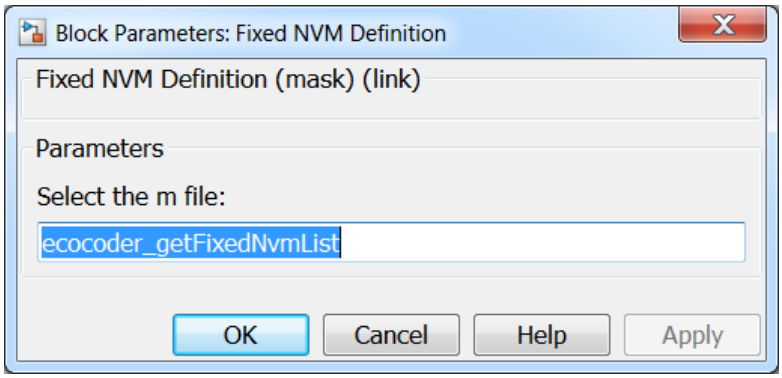
### 3.8.1 Fixed NVM Definition

This module is used to define fixed NVM data.

**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




**Block Parameters:**

1) Select the m file: Select the m file. The file contains configuration information for all variables, such as data types and initialization rules.

**Meaning of the parameters in the m file:**

1) Order: The order in which NVM variables are stored.

2) Name: Variable name.

3) Type: The data type of the variable.

4) Size: The data length of the variable.

5) Init: Initialized settings. 1 indicates that the initial value is equal to Value after code is updated, and 0 indicates that the initial value is qual to Flash after code is updated.

6) Value: The initial value of variable.

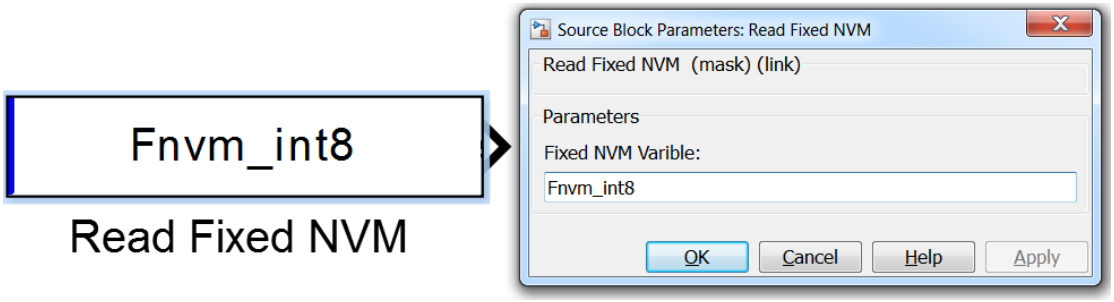
Note: You need to add an m file to the MATLAB path to use it. The definition format of the variables in the m file is as follows:

```
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
```

After updating the firmware program, the first power-on will determine whether a variable will use the default initial value or the original value from physical address, according to the definition in the Fixed NVM Definition module. In Fixed NVM Definition, as long as variables are not increased or decreased, and the type of variables does not change, the actual physical address of each variable remains unchanged.

### 3.8.2 Read Fixed NVM

This module is used for reading fixed NVM variables.




**Block Parameters**

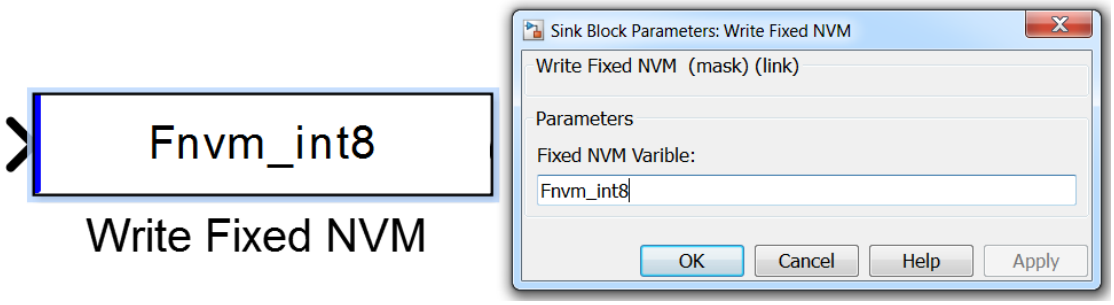
1) Fixed NVM Variable: Variable name.

**Block Output:**

The value of variables.

### 3.8.3 Write Fixed NVM

This module is used for writing fixed NVM variables.




**Block Parameters:**

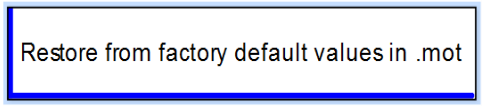
1) Fixed NVM Variable: Variable name.

**Block Output:**

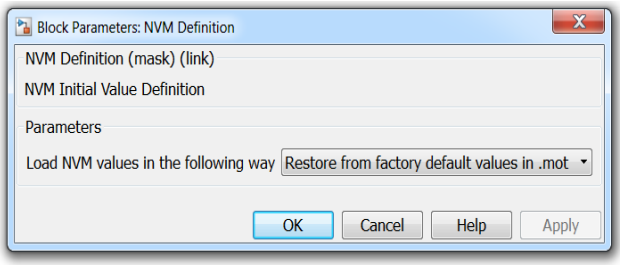
1) The value of variables.

### 3.8.4 NVM Definition

This module is used to select the initialization method of the non-fixed NVM.




NVM Definition

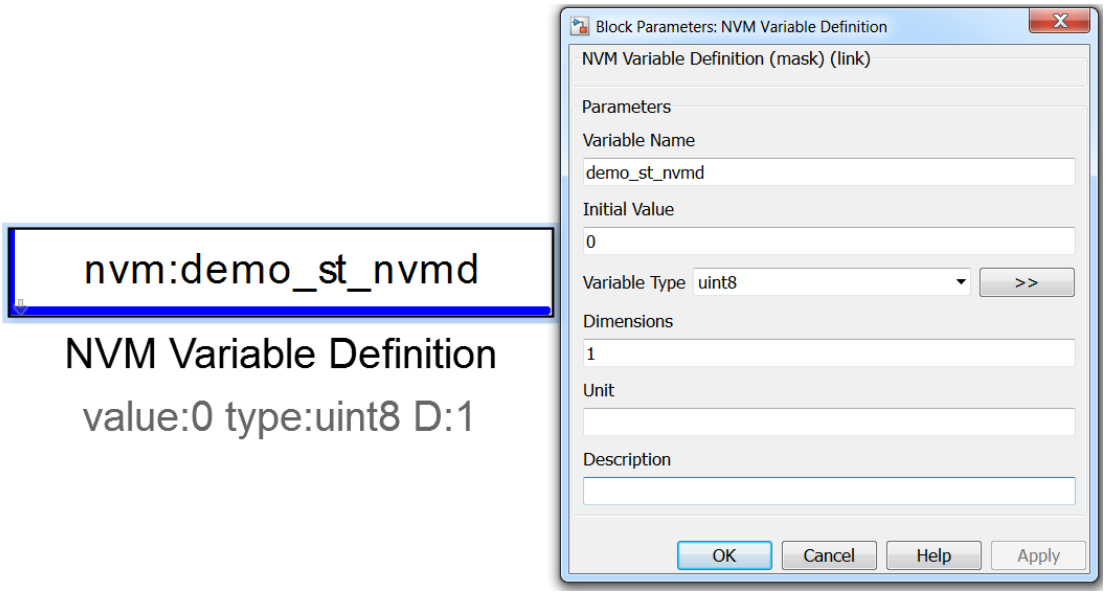



**Block Parameters:**

1) Load NVM value in the following way: Used to select the initialization method. If select Load previous saved values in flash, then the initial value of all non-fixed NVM variables after the code updated would be the value in Flash. If select Restore from factory default values in. mot, then all non-fixed NVM variables after the code updated, the initial value is the default value setting for variables.

### 3.8.5 NVM Variable Definition

This module is used to define NVM variables.




**Block Parameters:**

1) Variable_name: Variable name.

2) Init_value: Initial value.

3) Variable_type: The data type of the variable.

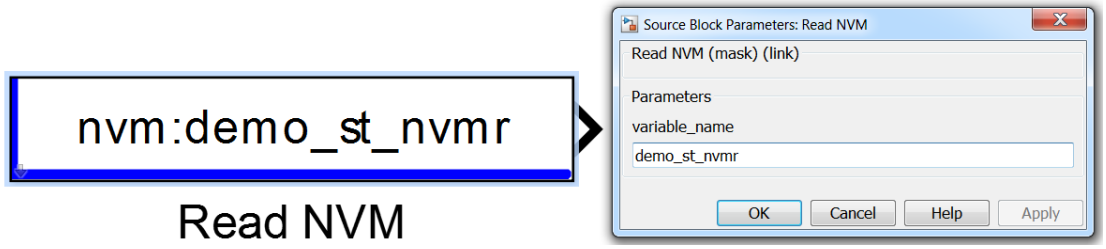
4) Dimensions: Data dimensions.

5) Unit: Unit.

6) Description: Description.

### 3.8.6 Read NVM

The module is used to read the values of NVM variables.




**Block Parameters:**

1) variable_name: Variable name.

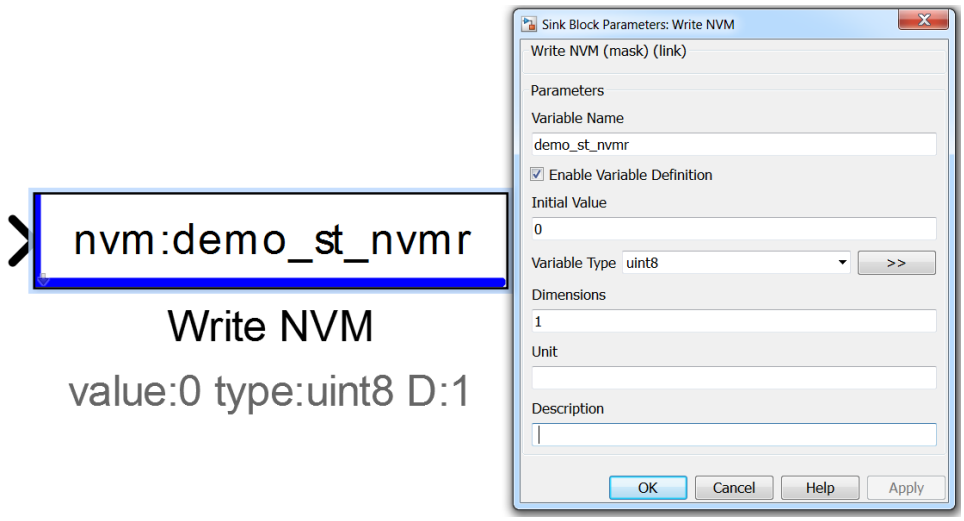
2) variable_type: The data type of the variable.

**Block Output:**

The NVM variable value

### 3.8.7 Write NVM

This module is used for the definition and write operation of NVM variables.




**Block Parameters：**

1) Variable_name: Variable name.

2) Enable Variable Definition: Whether the variable is defined to enable. If checked, in addition to the write function, there is also a variable definition function; if not checked, there is only the write function.

3) Init_value: Initial value.

4) Variable_type: The data type of the variable.

5) Dimensions: Data dimensions.

6) Unit: Unit.

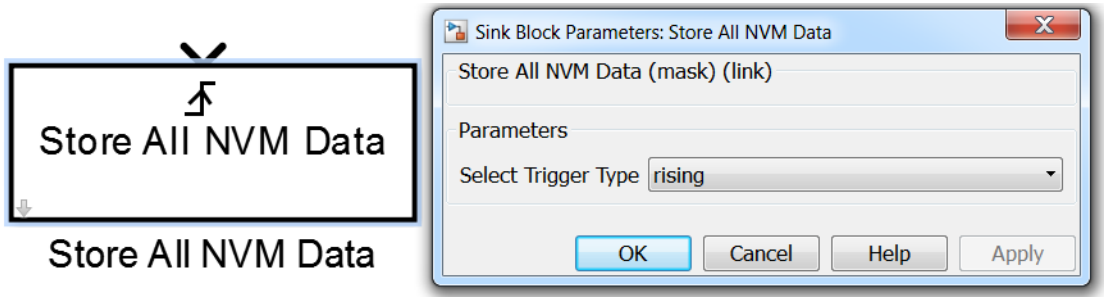
7) Description: Description.

**Input:**

1) The value of the variable.

### 3.8.8 Store All NVM Data

This module is used to store all fixed NVM variables and non-fixed NVM variables into non-volatile storage space.




**Parameter:**

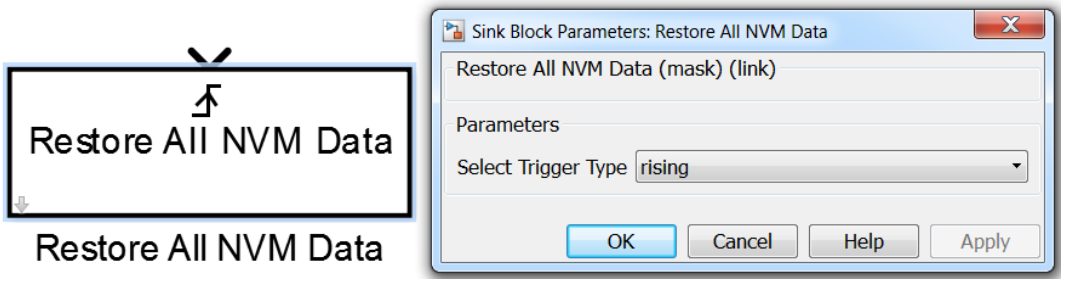
1) Select Trigger Type: Select the trigger type.

**input:**

1) Trigger signal.

### 3.8.9 Restore All NVM Data

This module is used to load all fixed NVM variables and non-fixed NVM variables into the RAM area.




**Parameter:**

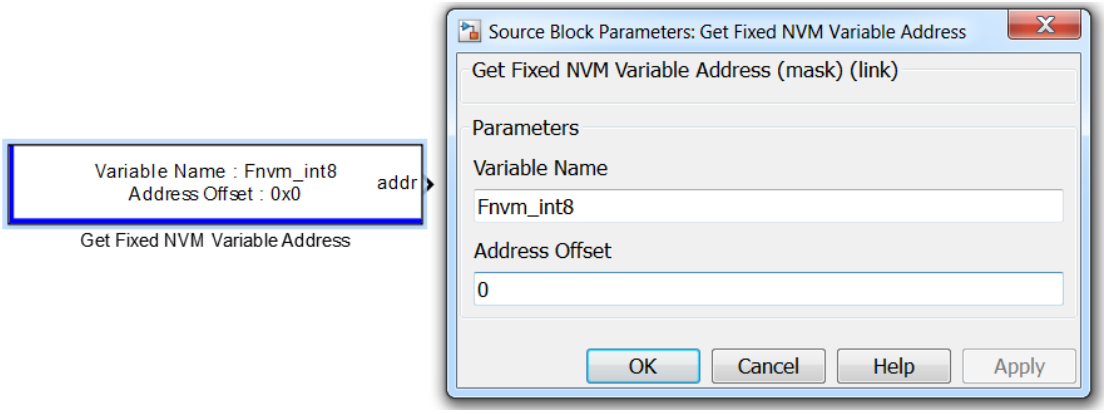
1) Select Trigger Type: Select the trigger type.

**Input:**

1) Trigger signal.

### 3.8.10 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:**

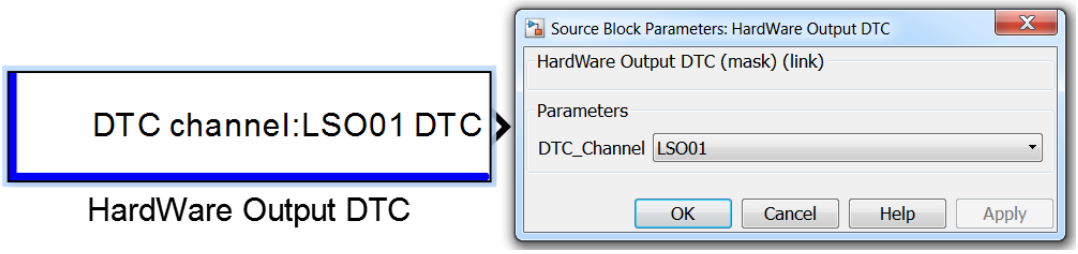
1) addr: the address of the variable

## 3.9 Diagnostic Blocks

Diagnostic blocks are designed to realize VCU diagnostic functions.

### 3.9.1 Hardware Output DTC

The module is used for the diagnosis of hardware high/low side and H-bridge drive modules, and can read the fault information of the hardware port, which is shown as DTC (Diagnostic Fault Code).




**Parameter:**

1) DTC_Channel: Troubleshooting channel selection.

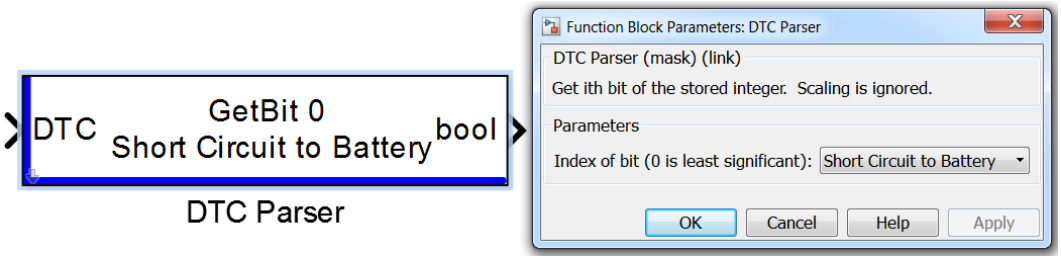
**Output:**

1) DTC: Fault code of the channel, and the result is one byte of data. The following table shows the failure types corresponding to each bit of the byte, and the corresponding bit is placed 1 means it has the failure.

| Fault Code bit | Fault type |
|---|---|
| Bit 0 | Short to power |
| Bit 1 | Short to ground |
| Bit 2 | Open circuit failure |
| Bit 3 | Overtemperature fault |
| Bit 4 | Power failure |
| Bit 5 | Load short fault |
| Bit 6 | Overload failure |
| Bit 7 | Other failure |

### 3.9.2 DTC Parser

This module is used to decode the hardware port fault code and determine the specific fault information represented by the fault code.




**Parameter:**

1) The type of failure.

**Input:**

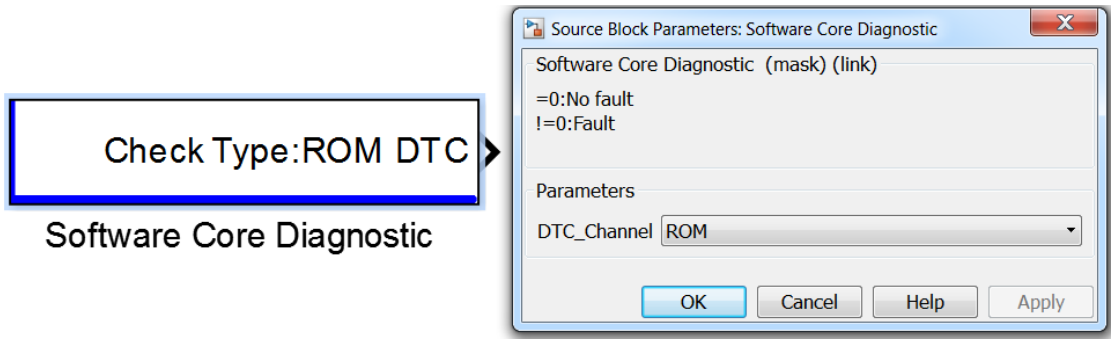
1) DTC: Fault code.

**Output:**

bool: 1 means that the failure occurred, and 0 means that the failure did not occur.

### 3.9.3 Software Core Diagnostic

This module is used for software kernel self-test to read the fault code of the corresponding module of the kernel.




parameter:

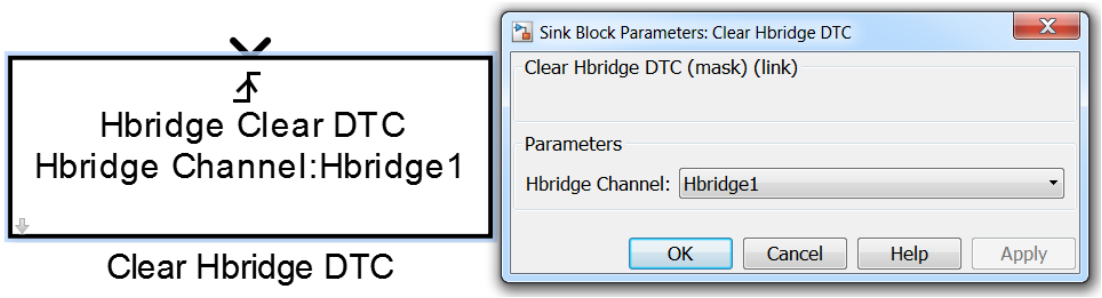
1) DTC_Channel: Module selection.

Output:

1) DTC: Fault code. 1 means that a failure occurred, and 0 means that it did not occur.

### 3.9.4 Clear H-bridge DTC

The module is used to clear the fault of the H-bridge channel, triggered by the rising edge.

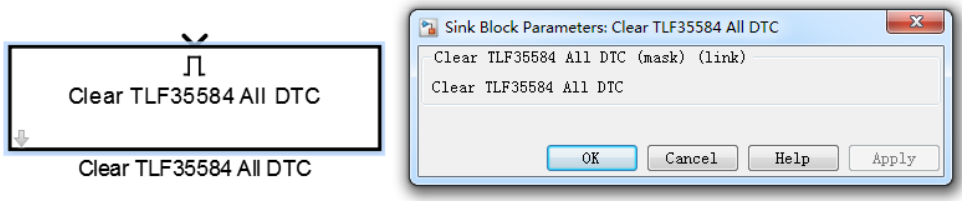



**Parameter:**

1) Hbridge Channel: Channel selection.

### 3.9.5 Clear TLF35584 All DTC

This module is used to clear all faults caused by TLF35584.

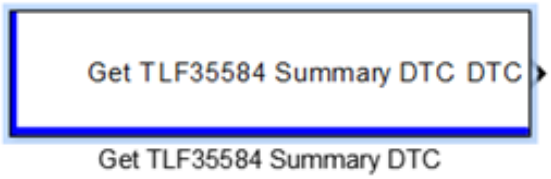



**Input:**

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

### 3.9.6 Get TLF35584 Summary DTC

This module is used to read the SUMMARY fault information of the TLF35584 module.



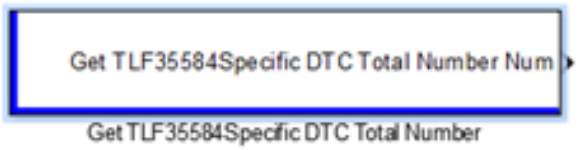

**Output:**

1) DTC: DTC summary from TLF35584.

Note: For more details, please refer to “TLF35584 Fault Code Table” from technical support

### 3.9.7 Get TLF35584Specific DTC Total Number

This module is used to read the number of fault codes that occur on the TLF35584 chip.



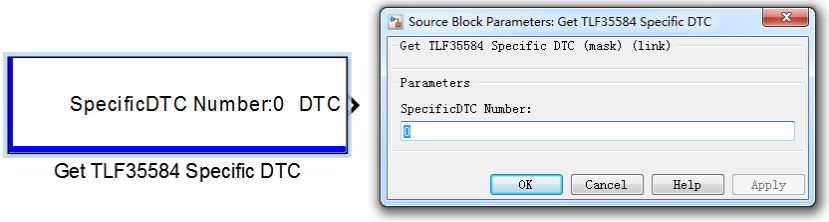

**Output:**

1) Num: the number of error code.

Note: For more details, please refer to “TLF35584 Fault Code Table” from technical support

### 3.9.8 Get TLF35584 Specific DTC

This module is used to get the specified fault code for the TLF35584 module.




**Parameter:**

1) SpecificDTC Number: Read the fault code of the specified DTC number, for example, read 5 fault codes from the "Get TLF35584Specific DTC Total Number" module. Here, if need to read the first fault code, you can set 0. If need to read all 5 fault codes, you need to drag and drop 5 modules, and set 0, 1, 2, 3, 4 in order.

Note:

1. Since the number of fault codes that occur on this chip is unknown, and the array of storage fault codes cannot be dynamically changed, the application layer needs to specify the length of the array when defining the array variable that reads the fault code of the property, for example, if it needs to read 10, it is necessary to define an array with a length of 10. The recommended length is 20.

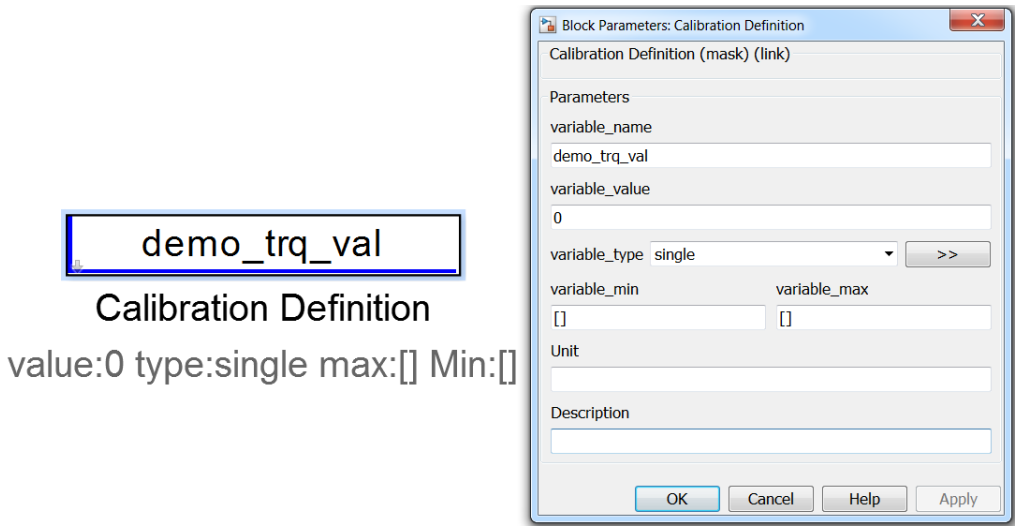
2. The fault code obtained by this module is the data in uint32, one data represents one fault code, if multiple faults are sent, multiple uint32 data are read in order. D31~ D0 represents Bit31 ~ Bit0, the fault code can be got according to the order.

3. For specified fault codes please refer to TLF35584 Fault Code Table from technical support.

## 3.10 Calibration & Measurement

### 3.10.1 Calibration Definition

This module is used to define calibration variables




**Block Parameters：**

1) variable_name: Variable name.

2) variable_value: Variable value.

3) variable_type: Variable type.

4) variable_min: The minimum value of the variable.

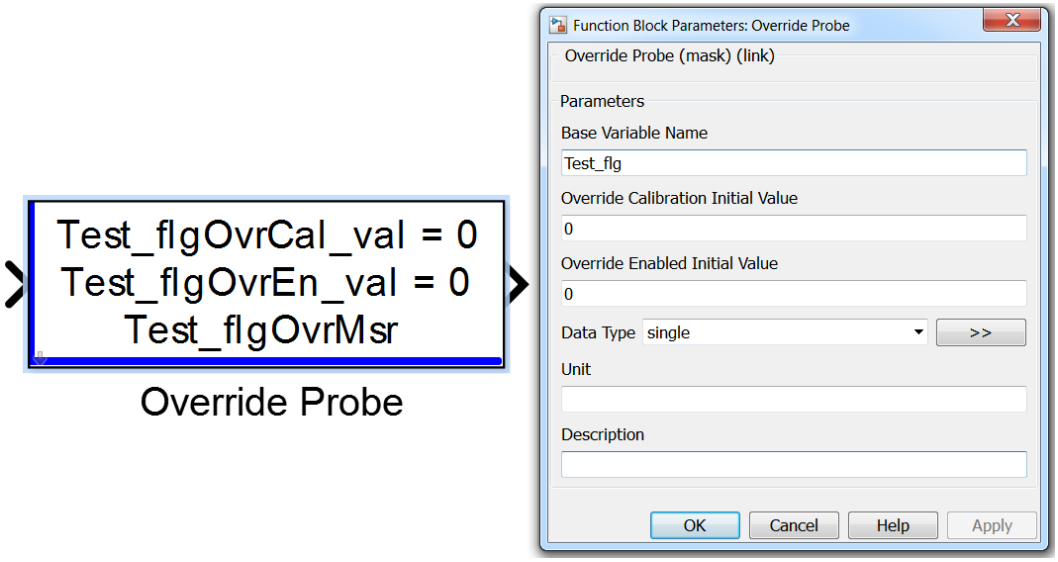
5) variable_max: The maximum value of the variable.

6) Unit: Unit.

7) Description: Description.

### 3.10.2 Override Probe

This module is used to add calibration variables to control output. Select whether the output variable is a calibration variable or measurement variable. "OvrMsr" is a measurement variable, "OvrCal_val" is a calibration variable, and "OvrEn_val" is a control variable.




**Parameter:**

1) Base Variable Name: Variable name.

2) Override Calibration Initial Value: The initial value of CALIBRATION VARIABLE.

3) Override Enable Initial Value: Control variable. If 1, output calibration variable, if 0, output measurement variable.

4) Data Type: Calibration variable type.

5) Unit: Unit.

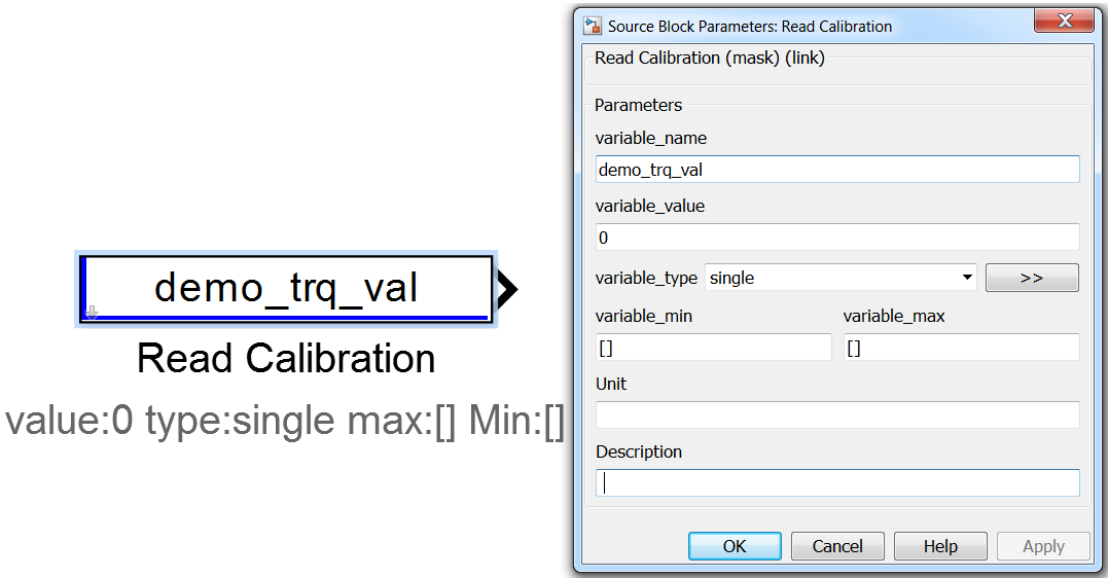
6) Description: Description.

**Output:**

1) Variable value.

### 3.10.3 Read Calibration

This module is used to define and read calibration variables.




**Parameter:**

1) variable_name: Variable name.

2) variable_value: Variable value.

3) variable_type: Variable type.

4) variable_min: The minimum value of the variable.

5) variable_max: The maximum value of the variable.

6) Unit: Unit.

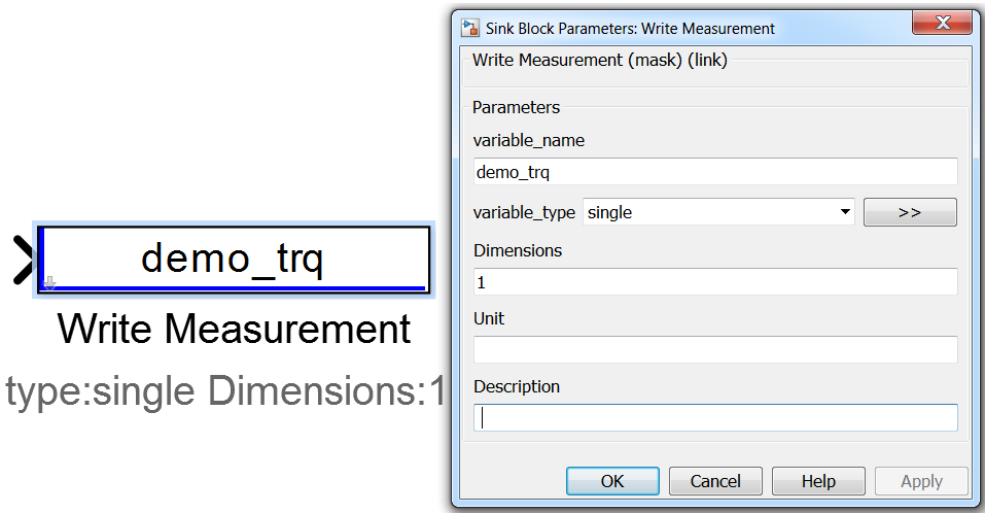
7) Description: Description.

**Output:**

1) Variable value.

### 3.10.4 Write Measurement

This module is used to define and update measurement variables.




**Block Parameters：**

1) variable_name: Variable name.

2) variable_type: Variable type.

3) Demisions: Variable dimension.

4) Unit: Unit.

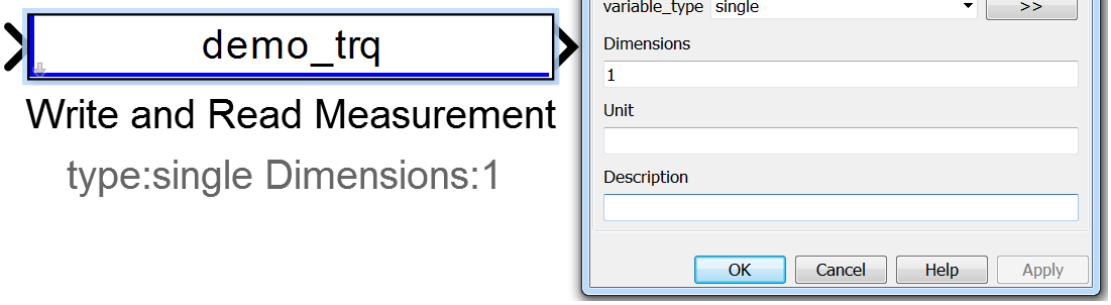
5) Description: Description.

**Block Input:**

1) Variable value.

### 3.10.5 Write and Read Measurement

This module is used to define and read measurement variables.




**Block Parameters:**

1) variable_name: Variable name.

2) variable_value: The initial value of the variable.

3) variable_type: Variable type.

4) Demisions: Variable dimension.

5) Unit: Unit.

6) Description: Description.

**Block Input:**

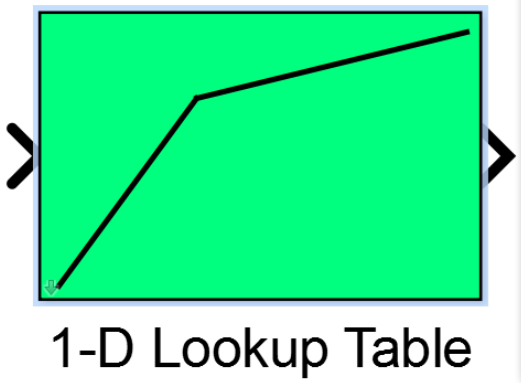
1) Variable value.

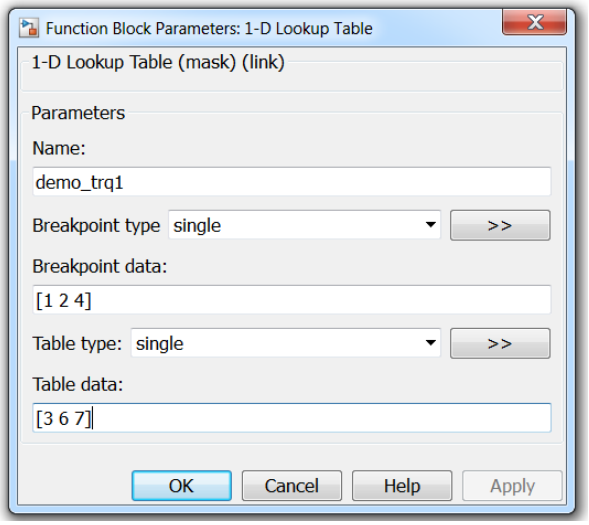
**Block Output:**

1) Variable value.

### 3.10.6 1-D Lookup Table

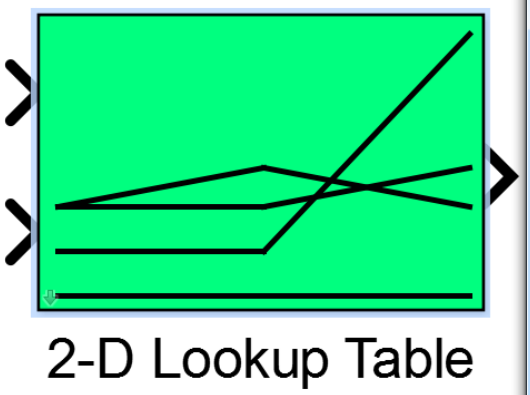
The module is used for the definition of the 1-D lookup table, without writing m variable, you can generate a calibration CUR. CUR name is “table name” + "_cur".

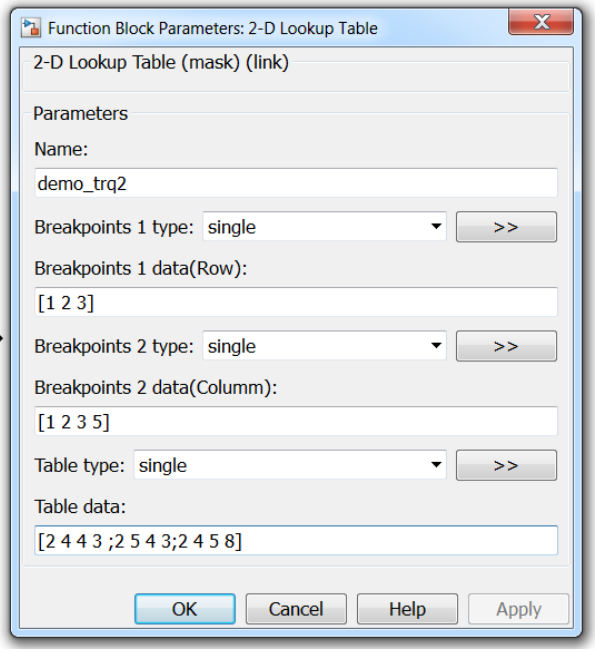



### 3.10.7 2-D Lookup Table

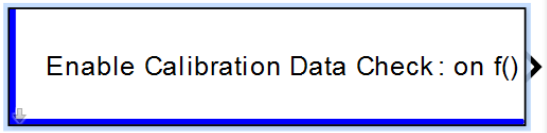
This module is used for the definition of 2-D lookup tables, without writing m variable, you can generate a calibration MAP. MAP name is table name + "_map".



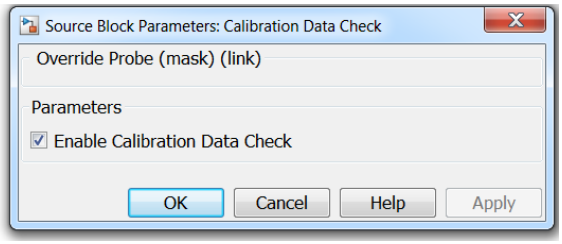



### 3.10.8 Calibration Data Check

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.




Calibration Data Check




**Block Parameters:**

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

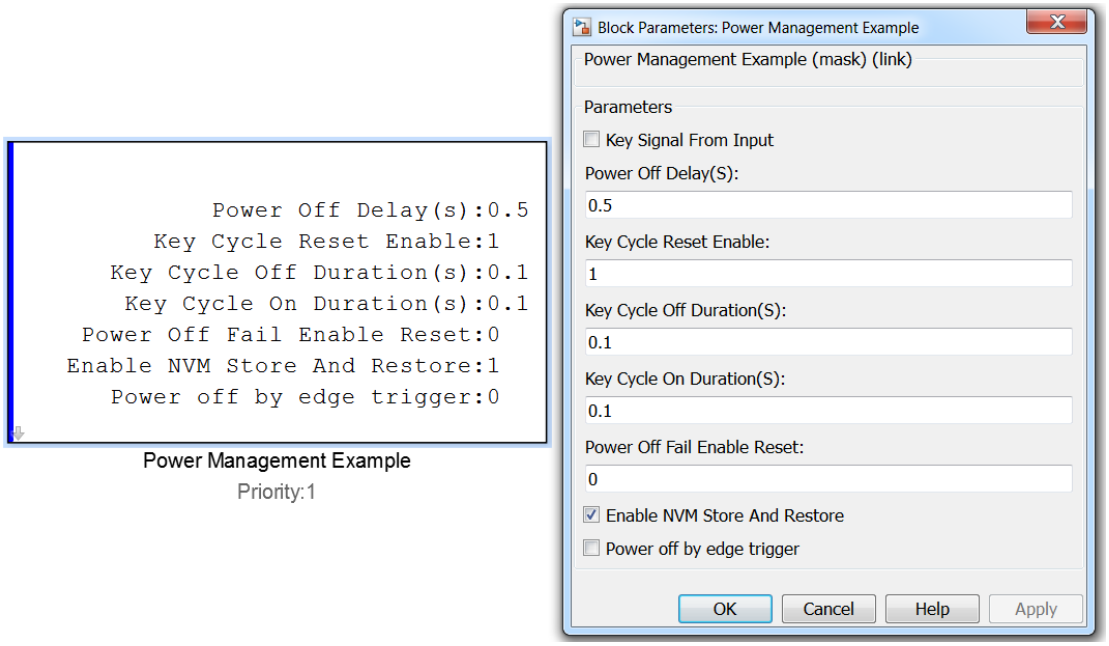
**Output:**

1) f() : task event

## 3.11 System Management Blocks

### 3.11.1 Power Management Example

Note: This module is used to control the system on and off, but does not include CAN wake-up or other non-key wake-up sources, so it is might not suitable for power off control in all cases. Users can refer to this module to build similar control model by themselves.




**Block Parameters:**

1) Key Signal From Input: Enables this option to read the key trigger signal

2) Power Off Delay: power off delay time setting, in second.

3) Key Cycle Reset Enable: Key Cycle reset enable. When the key is switched to off, but not power off, whether to reset when switching to ON again. Checking indicates that you want to reset; un-checking does not reset.

4) Key Cycle Off Duration: Key Off is considered Off after this duration.

5) Key Cycle On Duration: Key Off and then Key ON is considered ON after this duration.

6) Key AD2Volt Factor: When the key signal is acquired as an analog signal, the AD sample value is converted to the coefficient of the input voltage value. Its values are determined by the parameters in the controller technical manual "Table 2.2.1" and are calculated as follows:

Key AD2Volt Factor = (Divided Voltage Resistor + Serial Resistor) / (819 * Divided Voltage Resistor)

Note: This option is for the case when key signal is acquired as an analog signal. Ignore this setting if the key signal is a digital signal acquisition.

7) Key Off Threshold Volt(V): Power-off voltage threshold. When the external input voltage is less than this value, the system powers off.

8) Key On Hysteresis Volt(V): Power-on hysteresis return voltage. When the external input voltage is greater than (Key On Hysteresis Volt + Key Off Threshold Volt), the system powers on.

Note: When the system is powered on, the system will not power off as long as the input voltage is not lower than the Key Off Threshold Volt.

9) Power Off Fail Enable Reset: If 1, the VCU resets when power-off fails, and if it is 0, the VCU does not reset when power-off fails.

10) Enable NVM Store and Restore: Enables the NVM control option. If checked, all NVM variables are stored to Flash when powered down, and all NVM variables are loaded into Ram when powered on; if not checked, users need to build logic according to their own needs.

11) Set The Waiting Time(ms): Power-off timeout. If the power-off is not successful after this time, the power-off operation will not be performed.

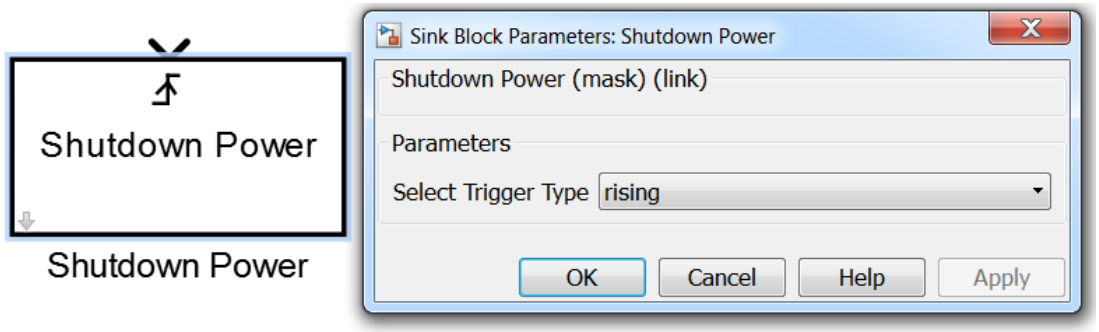
12) Power off by edge trigger: Power off trigger mode selection. If checked, the

falling edge of the key trigger signal will control to power down; if not checked, the key trigger signal will control power down.

Note: Different controllers need to configure slightly differently, here is the introduction of all the configuration parameters, please configure according to the window instructions.

### 3.11.2 Shutdown Power

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



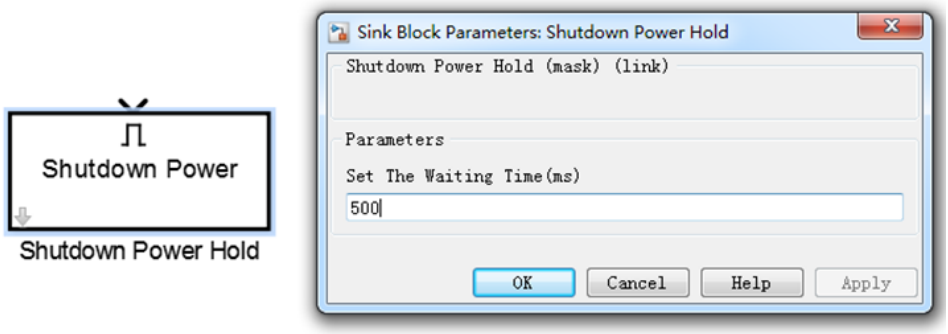

**Block Parameters:**

1) Trigger type: Trigger mode selection, including rising edge, falling edge, bilateral edge, function call four ways.

2) Set The Waiting Time(ms): Sets the power-down wait time in ms.

### 3.11.3 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:**

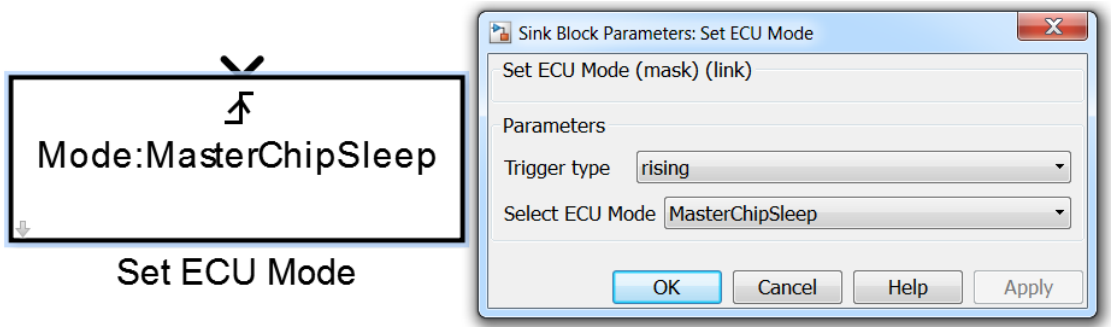
1) Trigger signal: set 1 to power off the controller.

**Note:**

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

### 3.11.4 Set ECU Mode

This module can set the working mode of ECU.




**Block Parameters:**

1) Trigger type: Trigger type.

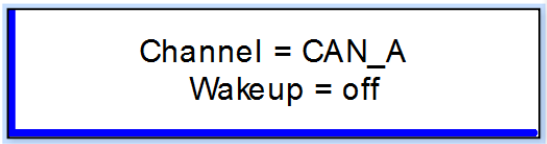
2) Select ECU Mode: Select the working mode. MasterChipSleep is the only selectable mode, which will put the master chip into sleep mode.

**Input:**

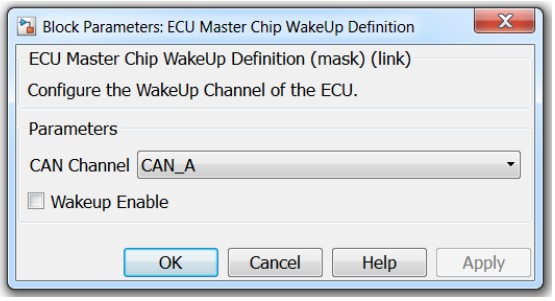
1) Trigger signal.

### 3.11.5 ECU Master Chip Wake-Up Definition

The module is used to wake up the master chip via CAN messages, switching it from sleep mode to working mode.




ECU Master Chip WakeUp Definition



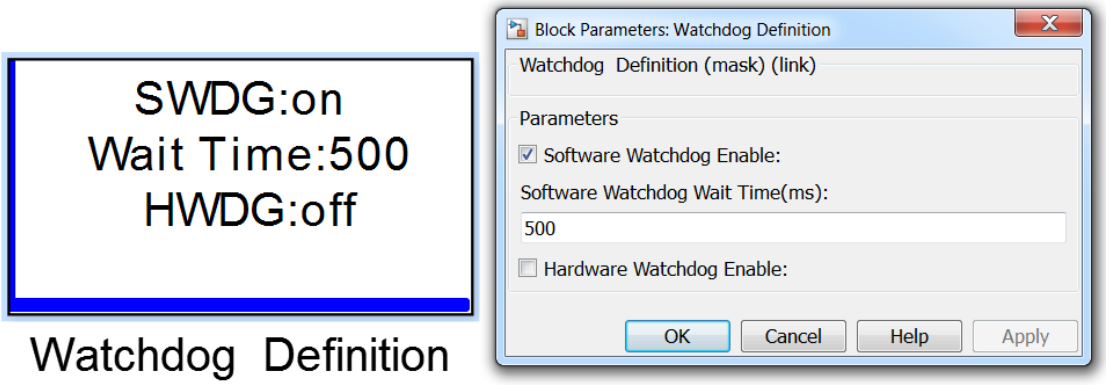

**Block Parameters:**

1) CAN Channel: Wake-up channel selection.

2) Wakeup Enable: Wake-up function enable setting. If checked, the selected channel has a wake-up function; if it is not checked, the selected channel does not have a wake-up function.

### 3.11.6 Watchdog Definition

Settings for software watchdog and hardware watchdog.




Settings for software watchdog and hardware watchdog.

**Block Parameters:**

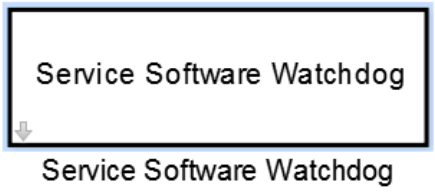
1) Software Watchdog Enable: Software watchdog enables control.

2) Software Watchdog Wait Time(ms): Wait time for feeding dogs, in ms. If the dog is not fed during the waiting time, a software reset will be performed.

3) Hardware Watchdog Enable: Hardware watchdog enable control, default dog feeding time is 600ms, perform hardware reset operation if timeout.

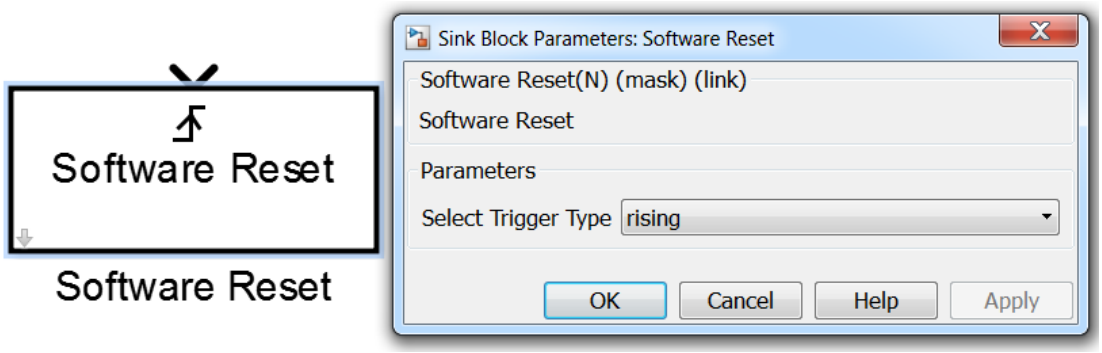
### 3.11.7 Service Software Watchdog

This module is used for software feeding dogs. The software watchdog needs to be enabled and put in the task schedule to periodically feeding dog.




### 3.11.8 Software Reset

This module is used for software reset.

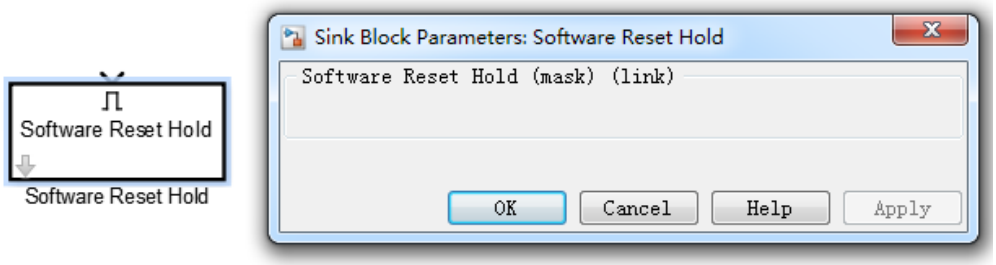



**Block Parameters:**

1) Select Trigger Type: Select the trigger type.

## 3.11.9 Software Reset Hold

This block is used for software reset.

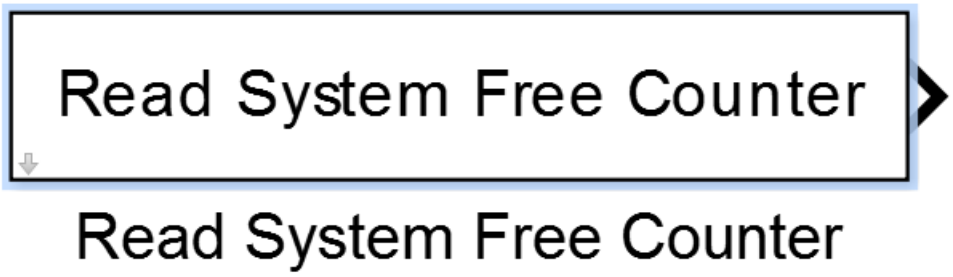



**Input:**

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

## 3.11.10 Read System Free Counter

The module is used to read the count value of the system free counter.



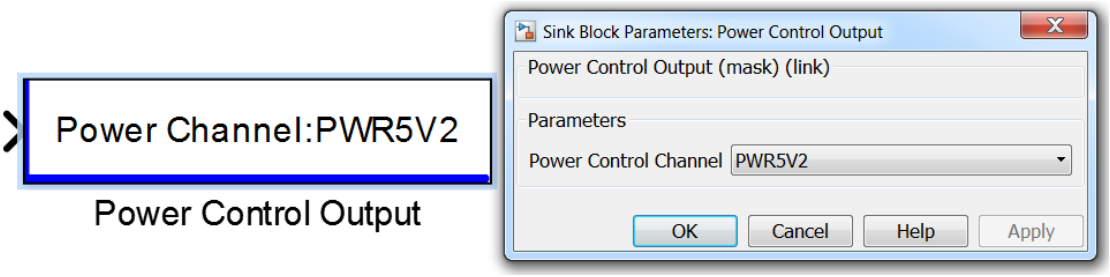

**Block Output:**

1) The count value of the system free counter, different controllers with different micro-chip have different counting frequency, as below:

| Micro-chip | Counting frequency |
|---|---|
| Infineon TC27x/TC29xx | 100MHz |
| NXP SPC57xx | 5MHz |
| NXP SPC56xx | 1MHz |

## 3.11.11 Power Control Output

This module is used to control the power output.




**Parameter:**

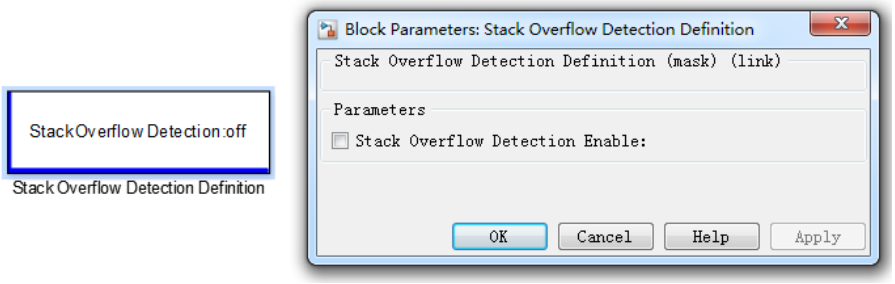
1) Power Control Channel: Channel selection.

**Input:**

1) 1 is powered on and 0 is powered off.

## 3.11.12 Stack Overflow Detection Definition

This block initializes the stack overflow detection.

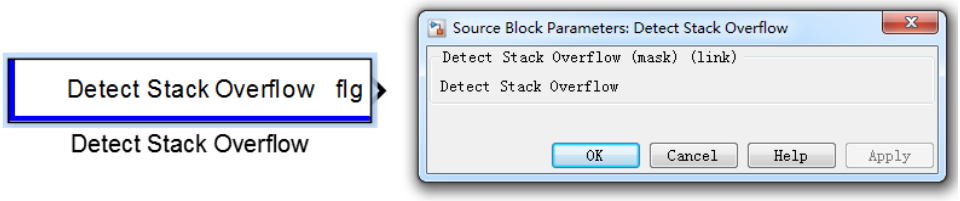



**Parameter:**

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

## 3.11.13 Detect Stack Overflow

This block is used for detecting stack overflow.

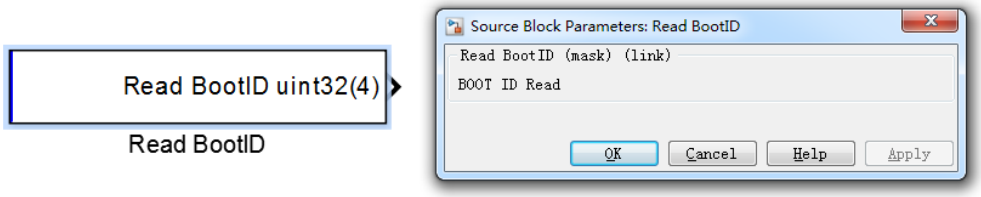



**Parameter:**

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

### 3.11.14 Read BootID

This block is used for reading the hardware serial number.

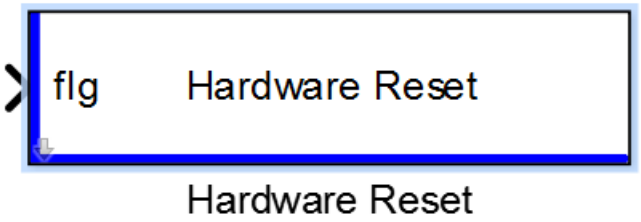



**Output:**

1) Uint32(4): 4 variables in unit32

### 3.11.15 Hardware Reset

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

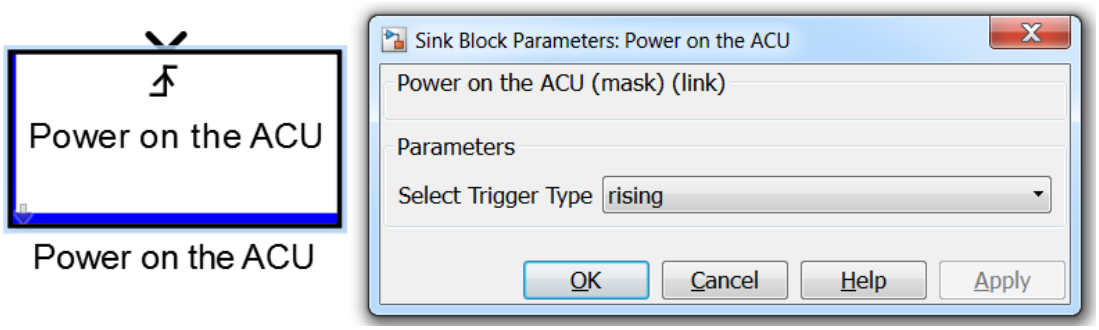



**Parameter:**

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

### 3.11.16 Power on the ACU

The module controls the Xavier power-on in controllers such as the EAXVA03 and EAXVA04, and the EAXVA03 can also be powered up via the module Power Control Output to select channel ACU.

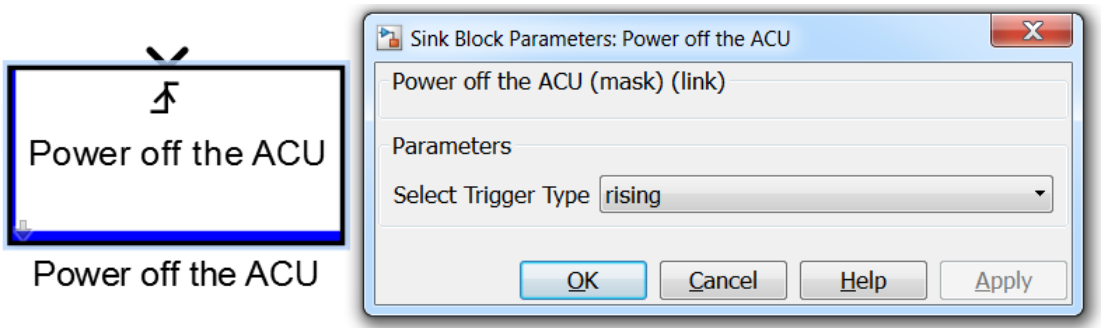



**Parameter:**

1) Trigger type: Trigger mode selection, including rising edge, falling edge, bilateral edge, function call four methods.

## 3.11.17 Power off the ACU

The module controls Xavier power-off within controllers such as the EAXVA03 and EAXVA04, and the EAXVA03 can also be powered down via the module Power Control Output to select channel ACU.

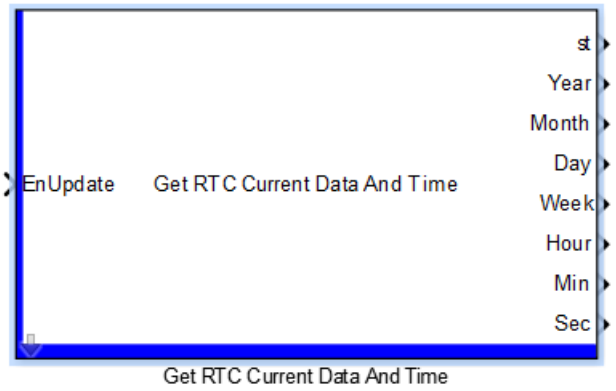



**Parameter:**

1) Trigger type: Trigger mode selection, including rising edge, falling edge, bilateral edge, function call four methods.

## 3.11.18 Get RTC Current Data And Time

This module allows you to read the current date and time in the RTC.




**Input:**

1) EnUpdate: Communicating with the RTC if 1. It is recommended to trigger it in edges or a large period ( do not always be 1), because communication with the RTC will takes up long task time.

**Output:**

1) st: Read and set the RTC result status, 0 is Success, other values mean Failed.

2) Year: Year, the valid range is 0-99.

3) Month: Month.

4) Day: Day.

5) Weeak: Week.

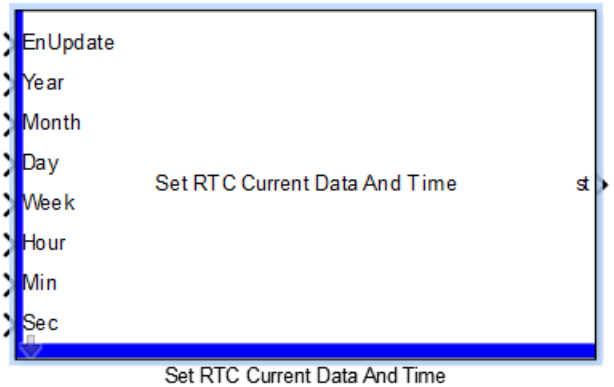
6) Hour: Hours.

7) Min: Min.

8) Sec: Seconds.

### 3.11.19 Set RTC Current Data And Time

This module allows you to set the current date and time of the RTC.



Set RTC Current Data And Time

**Input:**

1) EnUpdate: Communicating with the RTC if 1. It is recommended to trigger it in edges or a large period (do not always be 1), because communication with the RTC will takes long task time.

2) Year: Year, the valid range is 0-99.

3) Month: Month.

4) Day: Day.

5) Weeak: Week.

6) Hour: Hours.

7) Min: Min.

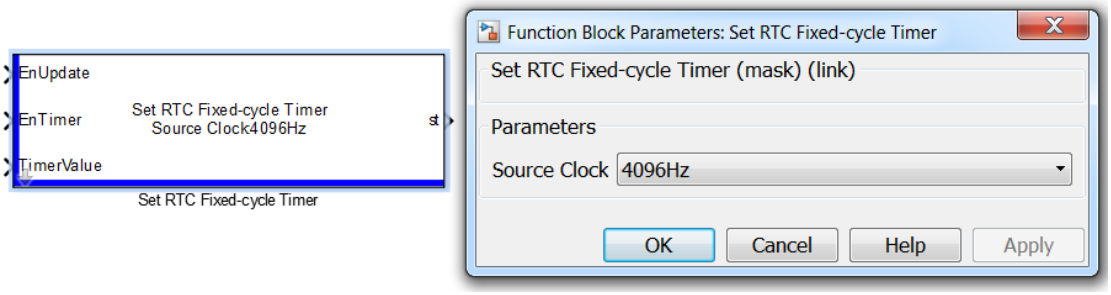
8) Sec: Seconds.

**Output:**

1) st: Read and set the RTC result status, 0 is Success, other values are Failed.

## 3.11.20 Set RTC Fixed-cycle Timer

Through this module, you can set the RTC to wake up the controller at regular intervals, and the timing period is determined by the input port TimerValue and the parameter Source Clock together. When enabled, it will periodically generate a pulse signal that wakes up the controller, as long as it is not turned off.




**Parameter:**

1) Source Clock: Clock source selection, there are four options, namely 4096Hz, 64Hz, SecondUpdate and MinuteUpdate. When in different options, input port TimerValue can set the time range as shown in the following figure:

| TimerValue | Source clock | | | |
|---|---|---|---|---|
| | 4096 Hz | 64 Hz | "Second" update | "Minute" update |
| 0 | – | – | – | – |
| 1 | 244.14 µs | 15.625 ms | 1 s | 1 min |
| 2 | 488.28 µs | 31.25 ms | 2 s | 2 min |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 41 | 10.010 ms | 640.63 ms | 41 s | 41 min |
| 205 | 50.049 ms | 3.203 s | 205 s | 205 min |
| 410 | 100.10 ms | 6.406 s | 410 s | 410 min |
| 2048 | 500.00 ms | 32.000 s | 2048 s | 2048 min |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 4095 | 0.9998 s | 63.984 s | 4095 s | 4095 min |

**Input:**

1) EnUpdate: communicating with the RTC IF 1, it is recommended to trigger it in edges or a large period (do not always be 1), because communication with the RTC will takes up long task time.

2) EnTimer: Cycle timing switch, 1 is on, 0 is off.

3) TimerValue: set time value, the value valid range is 1-4095, 0 is an invalid value, if it is greater than 4095 then it is 4095.

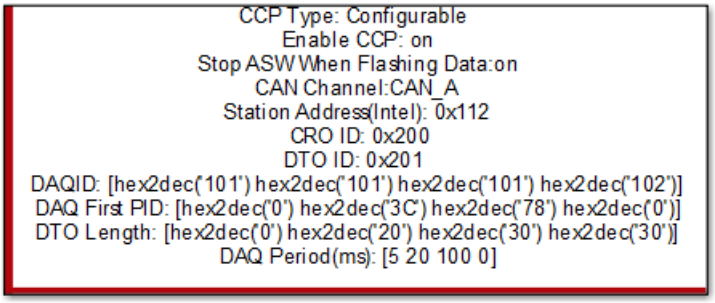
**Output:**

1) st: Read and set the RTC result status, 0 is Success, other values are Failed.

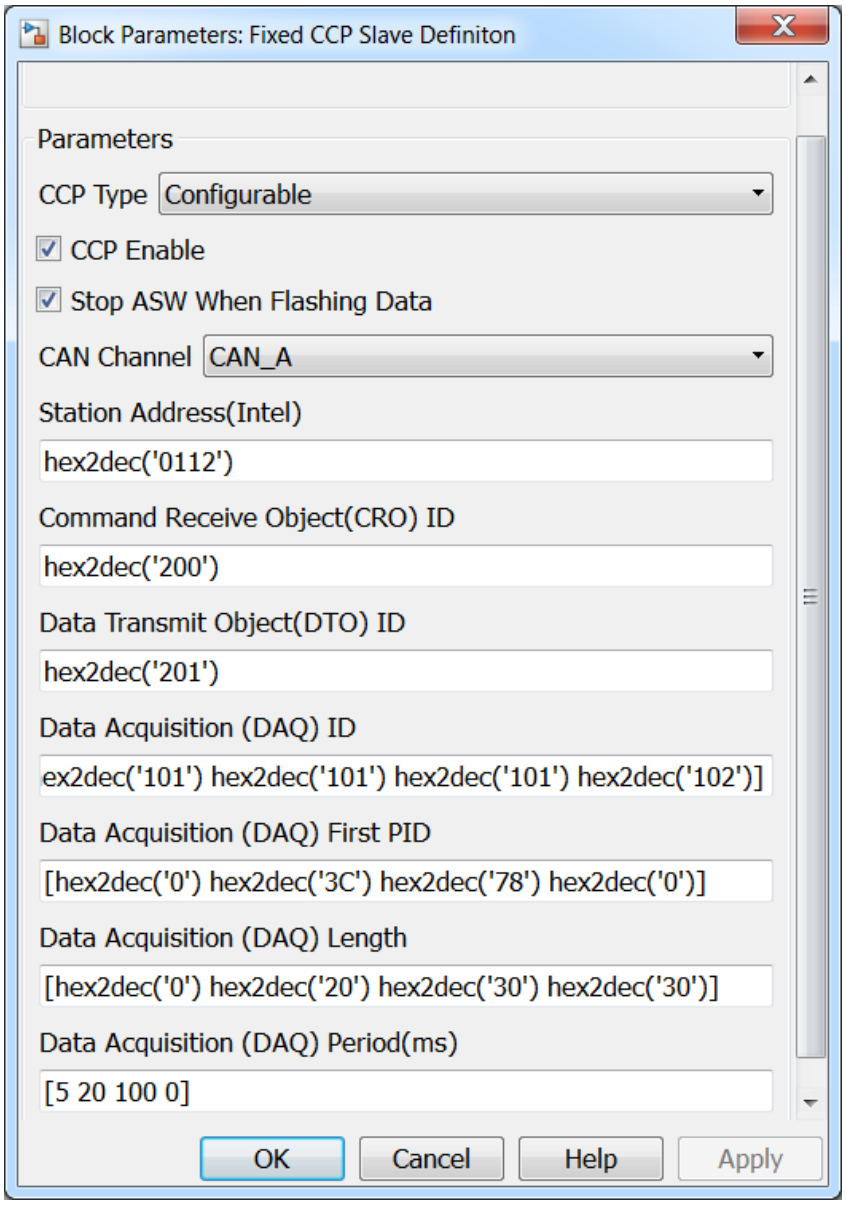
## 3.12 CCP

### 3.12.1 Fixed CCP Slave Definition

This module is used for CCP initialization setup. Only by dragging and dropping the module can be calibrated and measured via calibration software (EcoCAL, INCA, etc.).




Fixed CCP Slave Definiton




**Block Parameters:**

1) CCP Type: CCP type, divided into Simple and Referenceable. When selecting Simple to use EcoCAL for calibration and measurement, you need to load a2l and cal files. When you select Configurable, it will not generate cal file, need to load a2l and mot/hex files.

2) CCP Enable: CCP Enable.

3) Stop ASW When Flashing Data: Stops the application layer when flashing data.

4) CAN Channel: CAN channel selection.

5) Station Address (Intel): Station address settings.

6) Command Receive Object(CRO) ID: The address where the master sends the command.

7) Data Transmit Object(DTO) ID: The address of the slave reply command.

8) Data Acquisition (DAQ) ID: The address where the measurement data is sent from the slave.

9) Data Acquisition (DAQ) First PID: The starting address from which the measurement data is sent from the slave.

10) Data Acquisition (DAQ) Length: The length of the measurement data sent from the slave.

11) Data Acquisition (DAQ) Period(ms): The period in which measurement data is sent from the slave.

Note:

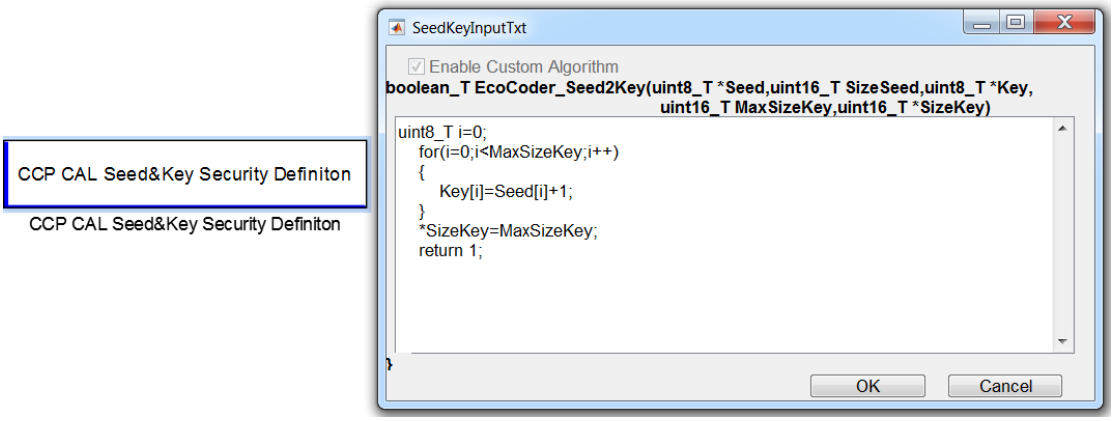
1. When selecting Configurable to use EcoCAL for calibration and measurement, EcoCAL 2.x.x.x version is required.

2. When calibrating and measuring with INCA, must select Configurable.

3. So far, up to 4 channels of DAQ lists are supported, and ignore this channel when the period or length is 0.

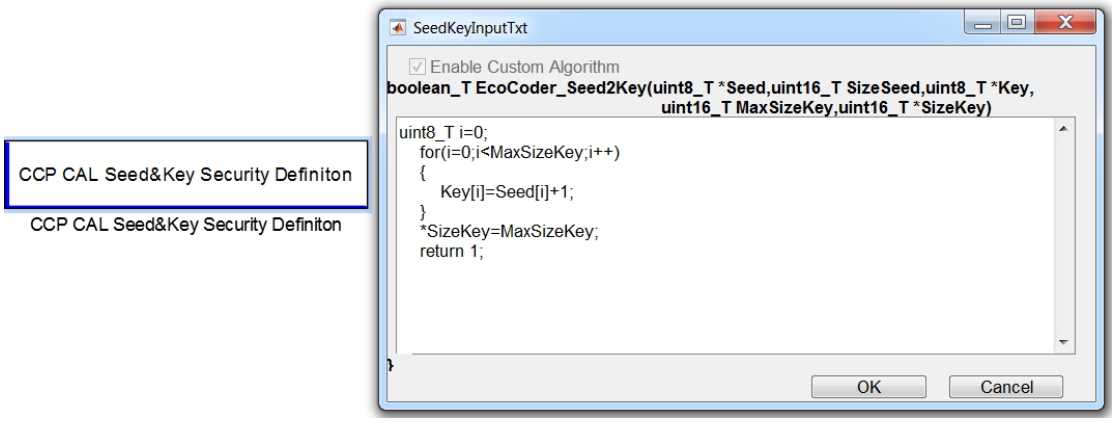
### 3.12.2 CCP/CAL Seed&Key Security Definition

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".



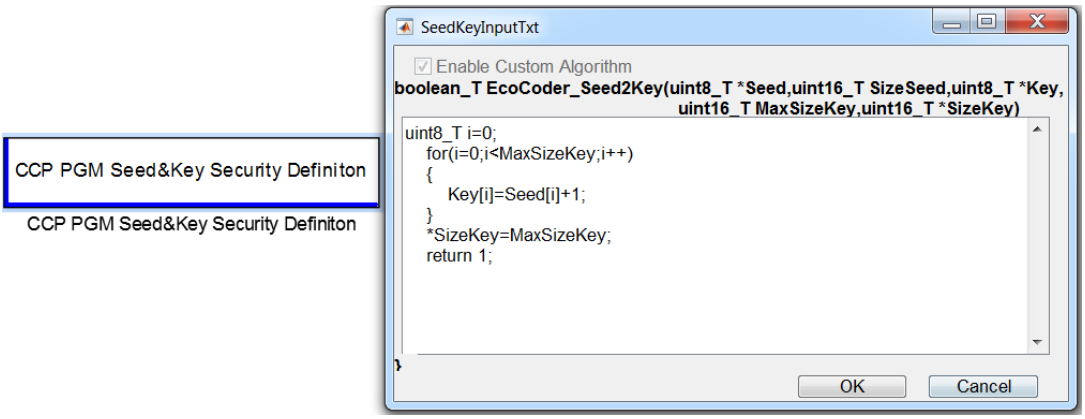

## 3.12.3 CCP DAQ Seed&Key Security Definition

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".



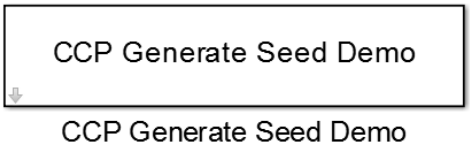

## 3.12.4 CCP PGM Seed&Key Security Definition

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".




### 3.12.5 CCP Generate Seed Demo

The module is an example of generating Seed, implemented via CCP Get Seed Trigger, CCP Set Seed, and some other algorithms.

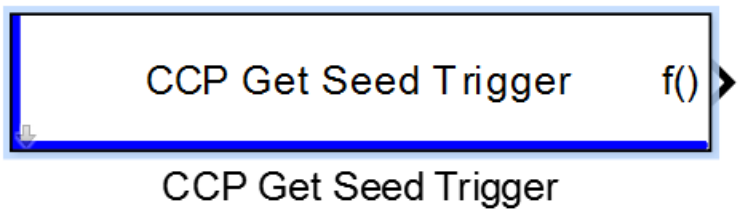



Note:

1. The module is only valid if the CCP type is Configurable.
2. This module is built for the convenience of users to understand and use. Users can customize it according to their own needs.

### 3.12.6 CCP Get Seed Trigger

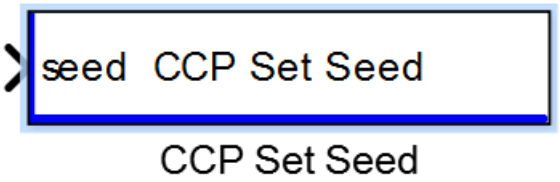
This module is used to trigger the get seed task.



Note: This module is only valid if the CCP type is Configurable. To use it, refer to the CCP Generate Seed Demo module.

### 3.12.7 CCP Set Seed

This module is used to set up Seed.



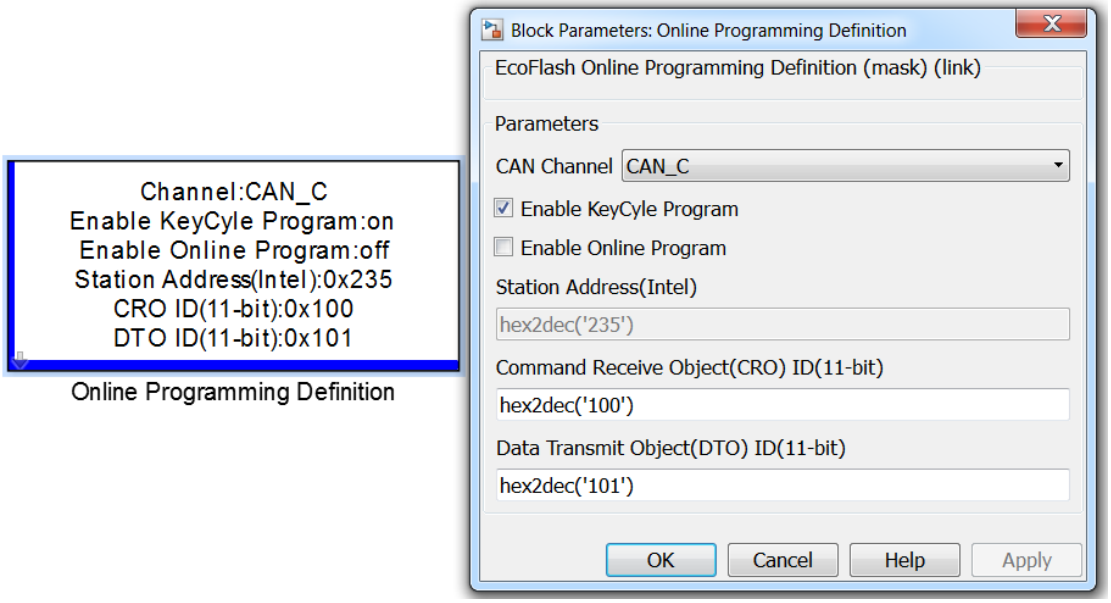

Note: This module is only valid if the CCP type is Configurable. To use it, refer to the CCP Generate Seed Demo module.

## 3.13 Programming Blocks

The flashing modules in this chapter can change the code and related parameters online, and can only be used if the hardware supports this function, if it is not supported, it will pop up and indicated that it is not supported.

### 3.13.1 Online Programming Definition

This module is used for online flashing programs, can also change the flashing parameters.




**Block Parameters**

1) CAN Channel: flashing channel selection.

2) Enable KeyCycle Program: Enables the key switch to update program, if checked, you can use the key switch to update program.

3) Enable Online Program: Enables online update programs, if checked, you can flash programs online.

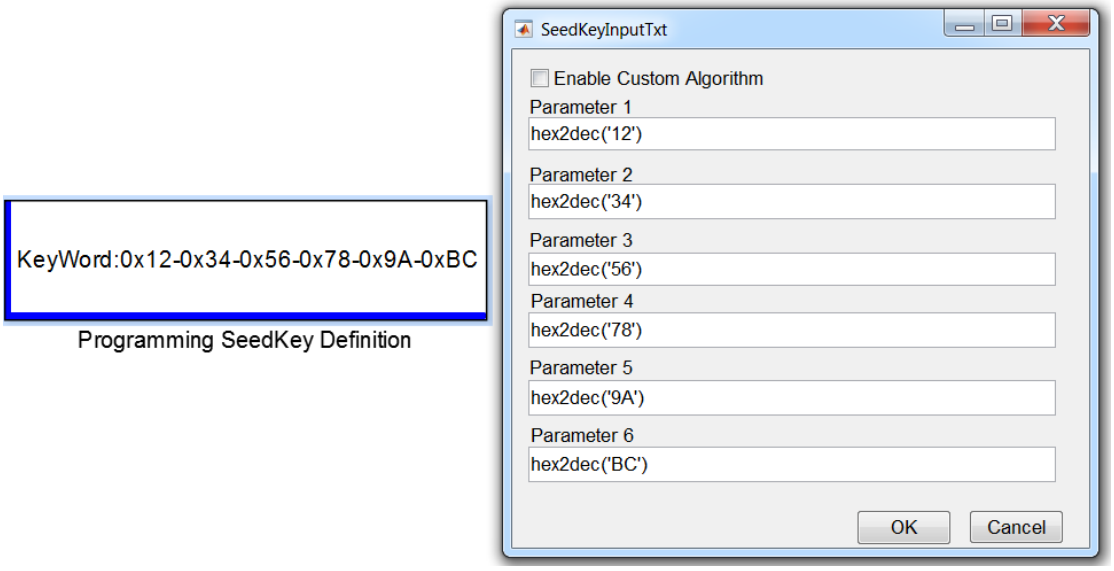
4) Station Address: Slave station address, cannot be changed.

5) Command Receive Object(CRO) ID: The address of the CRO.

6) Data Transmit Object(DTO) ID: The address of the DTO.

### 3.13.2 Programming Seed&Key Definition

This module is used to encrypt flashing files and must be dragged and dropped when using the Online Programming Definition module.




Users can define or modify the flashing key themselves. If you do not drag the module, the default parameters (0x12-0x34-0x56-0x78-0x9A-0xBC) will be used; if you drag and drop the module to set parameters, it will use the key set by the module to flash.

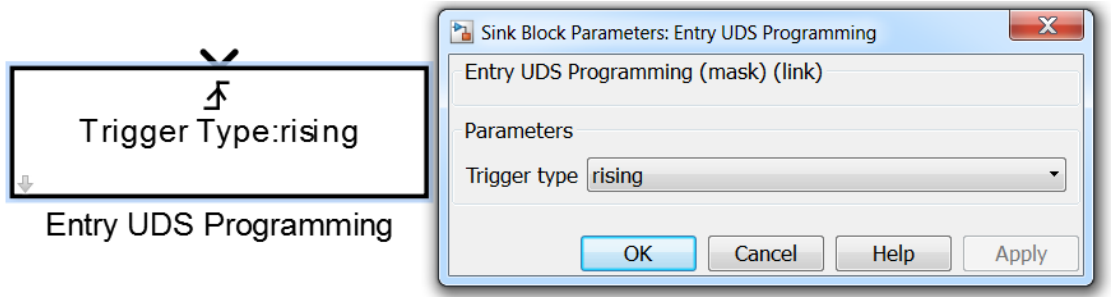
**Parameters**:

1) Enable Custom Algorithm: Enabled user-defined algorithms. If checked, the key algorithm can be defined in c, and if not checked, the key can be defined by setting parameters.

2) KeyWord1 ~ KeyWord6: flashing key settings, 6 bytes key.

### 3.13.3 Entry UDS Programming

This module is used for the main program to enter UDS flashing mode. Only when controller supports UDS flashing can use this function.



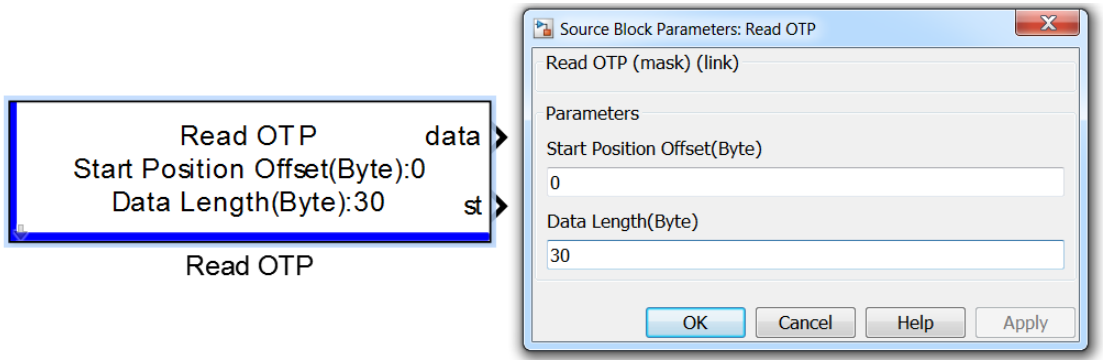
**Parameter:**

1) Trigger type: Select the trigger type.

## 3.14 Advanced Data Blocks

### 3.14.1 Read OTP

The module is used to read data from the OTP area.




**Parameter:**

1) Start Position Offset(Byte): The offset of the start address in bytes.

2) Data Length(Byte): The length of the data, in bytes.

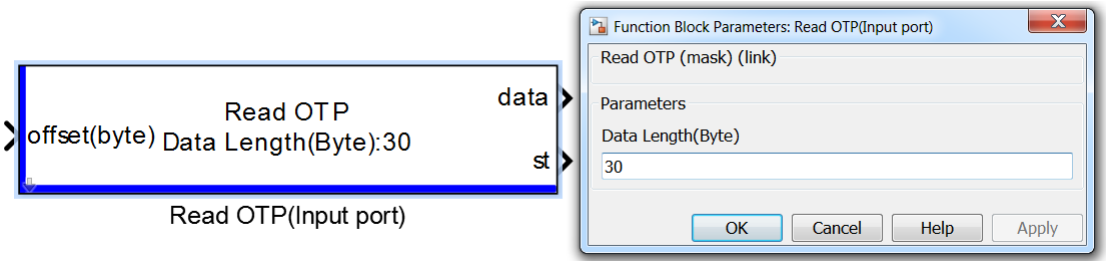
**Output:**

1) data: The data to be read.

2) st: Read status. 0 means read success, non-0 means read failure.

### 3.14.2 Read OTP (Input port)

The module is used to read data from the OTP area.




**Parameter:**

1) Data Length(Byte): The length of the data, in bytes.

**Input:**

1) offset(Byte): The offset of the starting address in bytes.

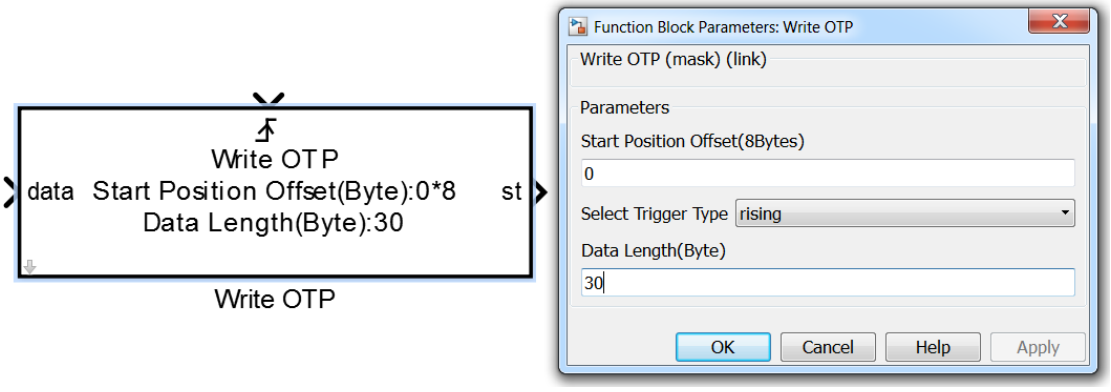
**Output:**

1) data: The data TO BE read.

2) st: Read status. 0 means read success, non-0 means read failure.

### 3.14.3 Write OTP

This module is used to write data to the OTP area.




**Parameter:**

1) Start Position Offset (8Bytes): The offset of the start address in 8 bytes.

2) Select Trigger Type: Select the trigger type.

3) Data Length(Byte): The length of the data, in bytes.

**Input:**

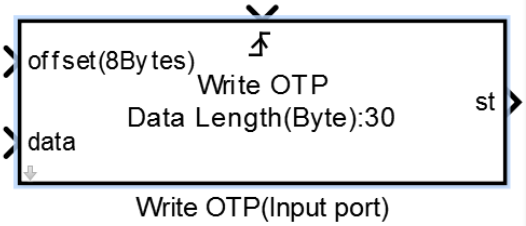
1) data: The data to be written.

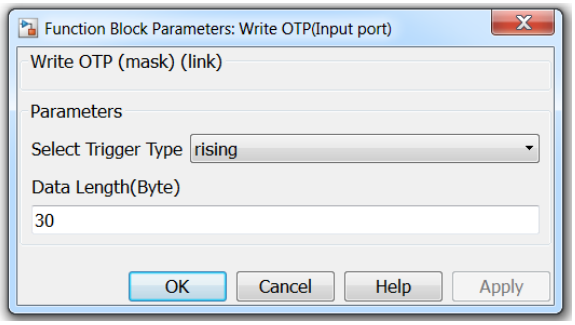
**Output:**

1) st: Write status. 0 means write success and a non-0 means write failure.

### 3.14.4 Write OTP (Input port)

This module is used to write data to the OTP area.




**Parameter:**

1) Select Trigger Type: Select the trigger type.

2) Data Length(Byte): The length of the data, in bytes.

**Input:**

1) offset(8Bytes): The starting address offset in 8 bytes.

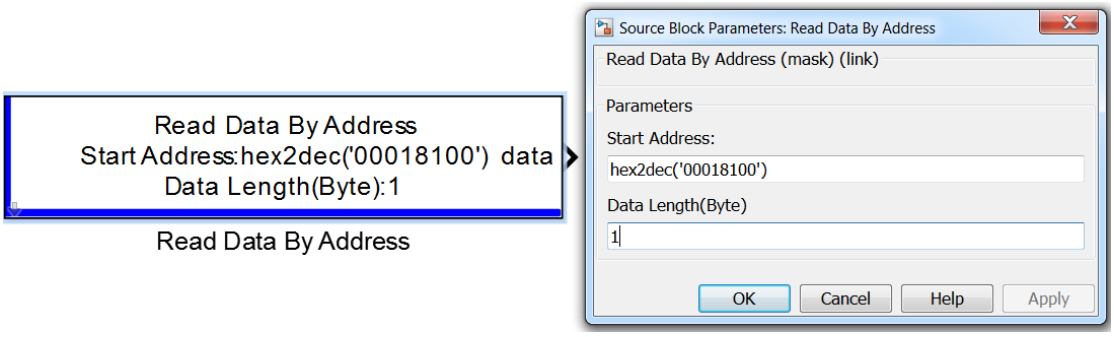
2) data: The data to be written.

**Output:**

1) st: Write status. 0 mneas write success and a non-0 means write failure.

### 3.14.5 Read Data by Address

The module is used to read data for a specific address.




**Parameter:**

1) Start Address: The start address.

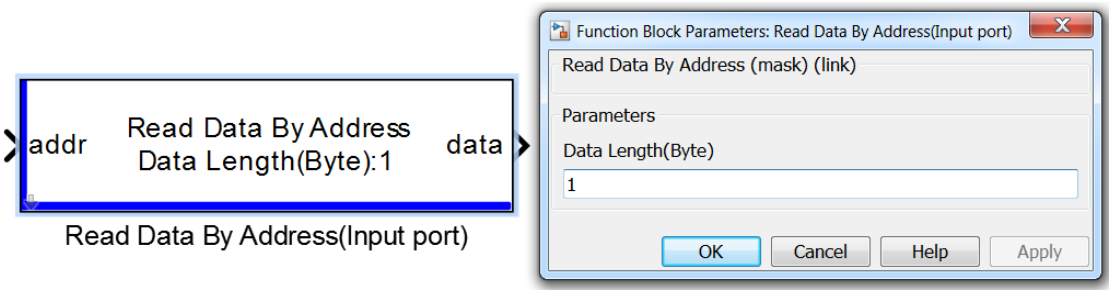
2) Data Length(Byte): The length of the data, in bytes.

**Output:**

1) data: The data to be read.

## 3.14.6 Read Data by Address (Input port)

The module is used to read data for a specific address.




**Parameter:**

1) Data Length(Byte): The length of the data, in bytes.

**Input:**

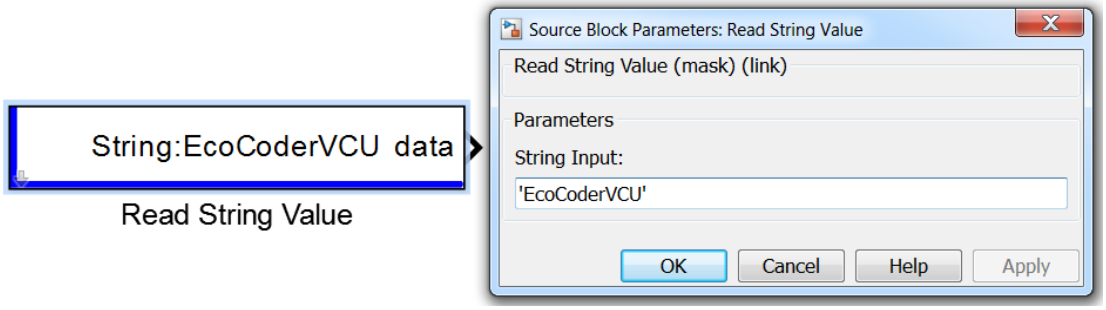
1) addr: The start address.

**Output:**

1) data: The data to be read.

### 3.14.7 Read String Value

This module is used to read the ASCII value for certain string.




**Parameter:**

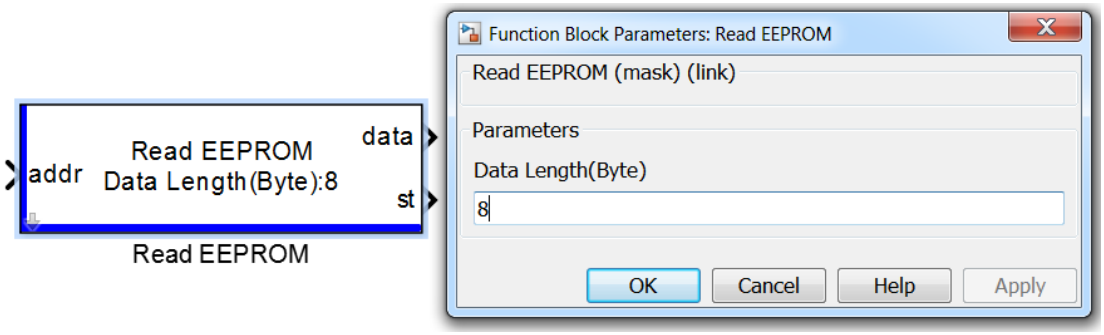
1) String Input: The string entered.

**Output:**

1) data: An array of ASCII values for the input string.

### 3.14.8 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

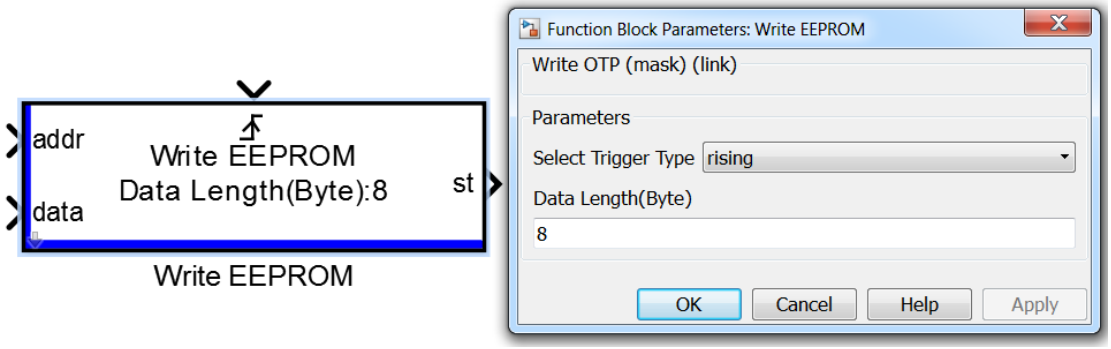
**Output:**

1) data: data read from EEPROM

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

### 3.14.9 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

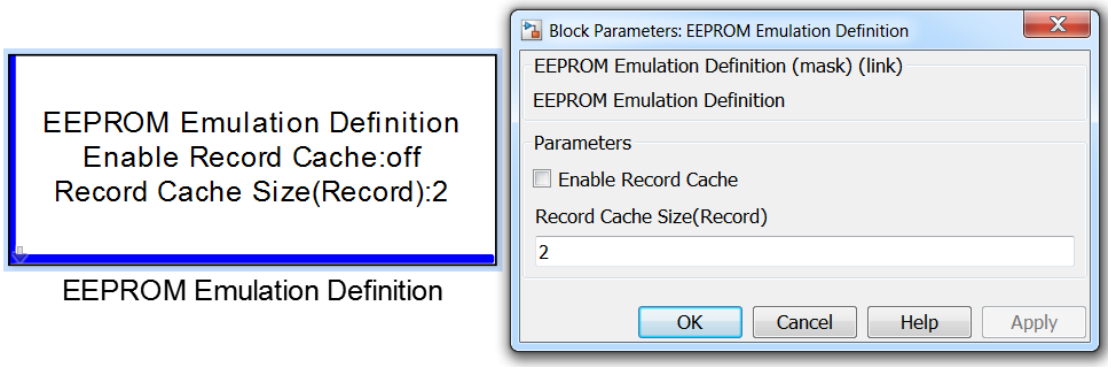
2) Data: the data to be written

**Output:**

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

### 3.14.10 EEPROM Emulation Definition

This module is used to set up the emulation EEPROM.



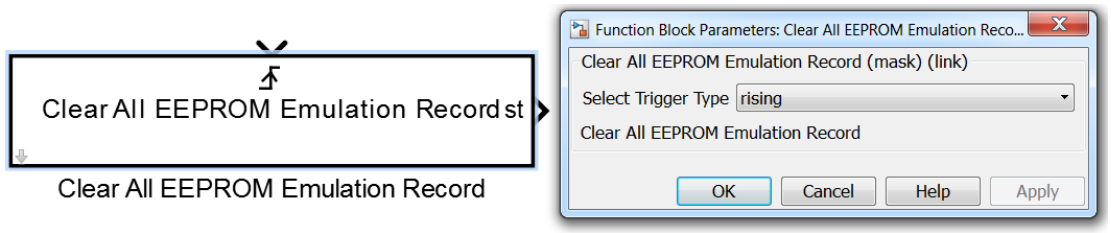

**Parameter:**

1) Enable Record Cache: Cache enable setting. If enabled, record information can be cached.

2) Record Cache Size(Record): The number of data records to be cached.

## 3.14.11 Clear ALL EEPROM Emulation Record

This module is used to clear the records in all analog EEPROM.




**Parameter:**

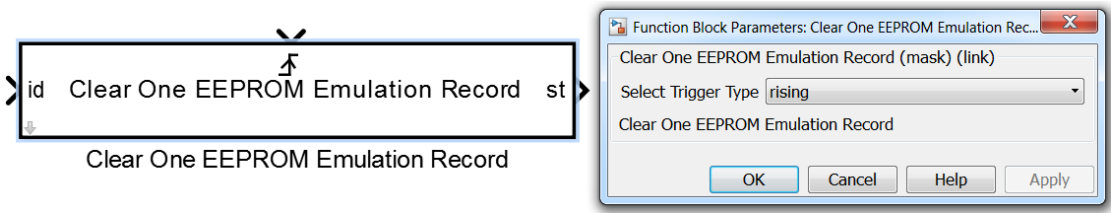
1) Select Trigger Type: Select the trigger type.

**output:**

1) st: output status, refer to the appendix table.

## 3.14.12 Clear One EEPROM Emulation Record

This module is used to clear individual records in the analog EEPROM.




**Parameter:**

1) Select Trigger Type: Select the trigger type.

**Input:**

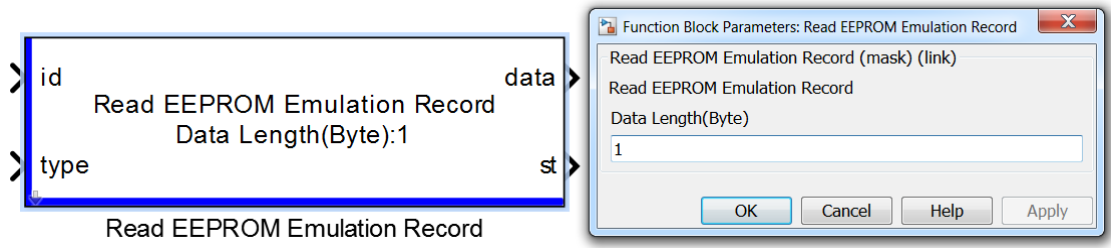
1) id: The address to be cleared for the record.

**Output:**

1) st: output status, refer to the appendix table.

## 3.14.13 Read EEPROM Emulation Record

The module is used to read the records of the emulation EEPROM.




**Parameter:**

1) Data Length: The length of the data in the record to be read.

**Input:**

1) id: The address of the record to be read.

2) Type: The type of record to read.

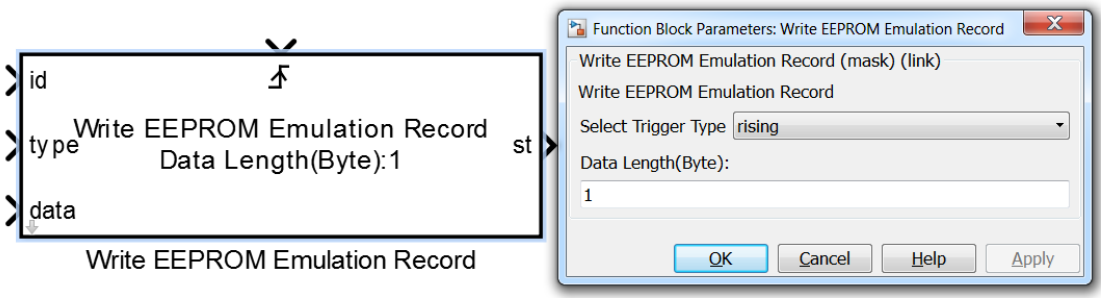
**Output:**

1) data: The data read.

2) st: output status, refer to the appendix table.

## 3.14.14 Write EEPROM Emulation Record

This module is used to write data to the emulation EEPROM.




**Parameter:**

1) Select Trigger Type: Select the trigger type.

2) Data Length: The length of the data to write to the record.

**Input:**

1) id: The address to write to the record.

2) Type: The type of record to write.

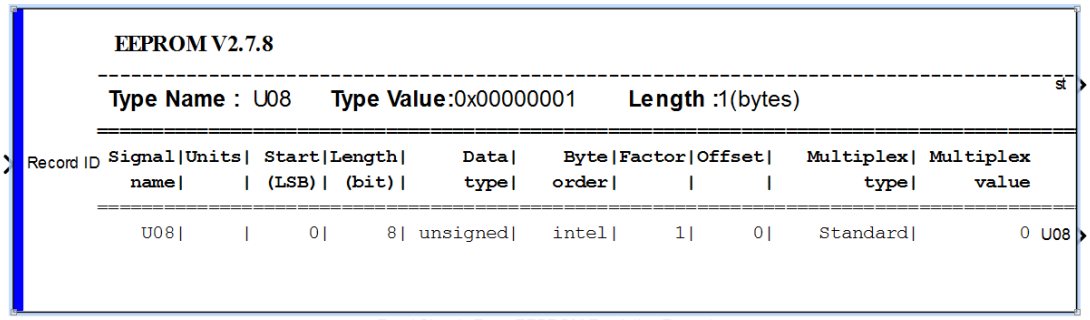
3) data: The data to write to the record.

**Output:**

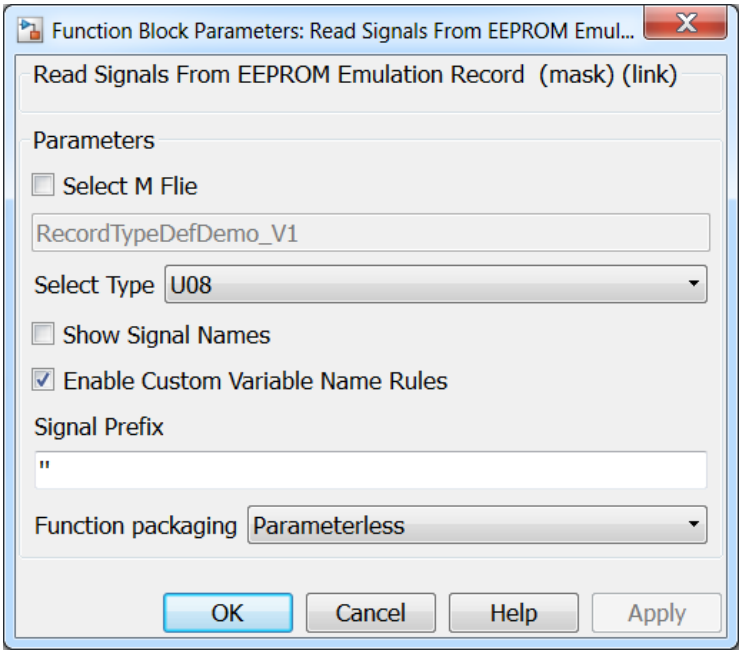
1) st: output status, refer to the appendix table.

## 3.14.15 Read Signals from EEPROM Emulation Record

This module is used to parse signals from emulation EEPROM data.




Read Signals From EEPROM Emulation Record




**Parameter:**

1) Select M File: m File selection. After checking, you can enter the name of the m file below.

2) Select Type: Select the record data type.

3) Show Signal Names: Show signal name. The name of the signal will be displayed on the output signal line if checking.

4) Signal prefix: Signal prefix. Used to add variable name prefixes on signal lines.

5) Function packaging: Module C code function settings, divided into Two types of Parameterless and Parameterized, of which Parameterless is a parameterless type function, code execution efficiency is high, while Parameterized is a function with parameter type, supporting the case that the C code corresponding to the module input and output signals is a local variable, avoid compilation errors.

**Input:**

1) Record ID: The ID of the record to be read and parsed.

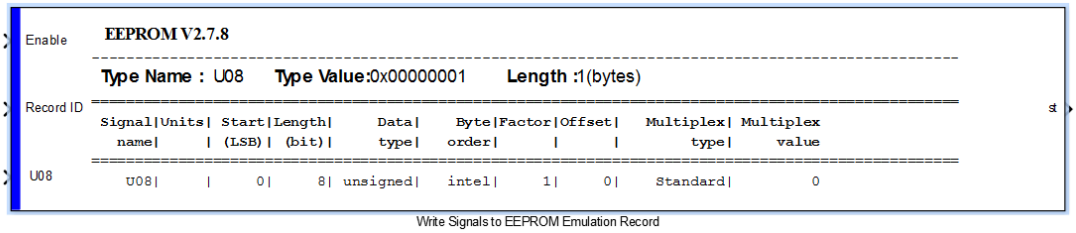
**Output:**

1) st: output status, refer to the appendix table.

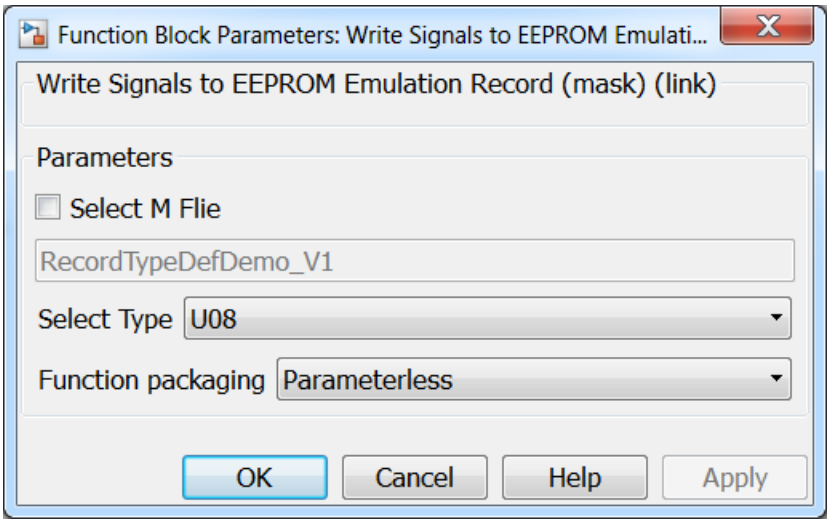
Note: The signal value parsed from the record is the physical value in real meaning.

## 3.14.16 Write Signals to EEPROM Emulation Record

This module is used to write signals to EEPROM.




Write Signals to EEPROM Emulation Record




**Parameter:**

1) Select M File: m File selection. After checking, you can enter the name of the m file below.

2) Select Type: Select the record data type.

**Input:**

1) Enable: Write record enable control, it is recommended to use edge trigger signal control.

2) Record ID: The address to write to the record.

3) To write a recorded signal, the signal value is the physical value.

4) Function packaging: Module C code function settings, divided into Two types of Parameterless and Parameterized, of which Parameterless is a parameterless type function, code execution efficiency is high, and Parameterized is a function with parameter type, which supports the case that the C code corresponding to the

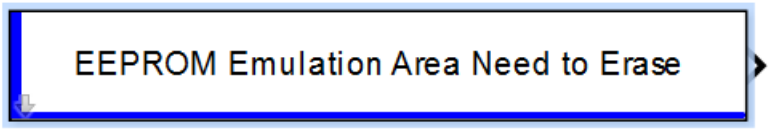
module input and output signals is a local variable to avoid compilation errors.

**Output:**

1) st: output status, refer to the appendix table.

### 3.14.17 EEPROM Emulation Area Need to Erase

This module is used to determine if the simulated EEPROM area needs to be erased.



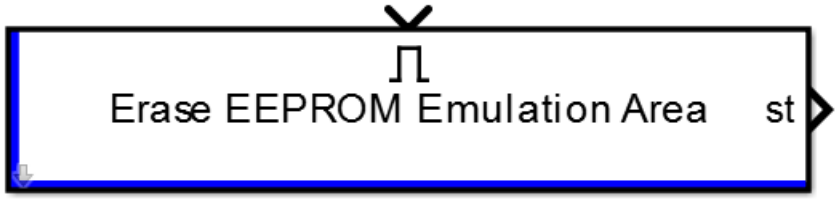

EEPROM Emulation Area Need to Erase

**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.

### 3.14.18 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.




Erase EEPROM Emulation Area

**Input:**

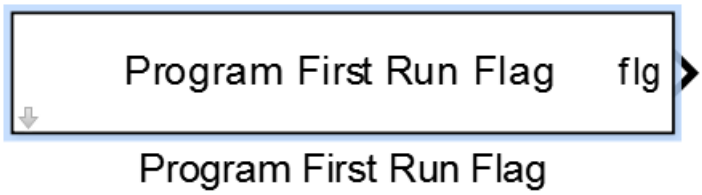
1) Trigger: input 1 to erase the emulation area

**Output:**

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

### 3.14.19 Program First Run Flag

This module is used to determine whether it is the first time the program is running after an update.

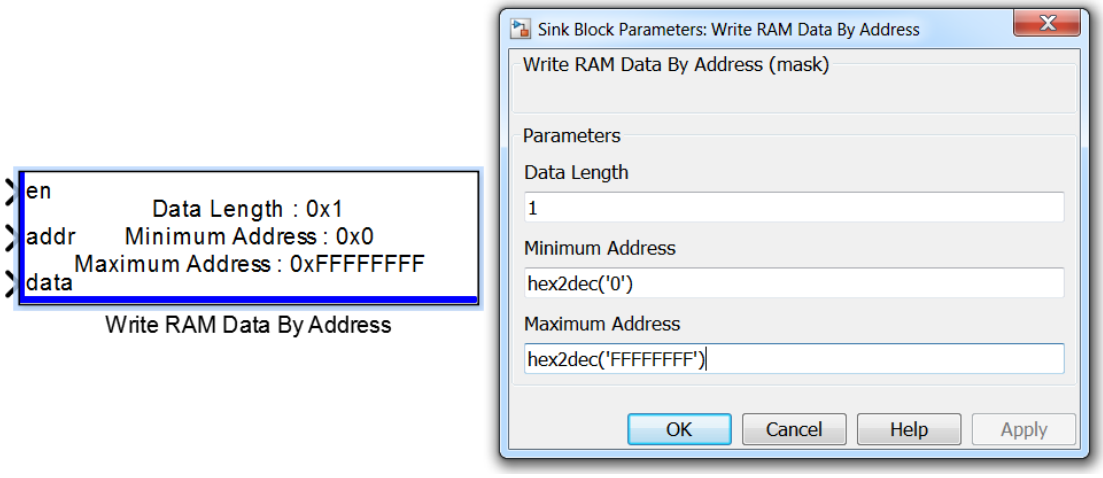



**Output:**

1) flg: Output flag value. If you output 1, it means that the program is running for the first time. After the successfully trigger Store All NVM Data module, the flag is clear to 0.

### 3.14.20 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

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

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

**Input:**

1) en: enable the writing operation

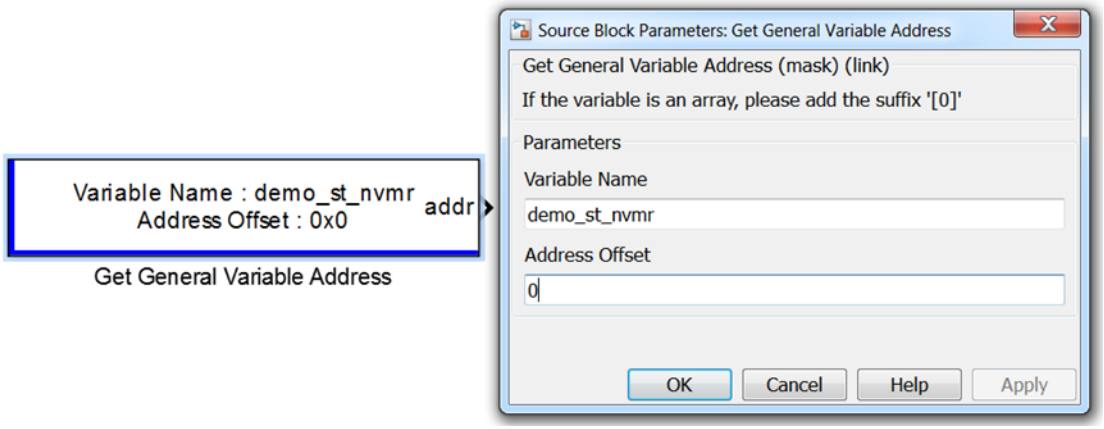
2) addr: start address

**Output:**

1) data

## 3.14.21 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

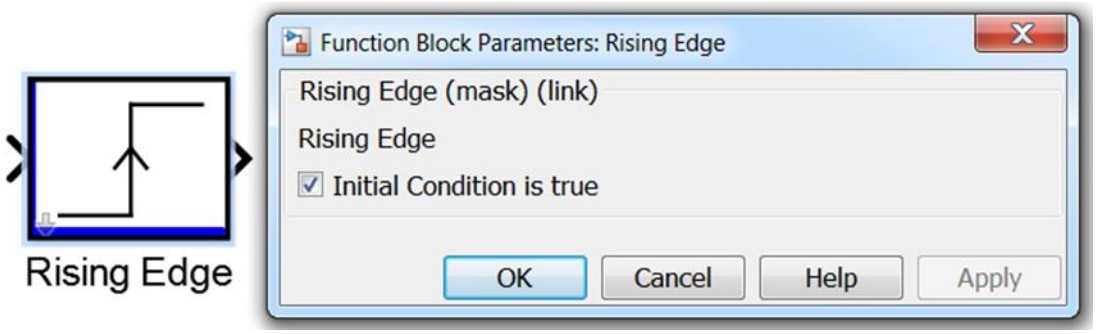
**Output:**

1) addr: the address of variable

## 3.15 Application Base Blocks

### 3.15.1 Rising Edge

This block is used for detecting rising edges.

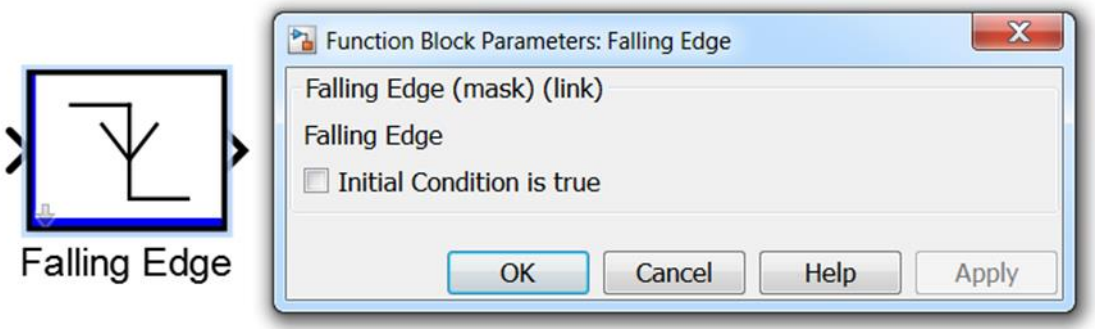



**Parameter:**

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

### 3.15.2 Falling Edge

This block is used for detecting falling edges.



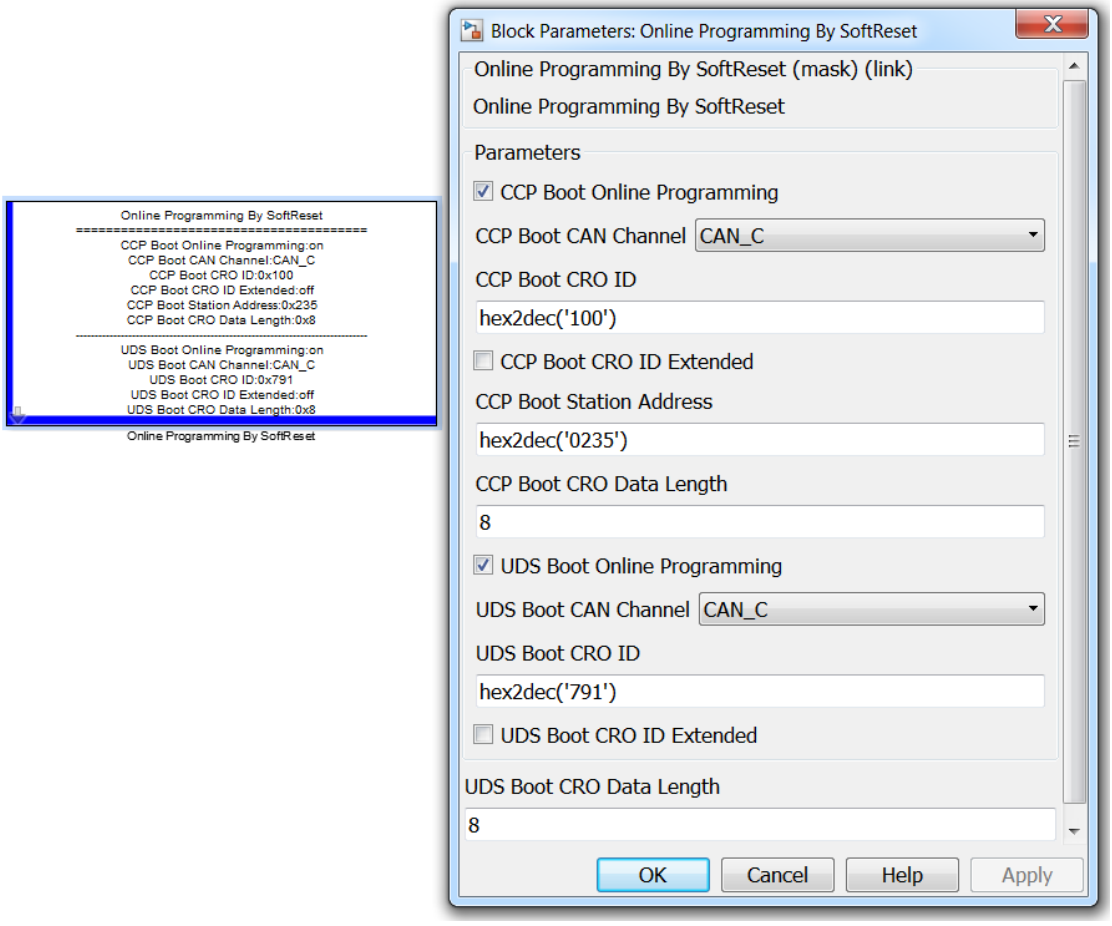

**Parameter:**

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

### 3.15.3 Online Programming by SoftReset

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

software.



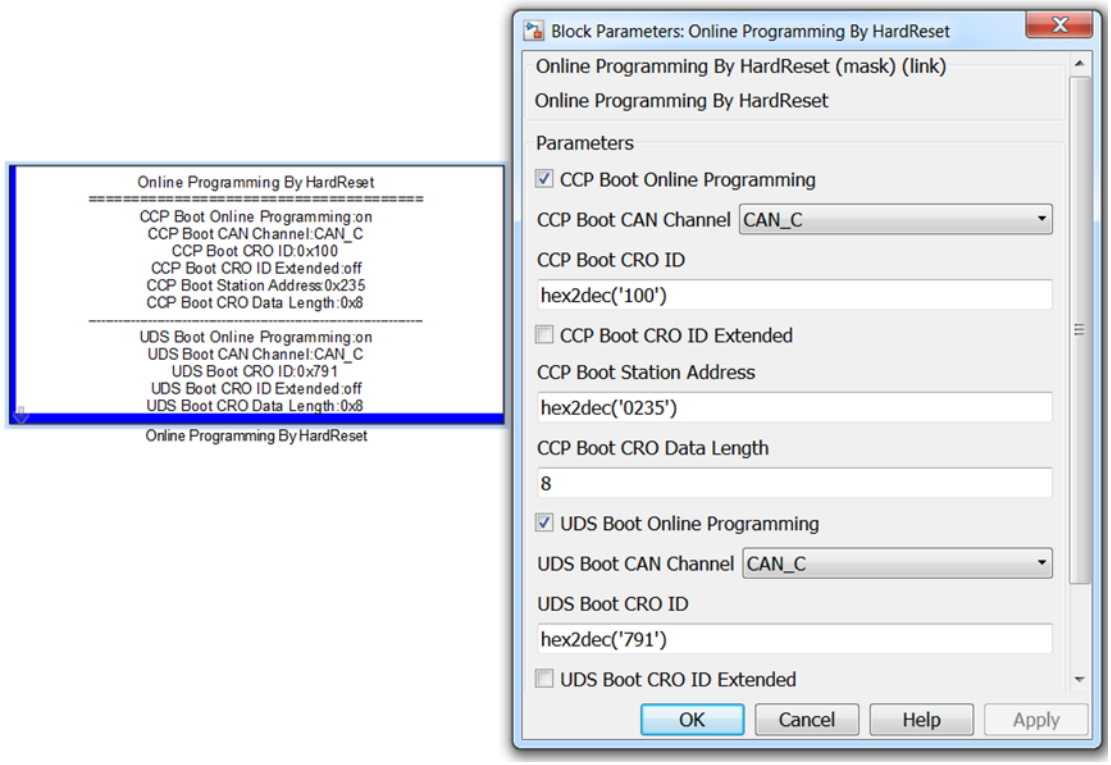

**Parameter:**

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

### 3.15.4 Online Programing By HardReset

This block can reset the controller hardware for online programming.




**Parameter:**

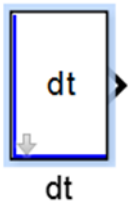
1) CCP Boot Online Programming: Check for enabling online programming
2) CCP Boot CAN Channel: Select CCP channel for online programming
3) CCP Boot CRO ID: Set CRO ID for online programming
4) CCP Boot CRO ID Extended: Select extended CRO frame for online programing
5) CCP Boot Station Address: Select 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

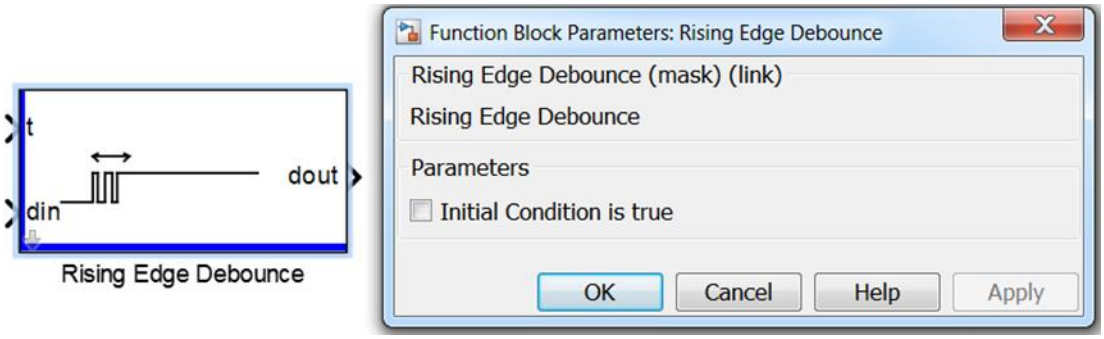
### 3.15.5 dt – time step length

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




### 3.15.6 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)

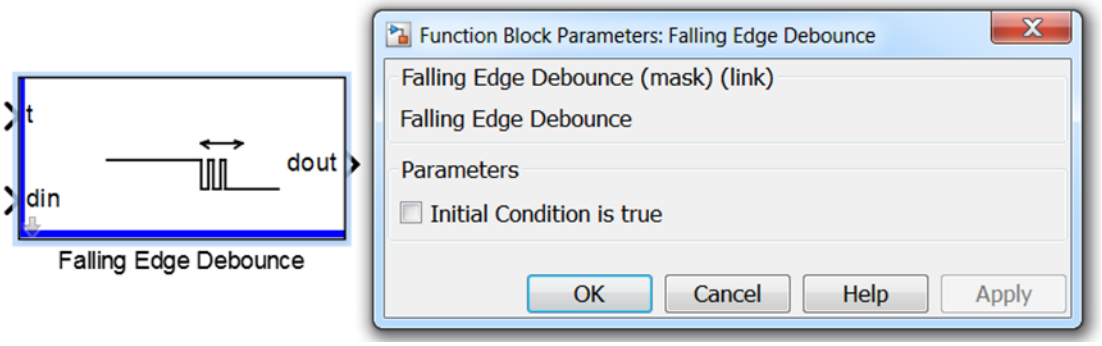
2) din: original input

**Output:**

1) dout: signal after debounce.

### 3.15.7 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:**

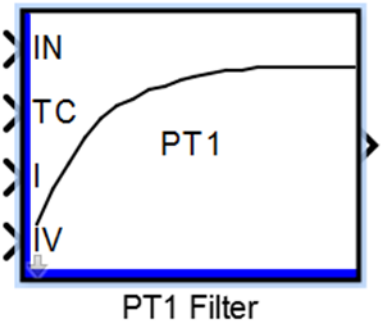
1) t: falling edge duration. (unit: second)
2) din: original input

**Output:**

1) dout: signal after debouncing.

### 3.15.8 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

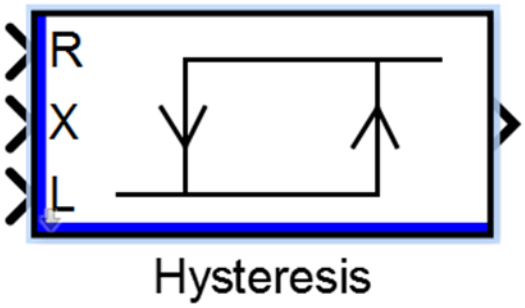
14) IV: Initialization value

**Output:**

1) Filtered signal

### 3.15.9 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.

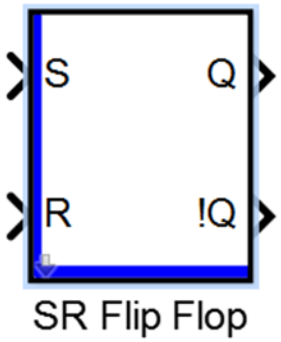
### 3.15.10 SR Flip Flop

!Q is the inverted Q.

When R=1 → Q=0

When R=0 and S=1 → Q=1

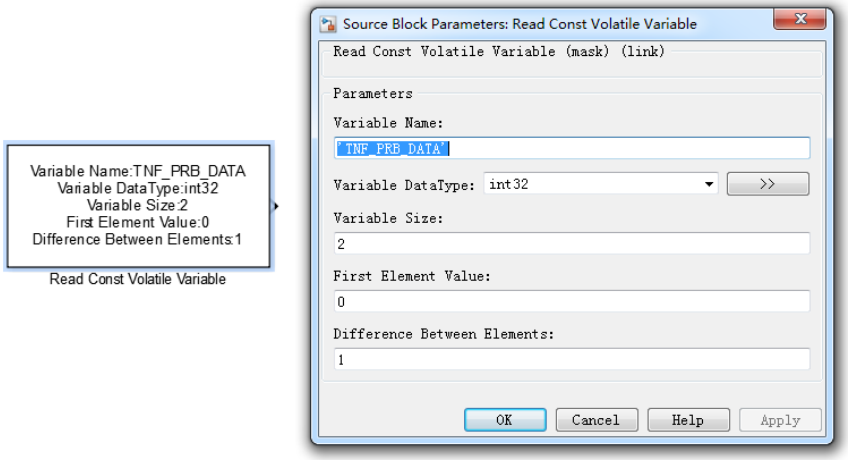
When R=0 and S=0 → Q=Previous value




## 3.16 Volatile Variable

### 3.16.1 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

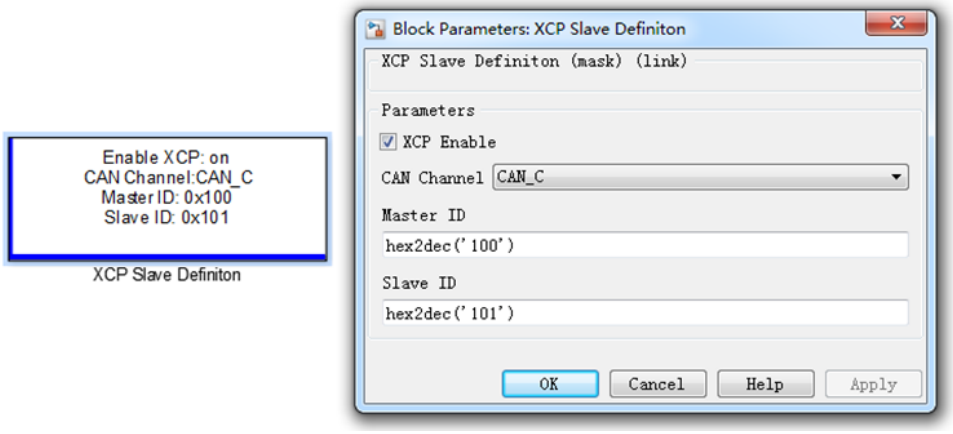
**Output:**

1) Read value from the specific variable

## 3.17 XCP Module

### 3.17.1 XCP Slave Definition

This block is used for initializing XCP module.



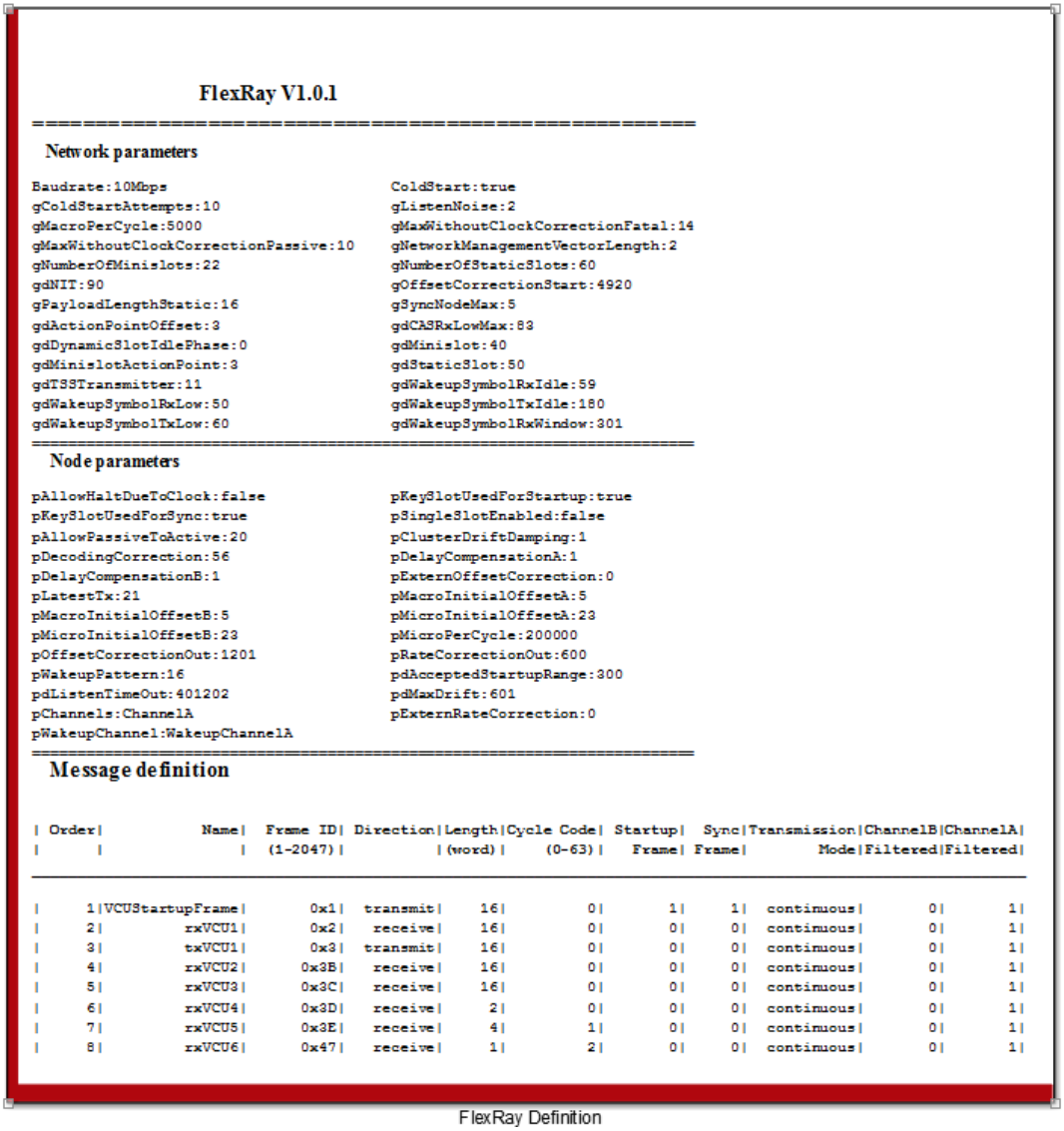

**Parameter:**

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

## 3.18 FlexRay Module

### 3.18.1 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
```

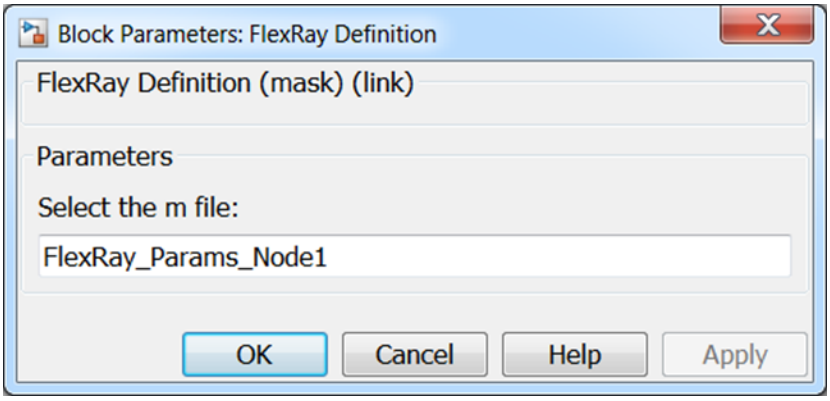
Node parameters

```
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

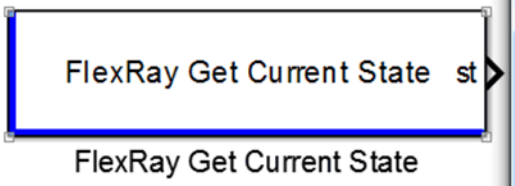


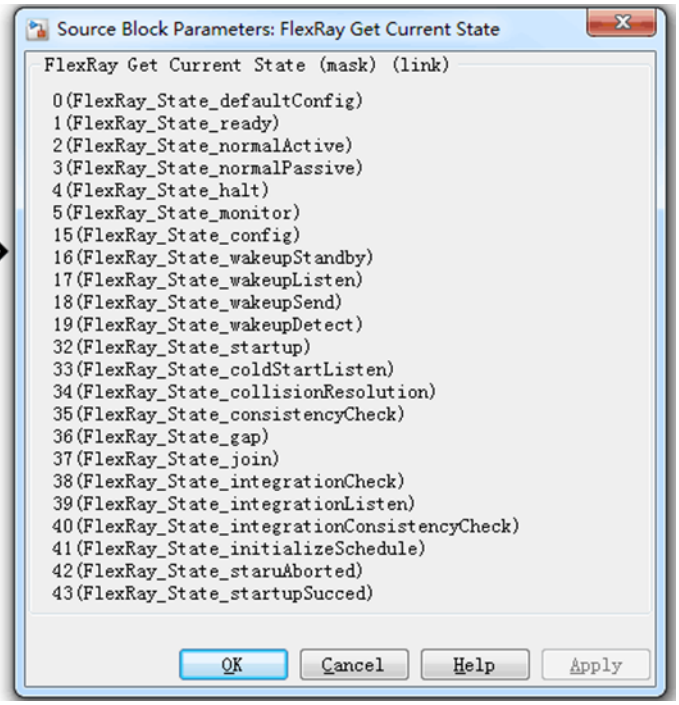

**Parameter:**

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.

### 3.18.2 FlexRay Get Current State

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



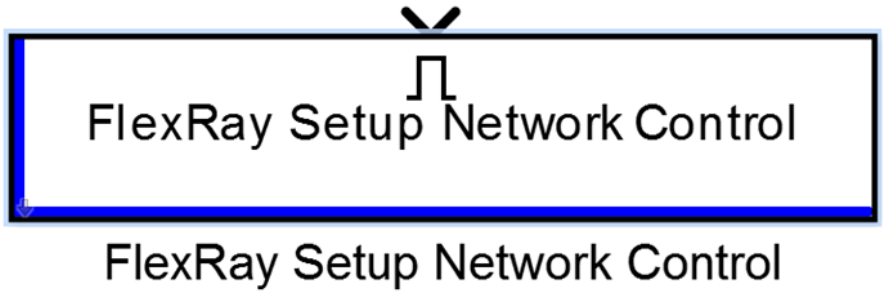




**Output:**

1) St: status of the network

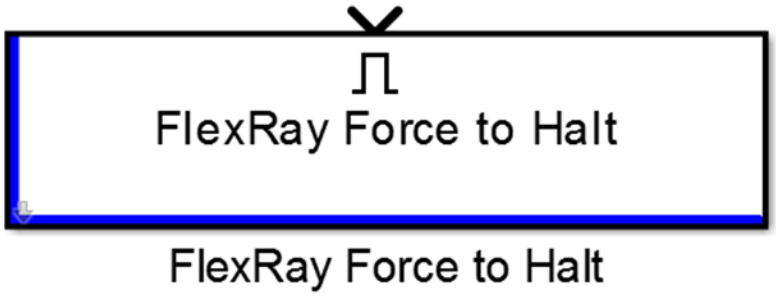
### 3.18.3 FlexRay Setup Network Control

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



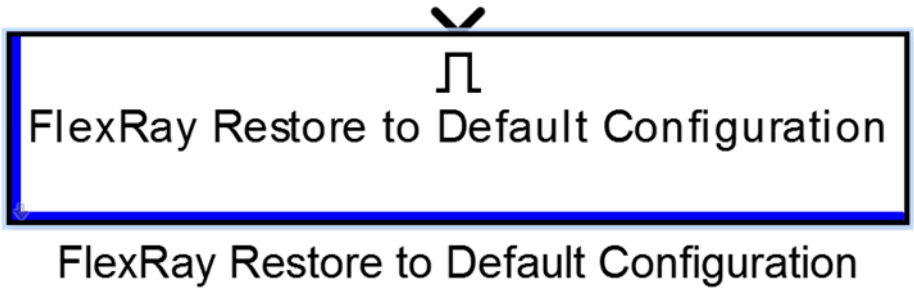

### 3.18.4 FlexRay Force to Halt

This block is used for force halt the FlexRay communication.



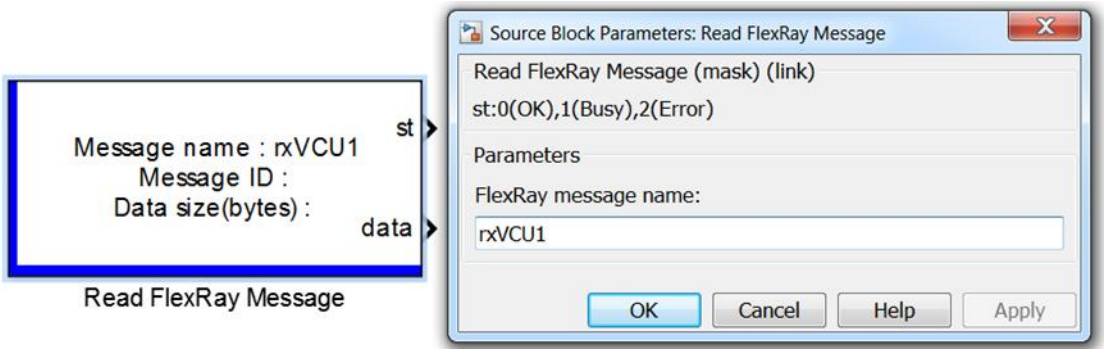

### 3.18.5 FlexRay Restore to Default Configuration

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



### 3.18.6 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.

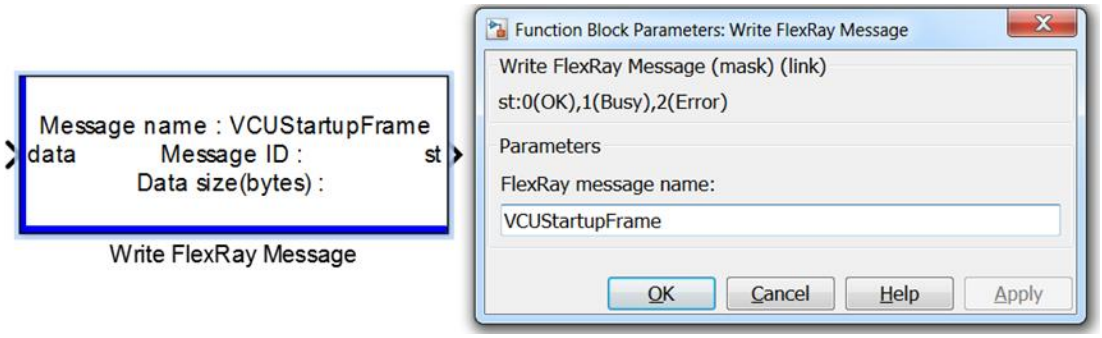
**Output:**

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

5) data: data from the message

### 3.18.7 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

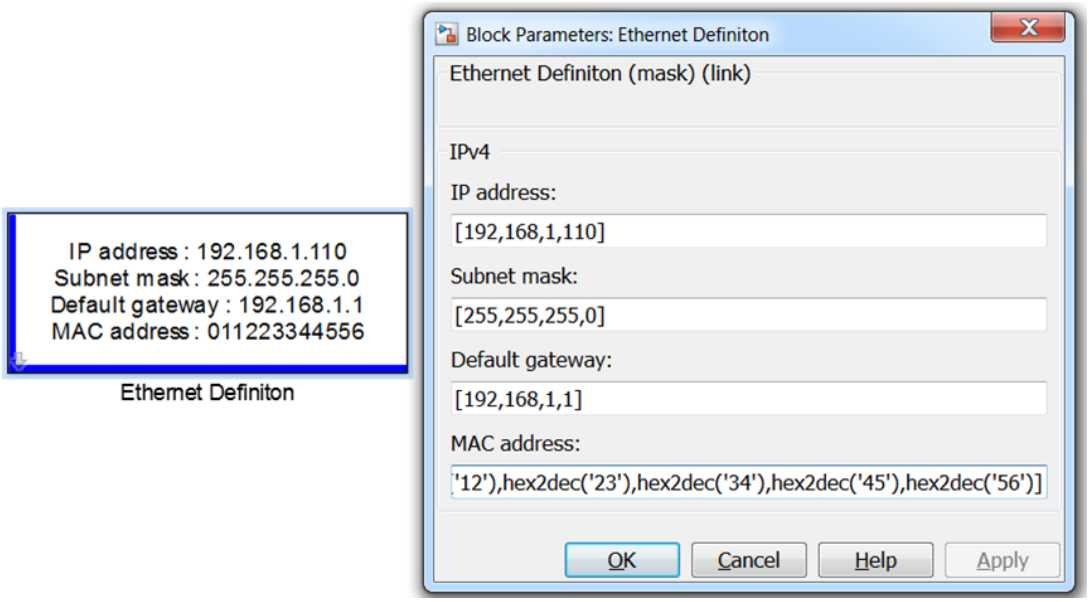
**Output:**

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

## 3.19 Ethernet

### 3.19.1 Ethernet Definition

This block is used for configuring the Ethernet.

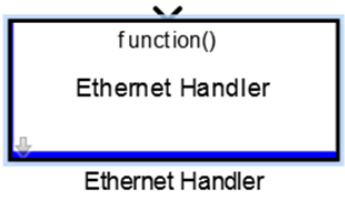



**Parameter:**

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

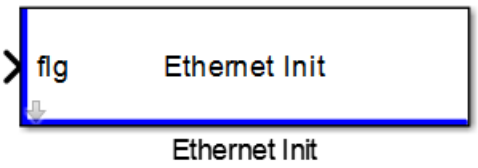
### 3.19.2 Ethernet Handler

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




### 3.19.3 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:**

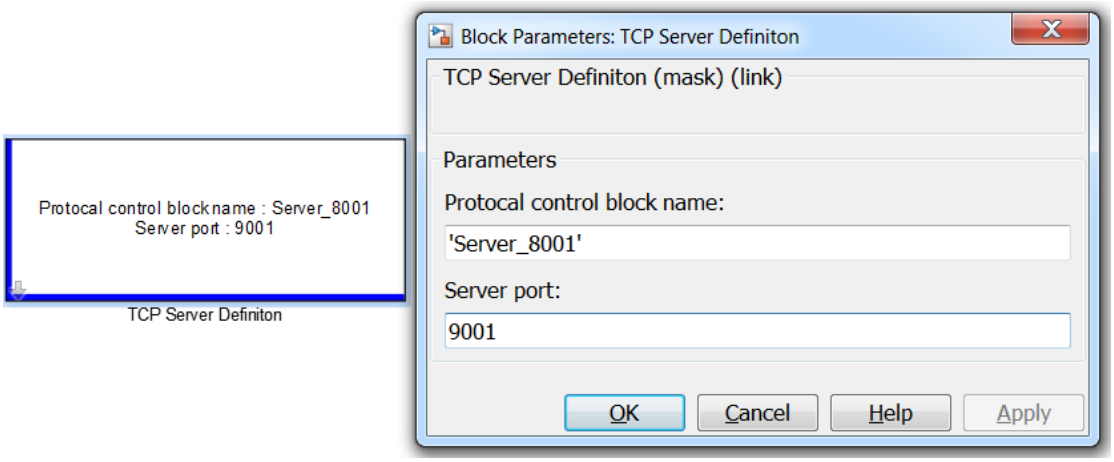
1) Flg: input 1 to initialize Ethernet.

## 3.20 TCP Protocol Blocks

TCP is the Ethernet transport layer protocol for establishing connection.

### 3.20.1 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.

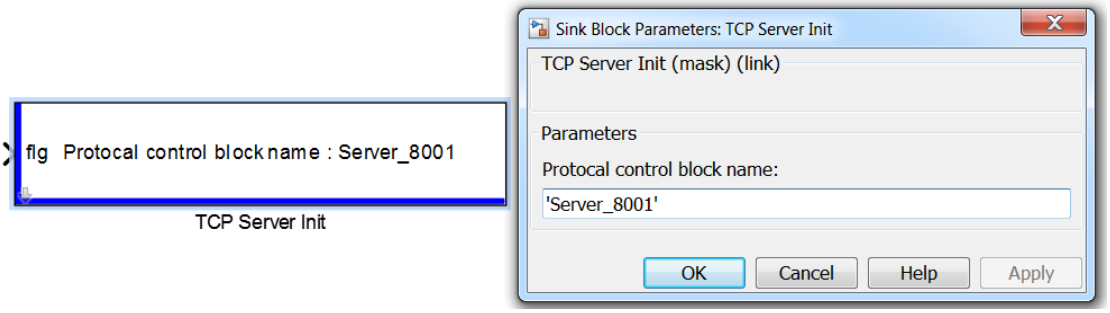



**Parameter:**

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

## 3.20.2 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:**

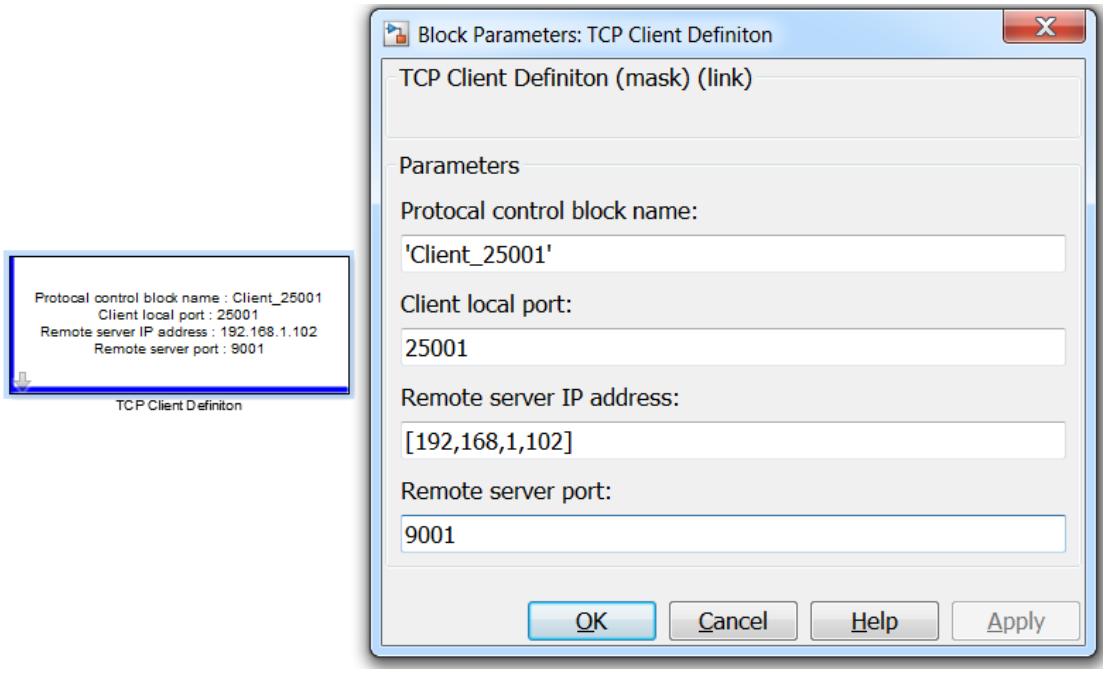
1) Protocol control block name: Name of the control block.

**Input:**

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

### 3.20.3 TCP Client Definition

This block is used to define a TCP client.

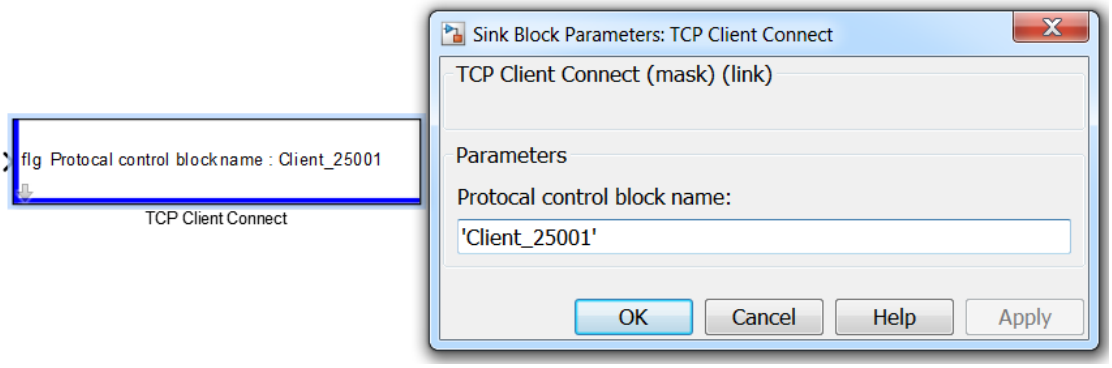



**Parameter:**

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

### 3.20.4 TCP Client Connect

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




**Parameter:**

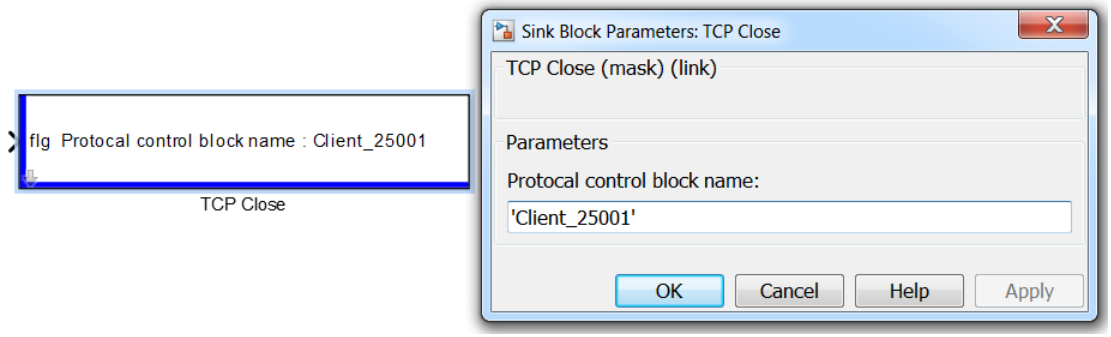
1) Protocol control block name: name of the control block

**Input:**

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

## 3.20.5 TCP Close

This block closes the current connection.




**Parameter:**

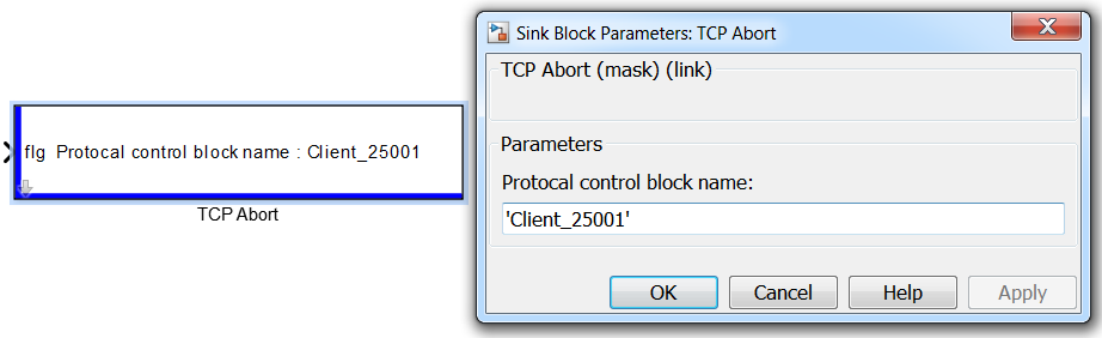
1) Protocol control block name: name of the control block

**Input:**

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

## 3.20.6 TCP Abort

This block aborts the current connection.




**Parameter:**

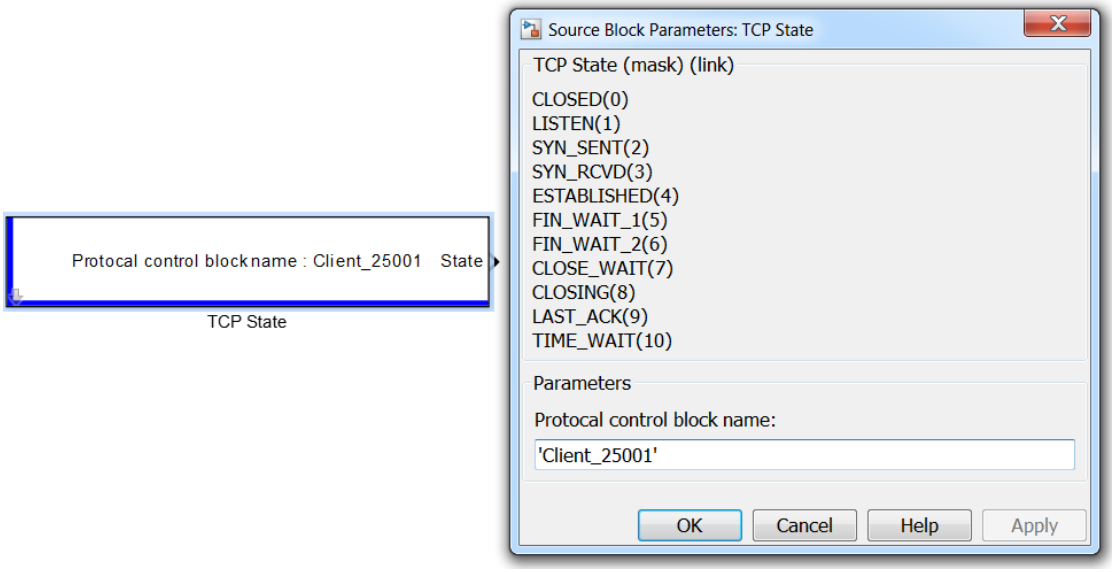
1) Protocol control block name: name of the control block

**Input:**

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

## 3.20.7 TCP State

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




**Parameter:**

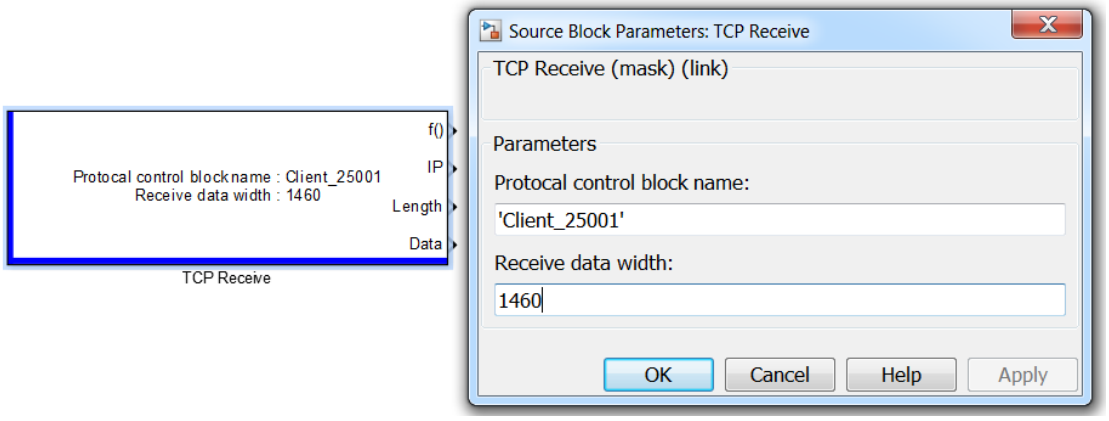
1) Protocol control block name: name of the control block

**Output:**

1) State: current state of the connection.

## 3.20.8 TCP Receive

This block is used for receiving TCP data.




**Parameter:**

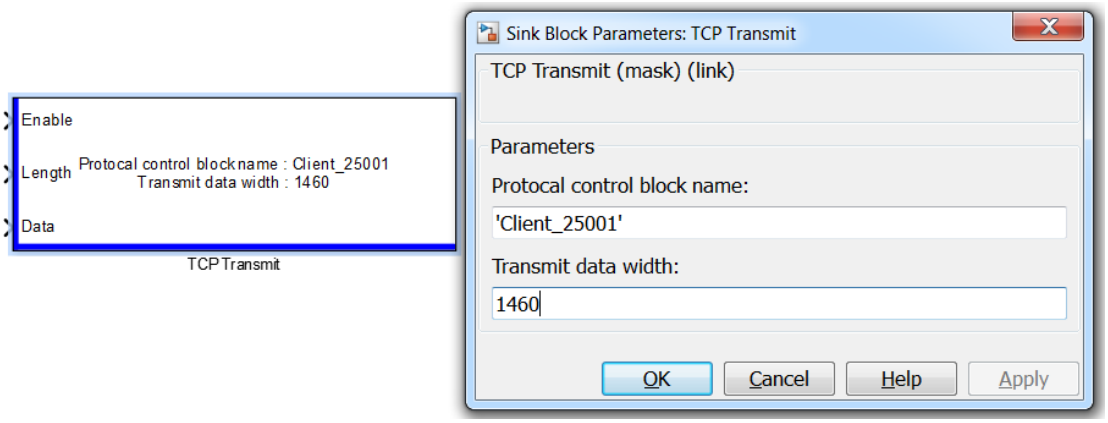
1) Protocol control block name: name of the control block
2) Receive data width: the width of the Data port

**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

## 3.20.9 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

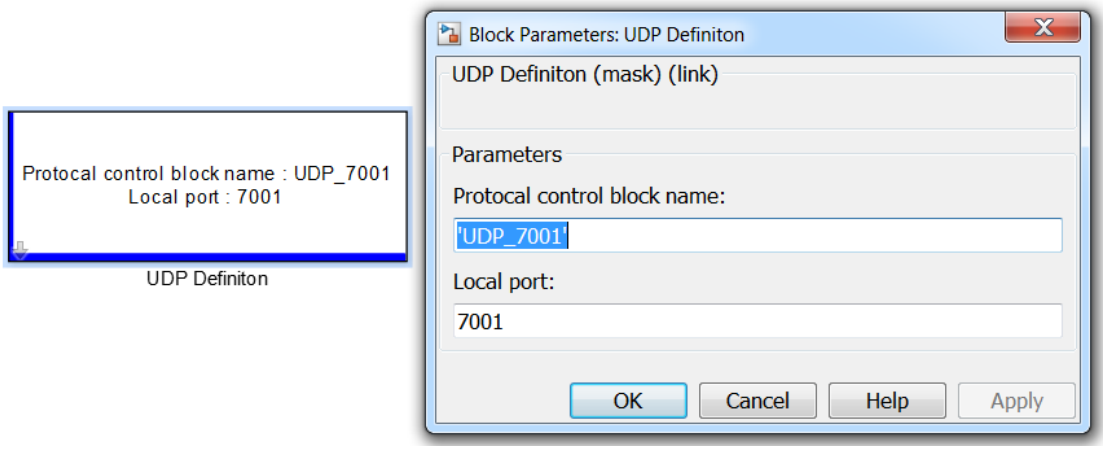
**Input:**

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

## 3.21 UDP Blocks

### 3.21.1 UDP Definition

This block is used for defining UDP ports.



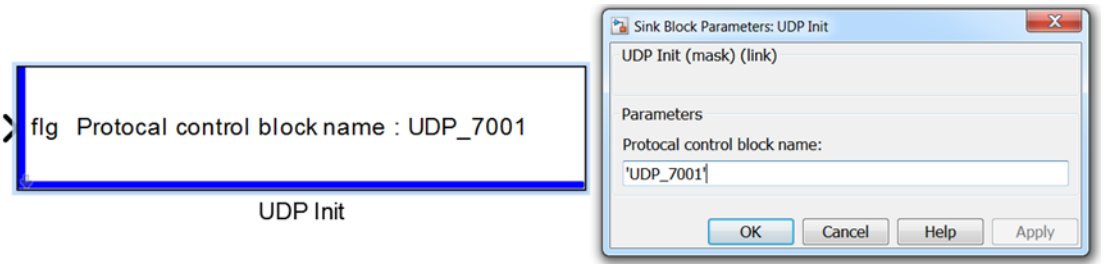
**Parameter:**

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

5) Local Port: local port configuration

## 3.21.2 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:**

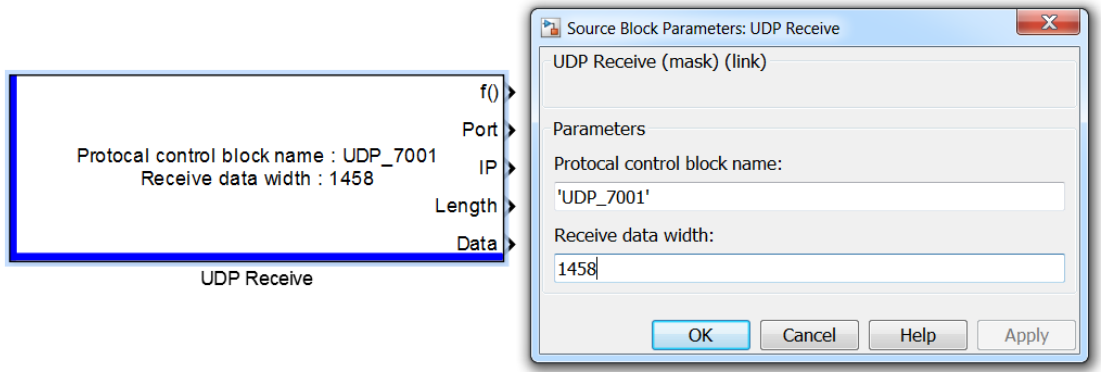
1) Protocol control block name: the name of the protocol control block

**Input:**

1) Flg: input 1 to enable the initialization.

## 3.21.3 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

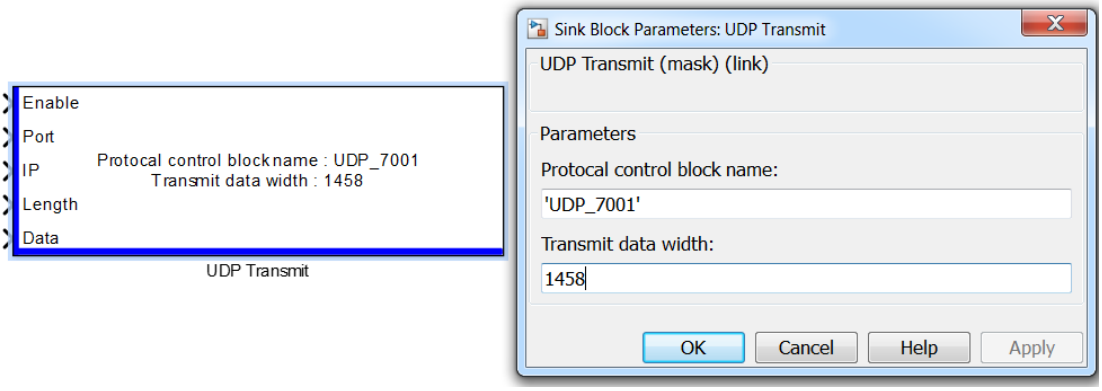
8) IP: IP address of the sender

9) Length: the actual length of the received data

10) Data: received data

### 3.21.4 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

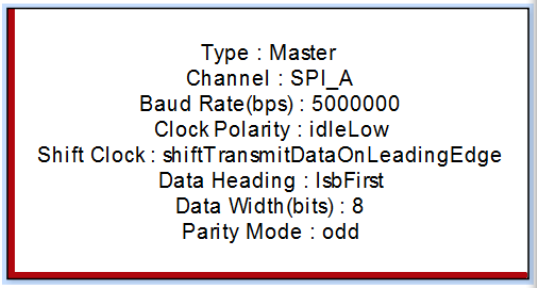
14) Length: the length of the data to be transmitted

15) Data: the data to be transmitted

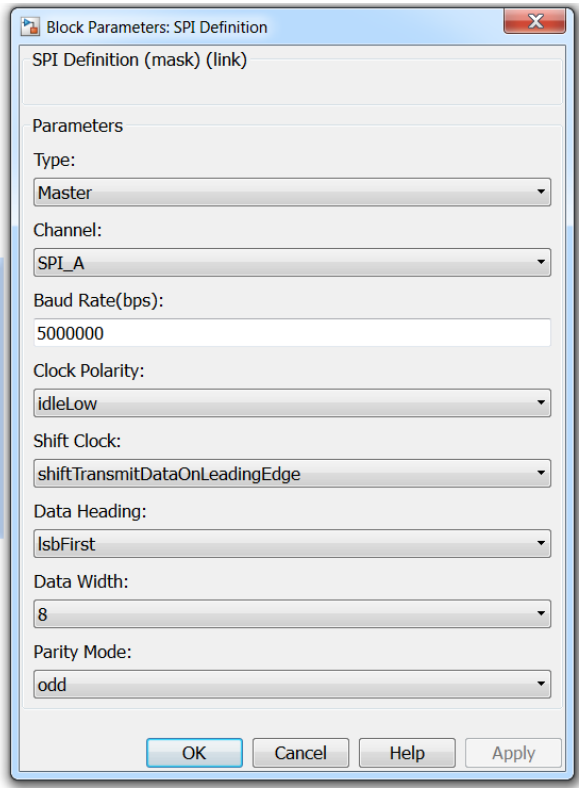
## 3.22 SPI Blocks

### 3.22.1 SPI Definition

Set parameters for SPI module, include master-slave mode, baud rate, etc.




SPI Definition

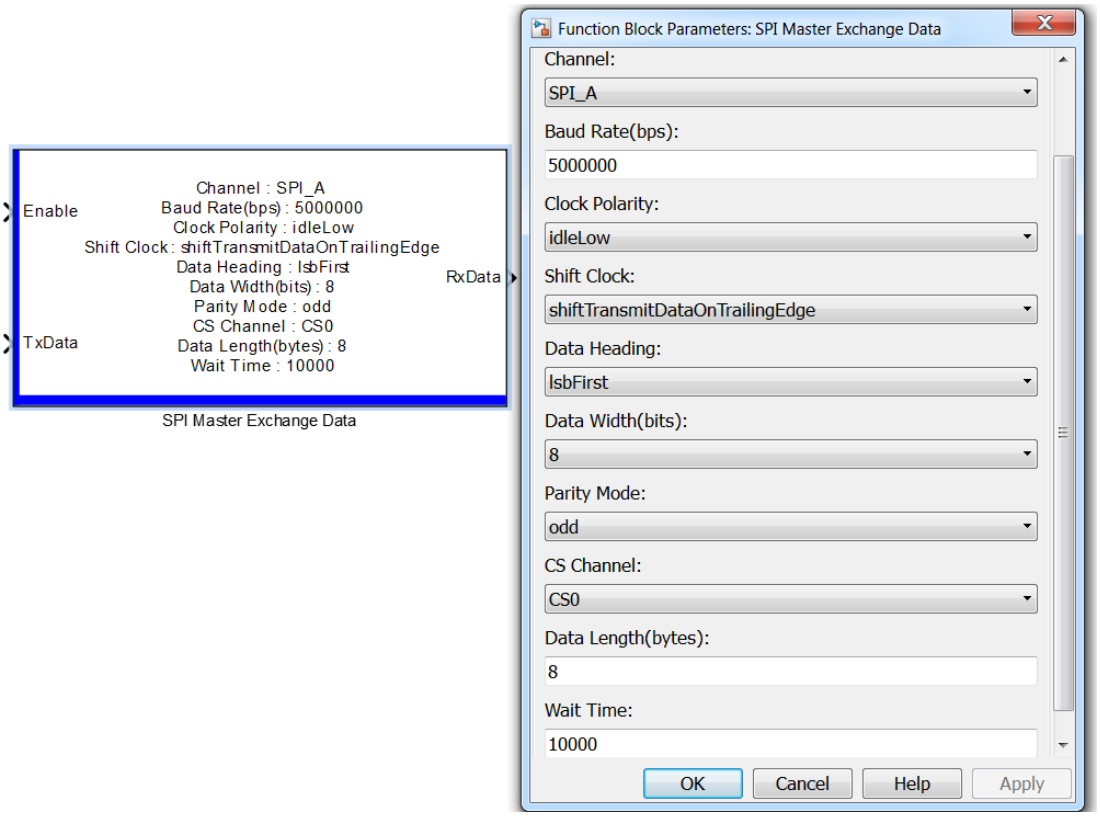



**Parameter:**

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

### 3.22.2 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:**

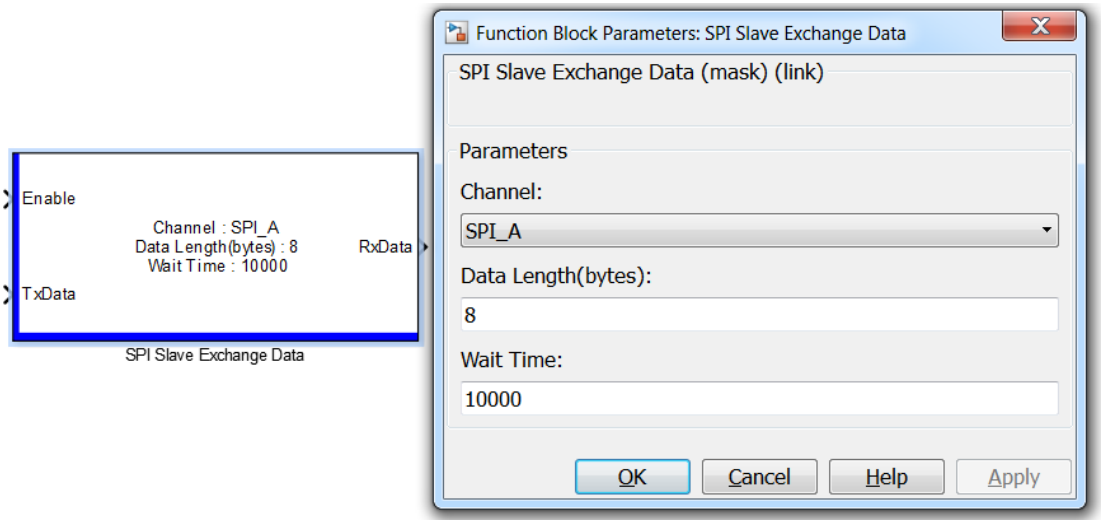
1) Enable: set to 1 to enable the transmission, set to 0 to disable the transmission
2) TxData: the data to be transmitted

**Output:**

1) RxData: the data to be received

### 3.22.3 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:**

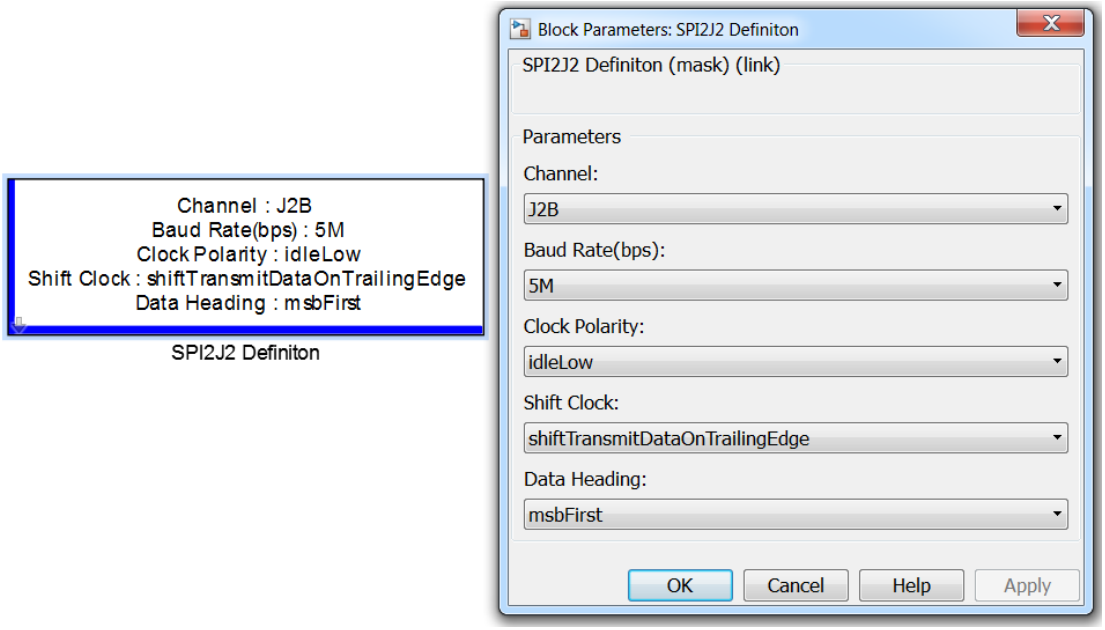
1) RxData: the data to be received

## 3.23 SPI2J2

The modules in the SPI2J2 subsystem are transport layer protocol modules that communicate with the J2 chip, and the application layer protocol that communicates with J2 can be implemented through these modules.

### 3.23.1 SPI2J2 Definition

Through this module, the parameters of the SPI communicating with J2 are set, where the parameters include baud rate, clock polarity, etc.




**Parameter:**

1) Channel: Channel selection.

2) Baud Rate (bps): Baud rate.

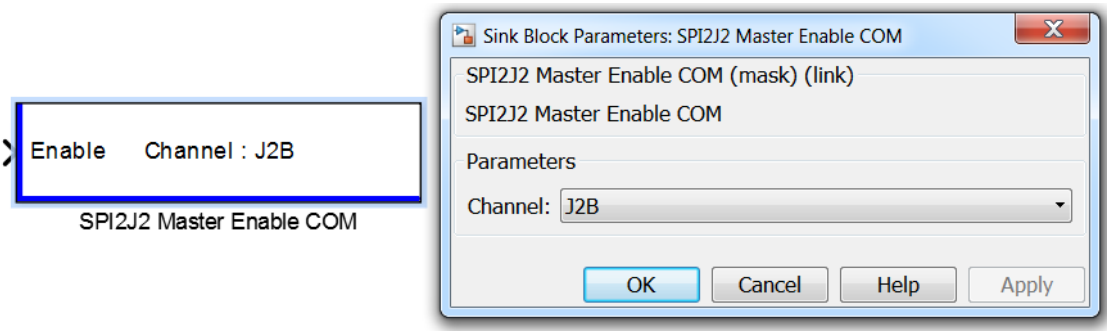
3) Clock Polarity: Clock polarity.

4) Shift Clock: Shift time.

5) Data Heading: Bit sequence for data transfer.

### 3.23.2 SPI2J2 Master Enable COM

This module enables you to turn on or off communication with J2.




**Parameter:**

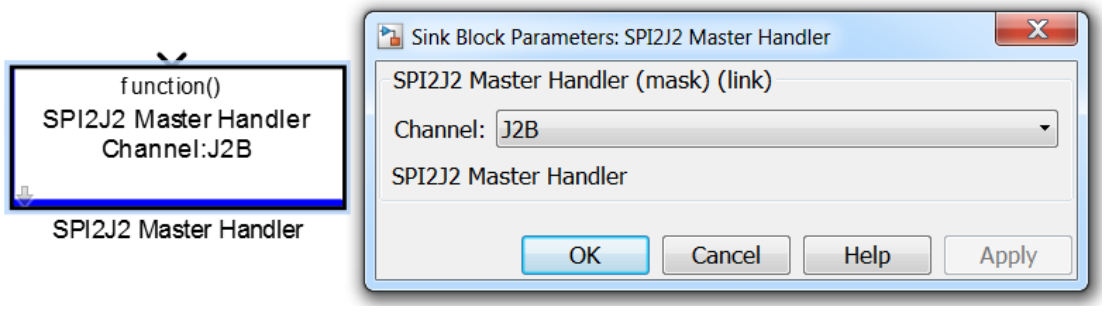
1) Channel: Channel selection.

**Input:**

1) Enable: 1 is to turn on communication, 0 is to turn off communication.

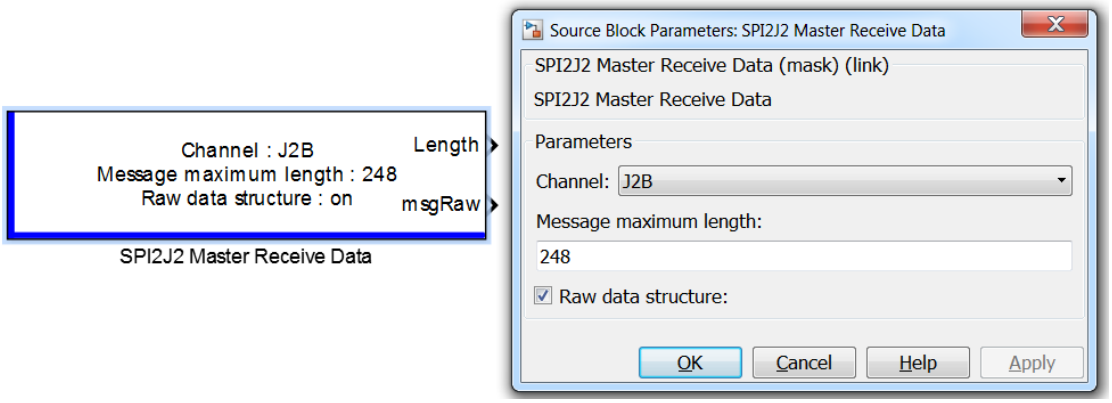
### 3.23.3 SPI2J2 Master Handler

The transport layer protocols of SPI and J2 only work if this module is called periodically.



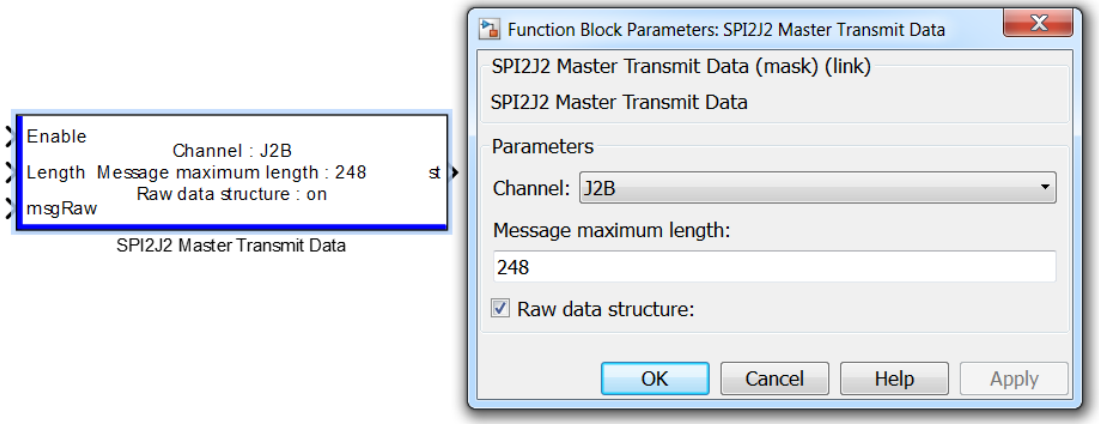
### 3.23.4 SPI2J2 Master Receive Data

This module allows you to read data sent from J2.




### 3.23.5 SPI2J2 Master Transmit Data

This module allows data to be sent to J2.




**Parameter:**

1) Channel: Channel selection.

2) Message maximum length: the maximum data length, which is the width of input port 2, when the parameter Raw data structure is enabled, it is the width of the input msgRaw, and when the parameter Raw data structure is not enabled, it is the total width of all members of the input msgBus, that is, the width of the member data in the msgBus plus 16.

3) Raw data structure: data structure of port 2, which is the data of the original array type when enabled and the data of the struct type when not enabled.

**Input:**

1) Length: The total length of the original data to be sent.

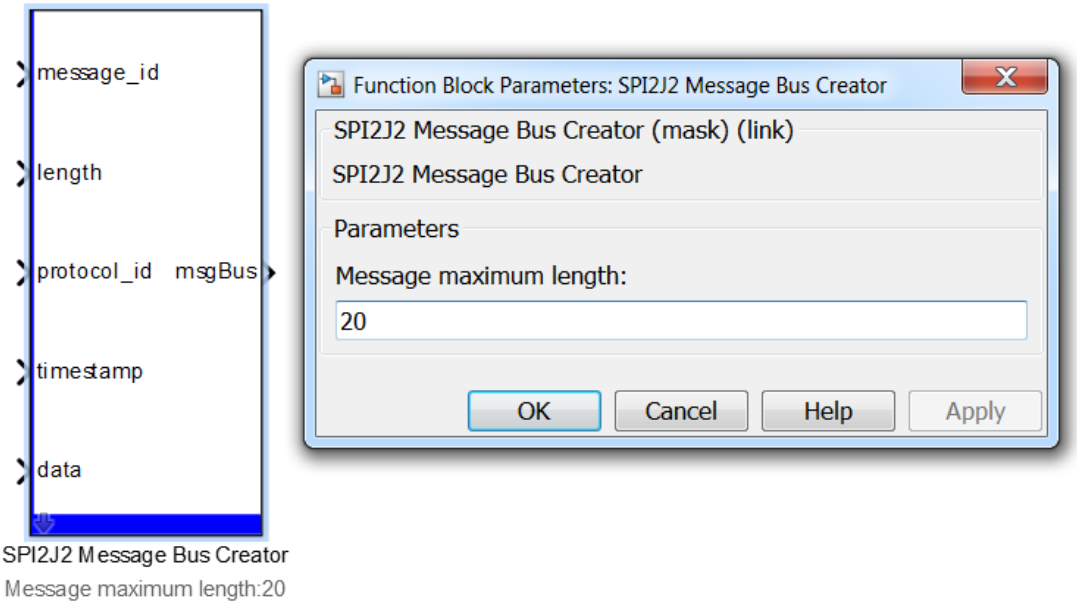
2) msgRaw or msgBus: The data to be sent, when raw data structure enables, raw array type is msgRaw, and when not enabled, it's structure type msgBus.

**Output:**

1) st: Send status, 0 is send successful, other values are send failed.

### 3.23.6 SPI2J2 Message Bus Creator

This module allows you to generate the data structures required by SPI2J2 Master Transmit Data




**Parameter:**

1) Message maximum length: The maximum data length, which is the total width of all members of the output port msgBus, that is, the data width of the members in the msgBus plus 16.

**Input:**

1) message_id: Application layer message ID.

2) length: The length of the application layer data.

3) protocol_id: Application layer protocol ID.

4) timestamp: Application layer protocol timestamp.

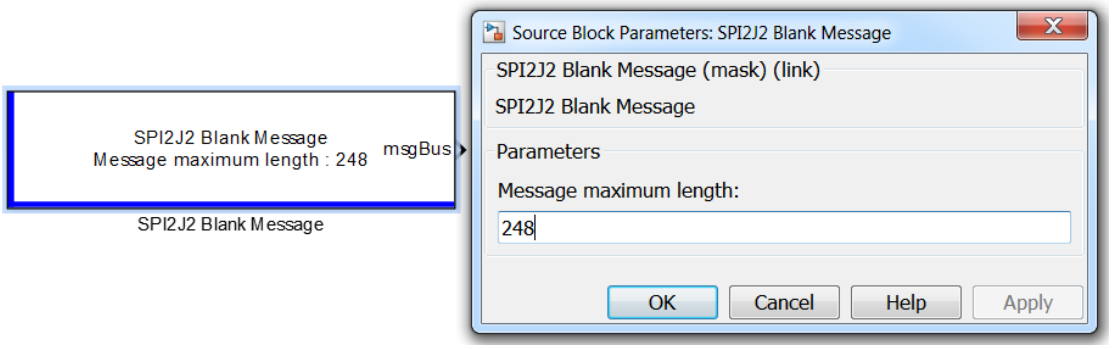
5) data: Application-layer data.

**Output:**

1) msgBus: A bus data structure consisting of input signals.

### 3.23.7 SPI2J2 Blank Message

This module can be used with Simulink's Bus Assignment library to update the members of the bus and then compose the data structures required by SPI2J2 Master Transmit Data.




**Parameter:**

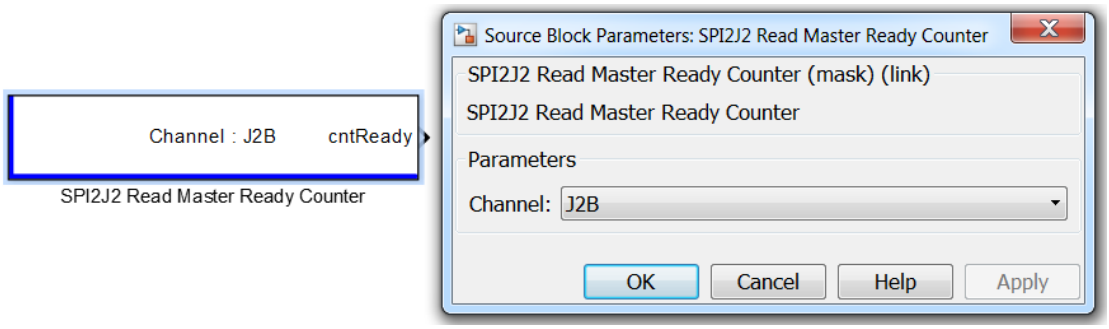
1) Message maximum length: The maximum data length, which is the total width of all members of the output port msgBus, that is, the data width of the members in the msgBus plus 16.

**Output:**

1) msgBus: A bus data structure that conforms to the J2 application layer.

### 3.23.8 SPI2J2 Read Master Ready Counter

This module can get the total number of Ready signals sent by J2.




**Parameter:**

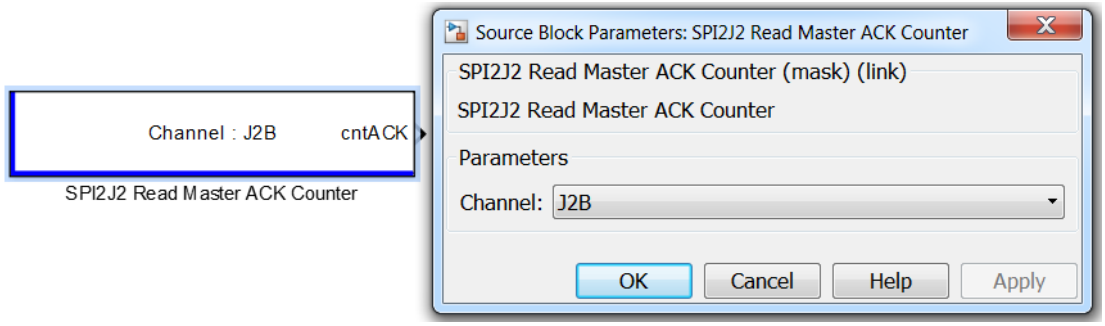
1) Channel: Channel selection.

**Output:**

1) cntReady: The total number of Ready signals.

## 3.23.9 SPI2J2 Read Master ACK Counter

This module can get the total number of ACK signals sent by J2.




**Parameter:**

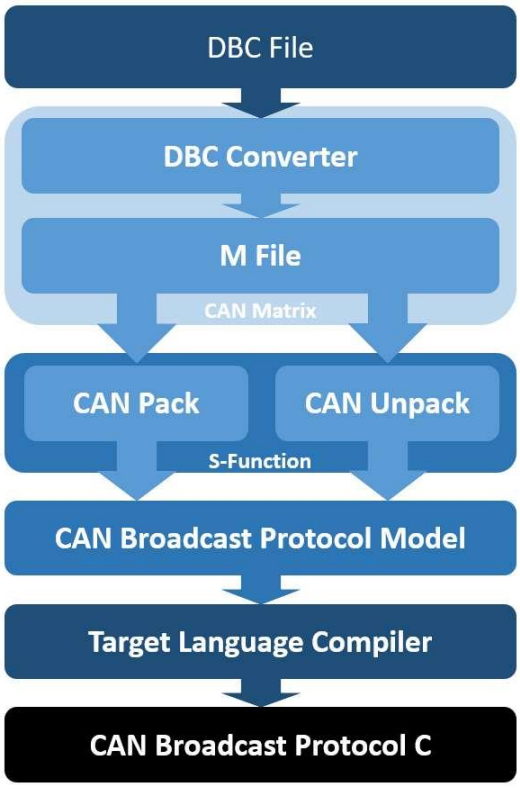
1) Channel: Channel selection.

**Output:**

1) cntReady: The total number of Ready signals.

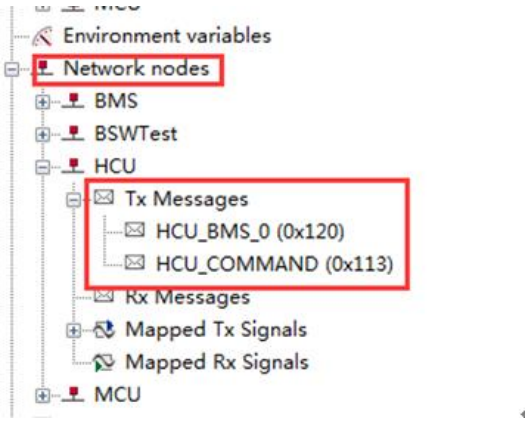
# Chapter 4 CAN Theory of Ecotron

The specific implementation of the application layer can be defined by the broadcast protocol matrix through the dbc file or the m file. The whole code generation process is as follows:



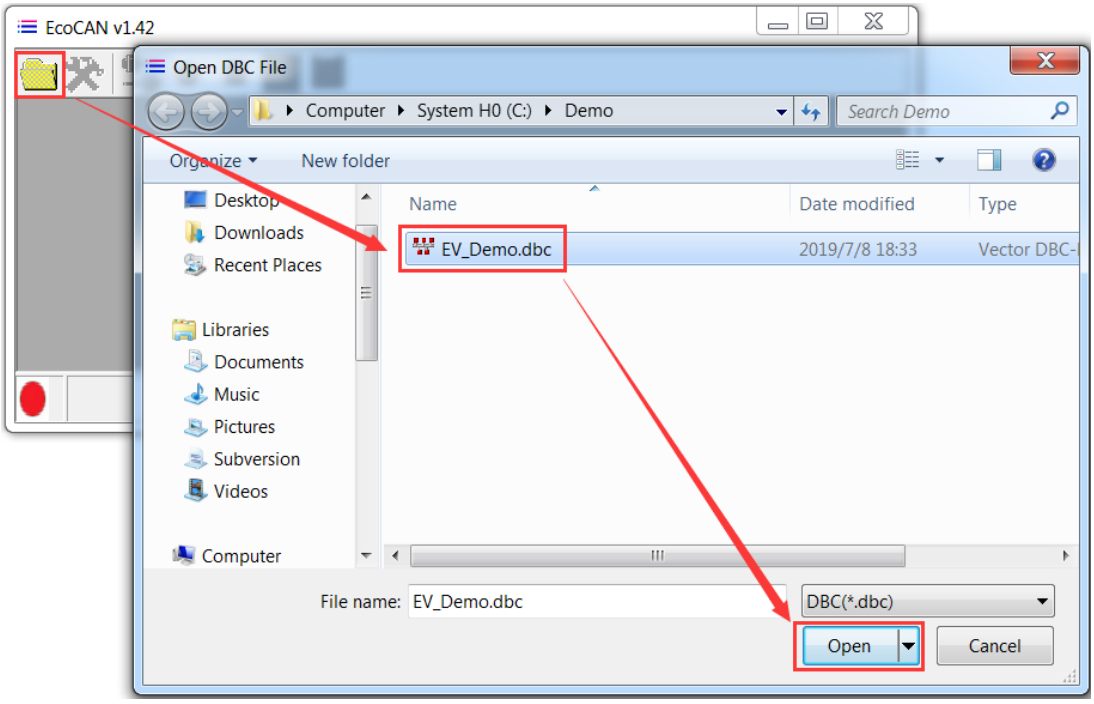

## 4.1 Convert DBC to m File

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*.

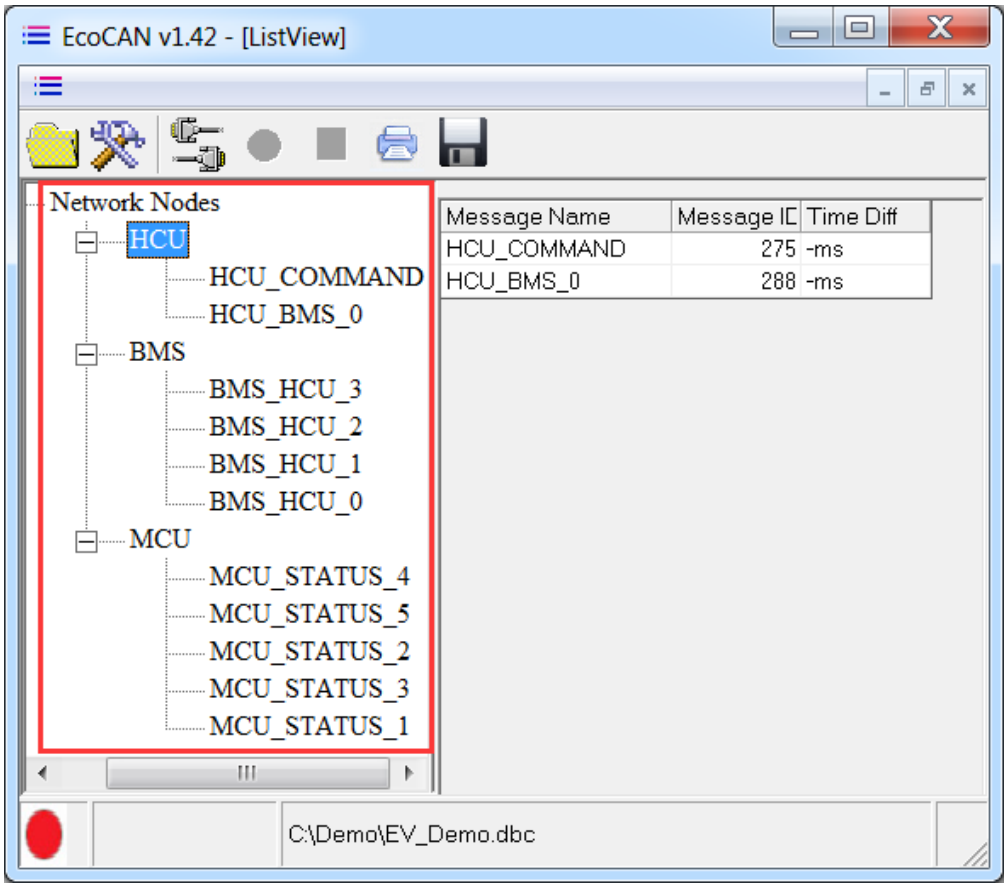



**Process:**

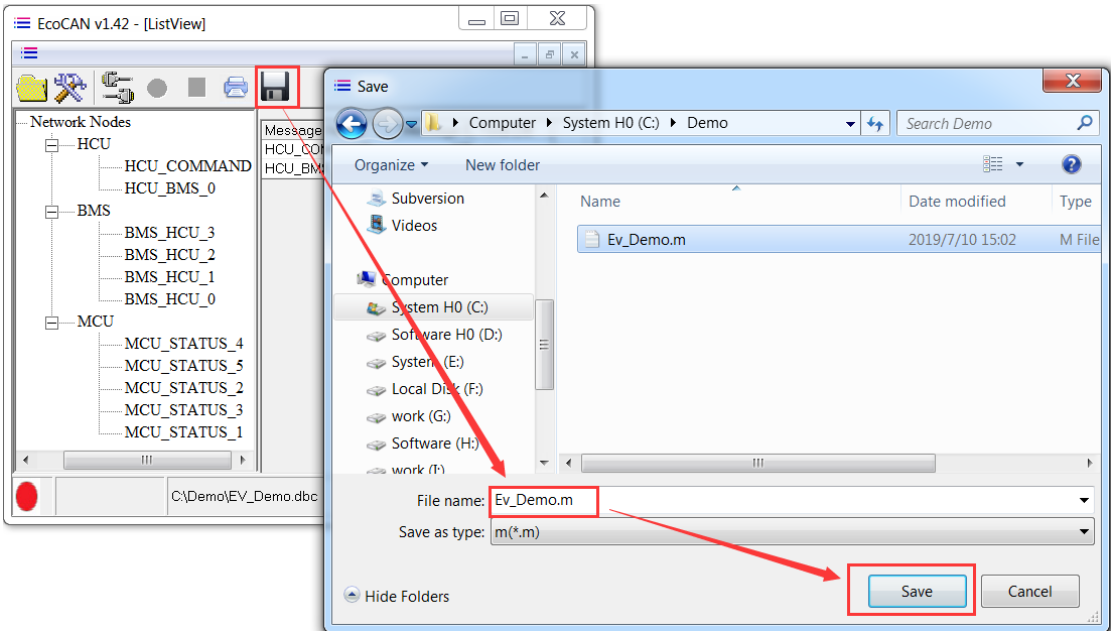
1. Open the DBC file to be converted in EcoCAN.




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



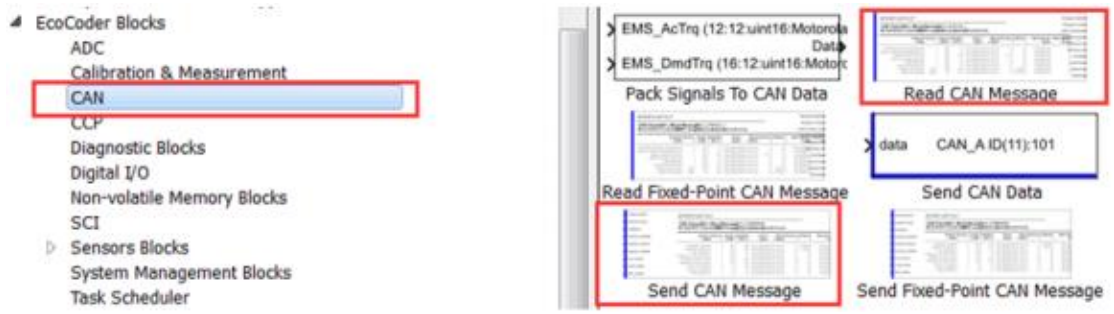

3. Click the indicated button and export the DBC file to m file, users can specify the saving path.



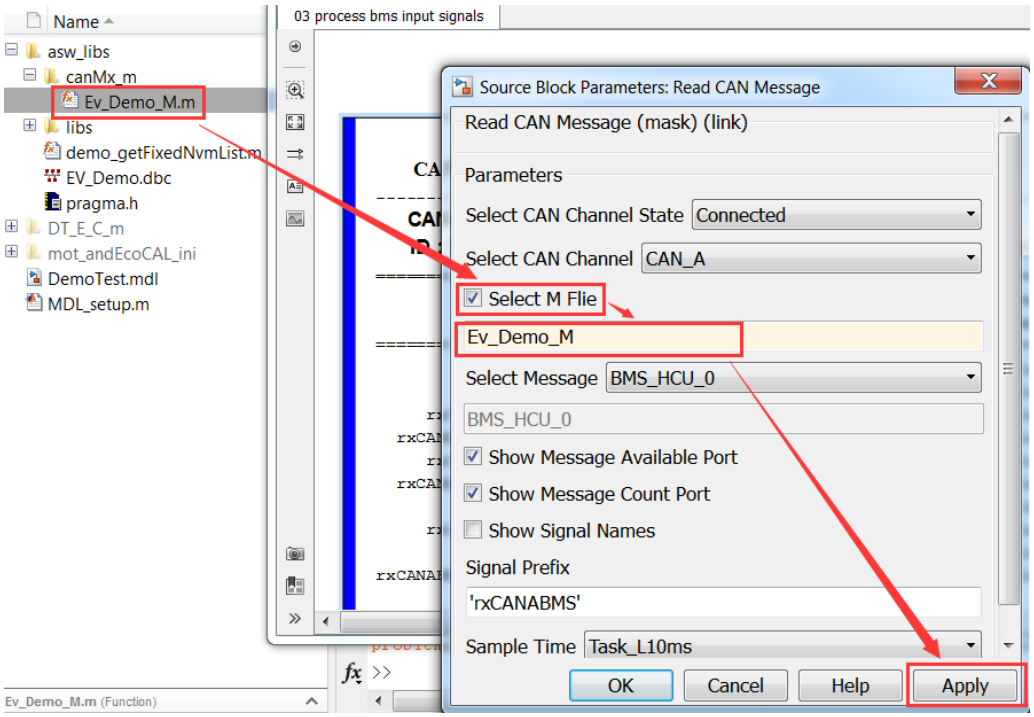
## 4.2 EcoCoder CAN Blocks

### 4.2.1 Select CAN library

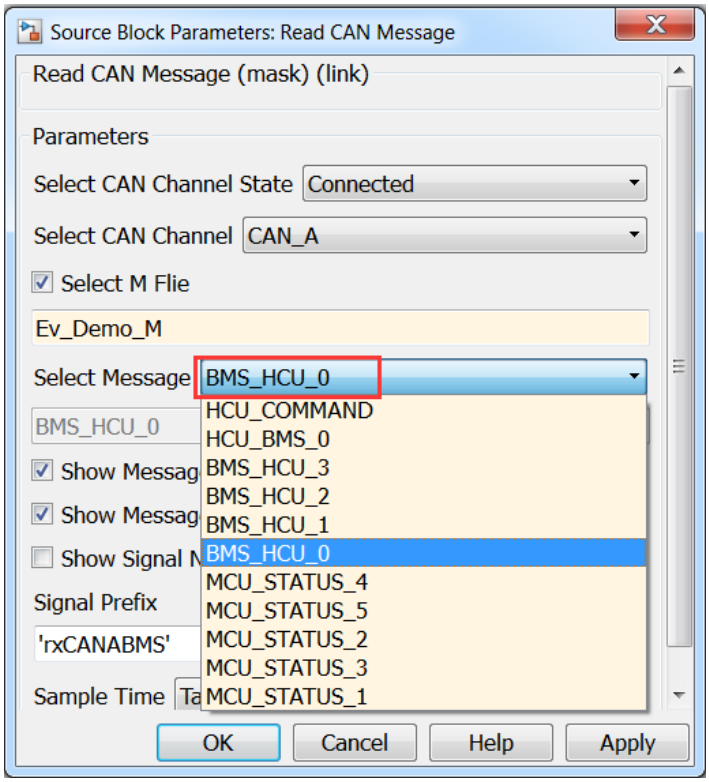
Please select 'Read CAN Message' or 'Send CAN Message' if fixed-point tool has not been installed in Matlab.



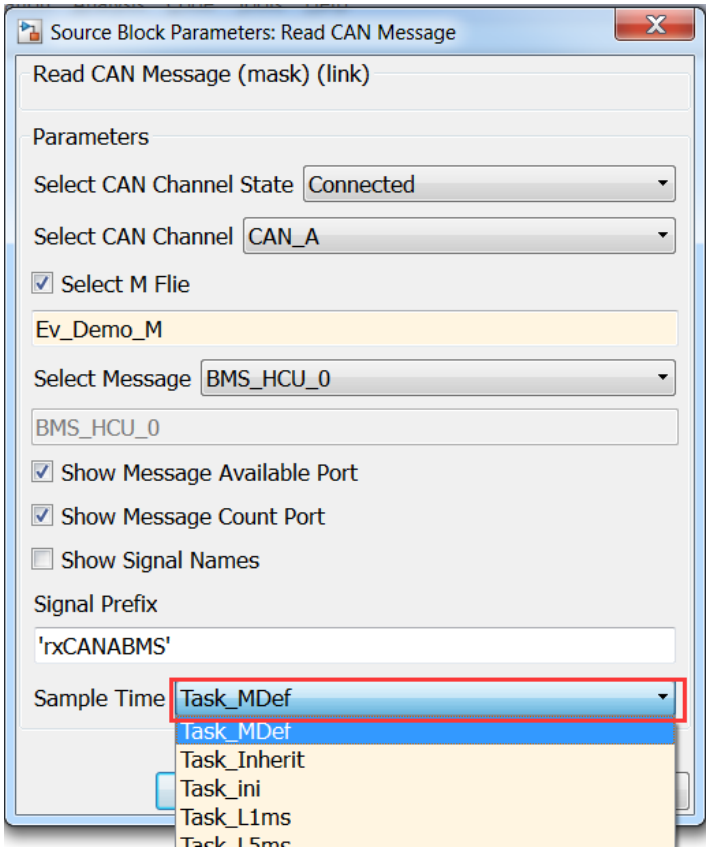

### 4.2.2 Select m file



### 4.2.3 Select CAN Message



### 4.2.4 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).

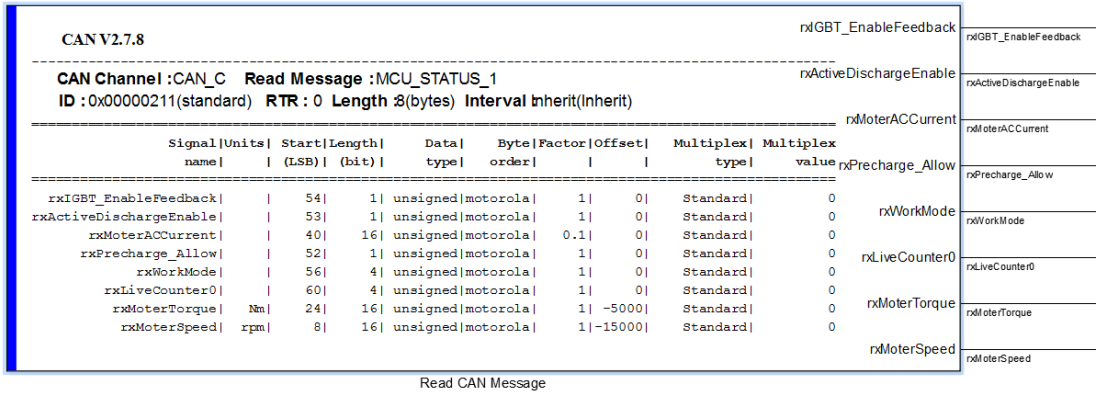
```
%Message Number:1
case 'HCU_COMMAND'
    ECOCAN.HCU_COMMAND = struct;
    ECOCAN.HCU_COMMAND .name = 'HCU_COMMAND';
    ECOCAN.HCU_COMMAND.description = 'HCU_COMMAND';
    ECOCAN.HCU_COMMAND.protocol  = 'ECOCAN';
    ECOCAN.HCU_COMMAND.id = hex2dec('113');
    ECOCAN.HCU_COMMAND.idext = 'STANDARD';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

The sampling time in the m file is as follows, and the periods that are not in the table need to be achieved by building the model:

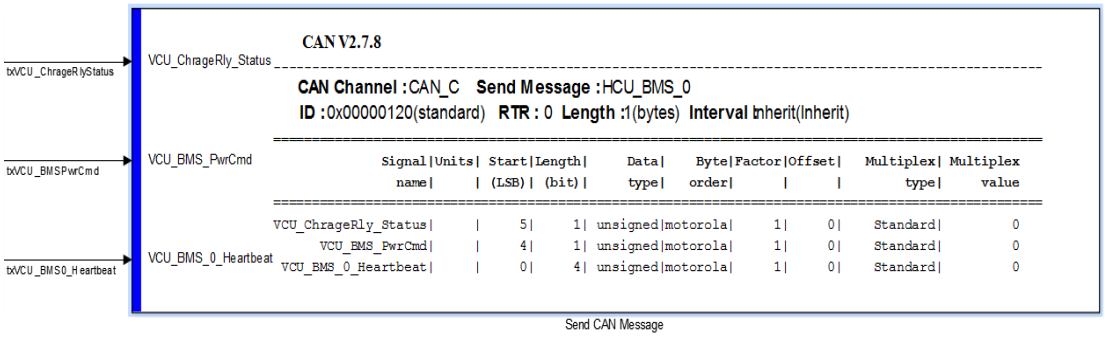
| | | | | | |
|---|---|---|---|---|---|
| Interval | -1 | 1 | 5 | 10 | 20 |
| Sample Time | Task_Inherit | Task_L1ms | Task_L5ms | Task_L10ms | Task_L20ms |
| Interval | 50 | 100 | 200 | 500 | 1000 |
| Sample Time | Task_L50ms | Task_L100ms | Task_L200ms | Task_L500ms | Task_L1000ms |

## 4.2.5 CAN demo model

The following are the use cases for CAN packaging and unpacking modules



Read CAN Message




Send CAN Message

# Chapter 5 Custom Variable Type

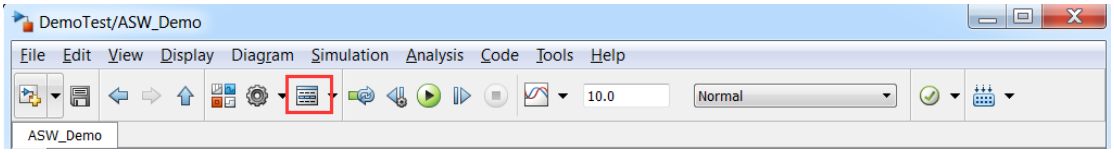
There are two ways to define monitoring/calibration/NVM variables. One is to custom variable type, and the other is to use the definition block in the EcoCoder library.

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.

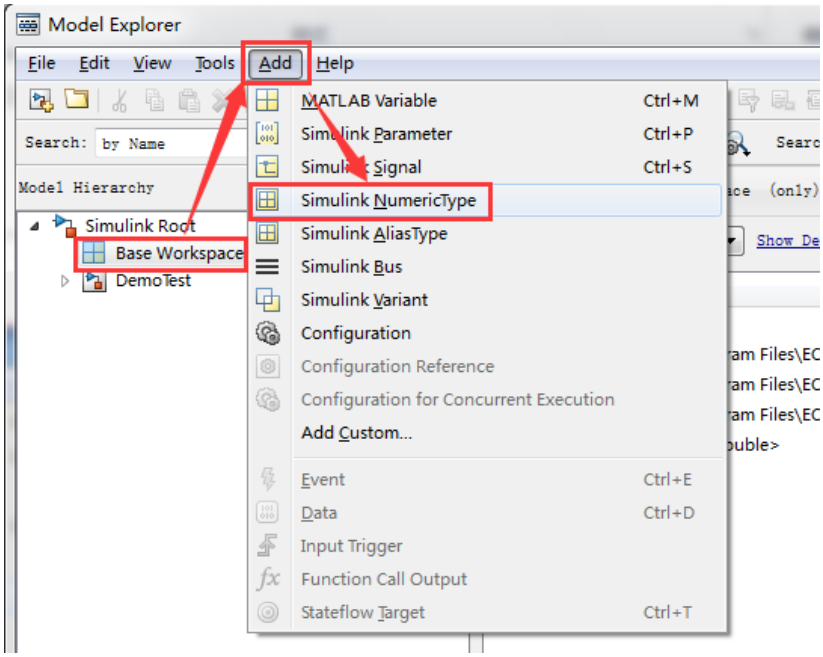
EcoObj is a custom data storage definition package. It is an extension of the simulated signal object and the simulation parameter object. Define custom data objects and classes by using the ASAP2 standard. It generates the product code and the ASAP2 file (or a2l file). You can use EcoObj or MATLAB's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

## 5.1 Customize Variable Types

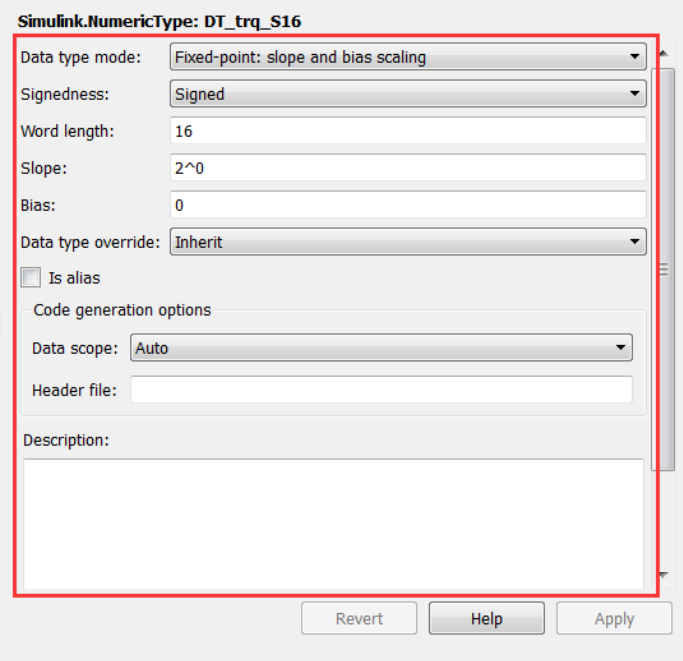
1. Open “Model Explorer”




2. Base Workspace > Add > Simulink Numeric Type




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

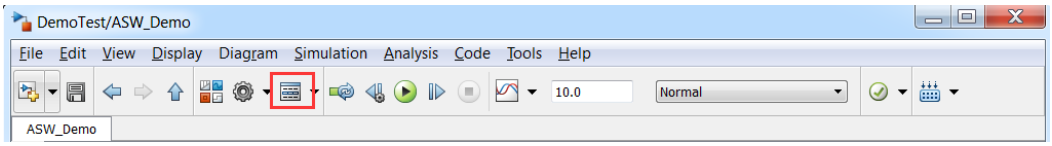



## 5.2 Add Variables to Workspace

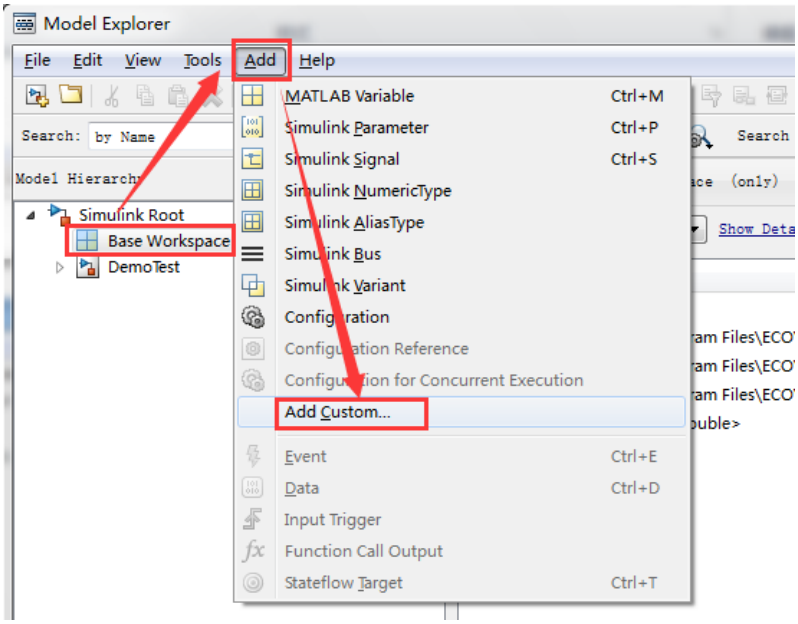
Add "EcoObj.Signal" or "EcoObj.Parameter" to "Base Workspace" via "Model Explorer" as

shown below.

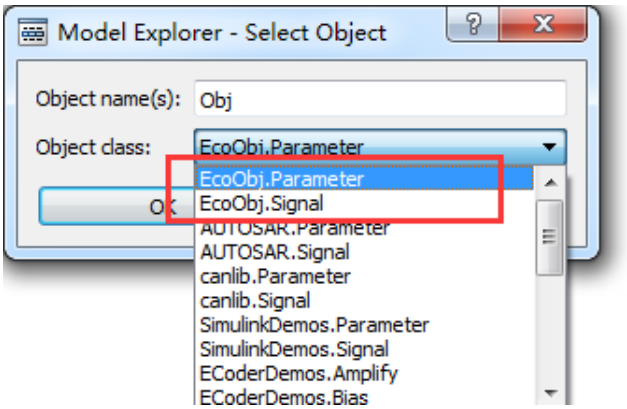
1. Open “Model Explorer”




2. Base Workspace > Add > Add Custom…

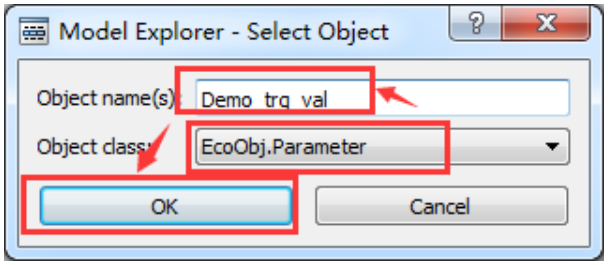



3. Click “Add Custom”, choose “EcoObj.Signal” or “EcoObj.Parameter”.

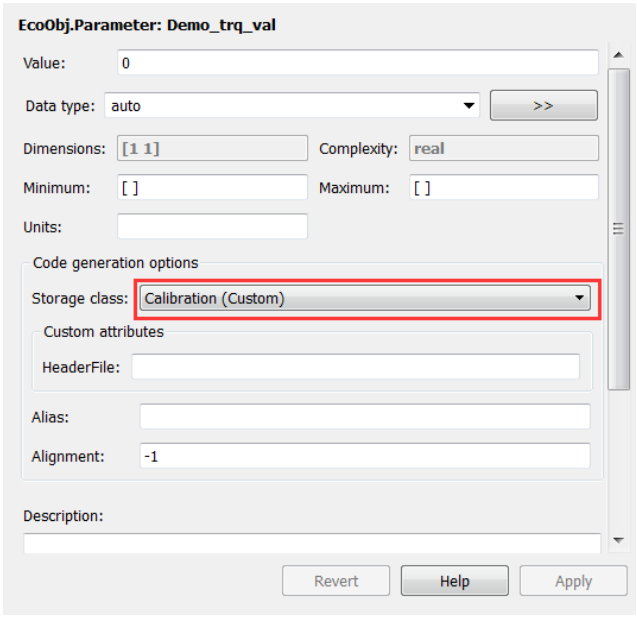


## 5.3 Customize Calibration Variables

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

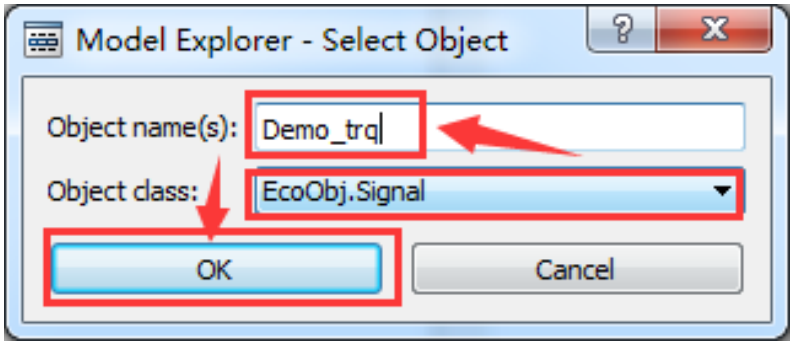



2. Set the properties through the window on the right. To define calibration variables, “Calibration(Custom)” must be chosen in “Storage Class”.

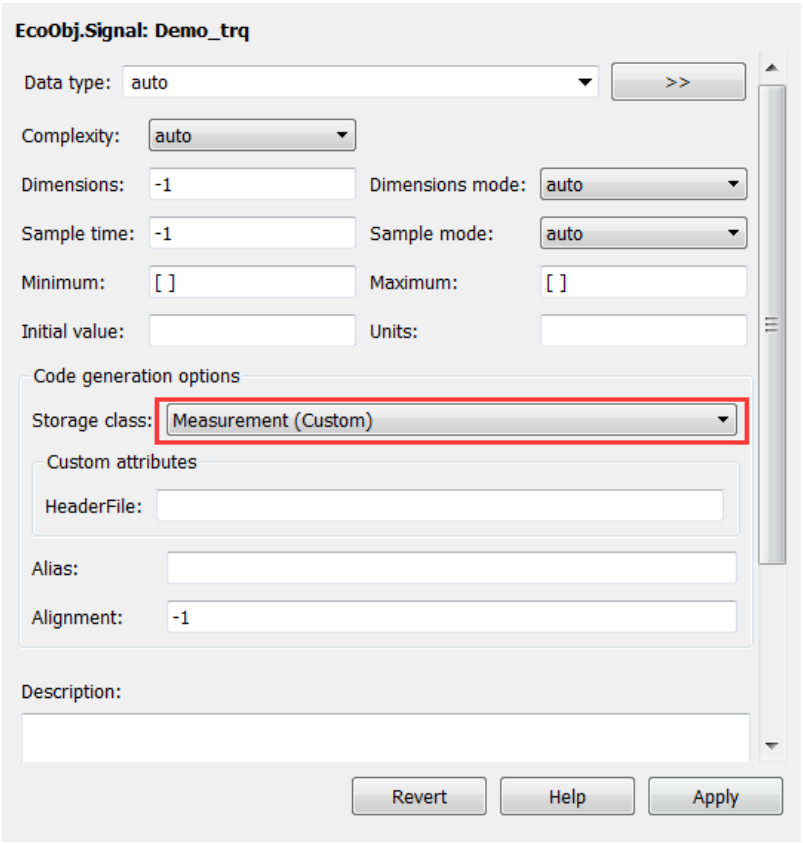


## 5.4 Customize measurement Variables

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



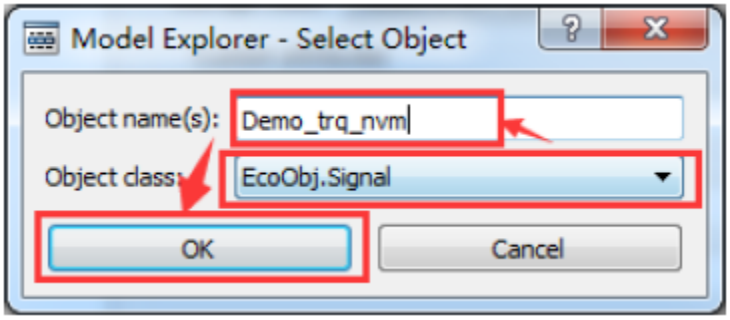

2. Set the properties through the window on the right. To define measurement variables “Measurement (Custom)” must be chosen in “Storage Class”.



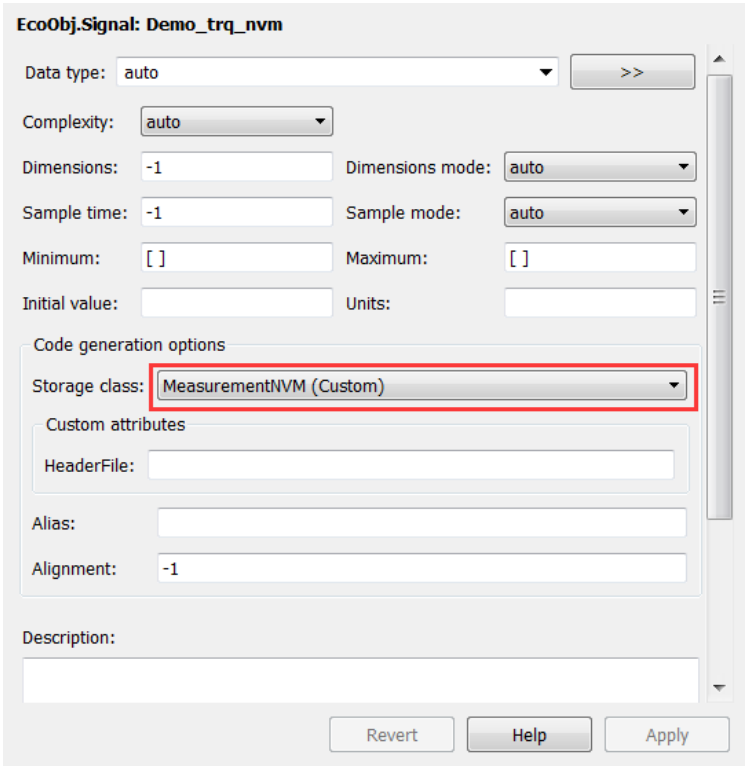
## 5.5 Customize NVM Variables

NVM variables are used to store data that can be saved after power failure, generally used to store fault codes or some vehicle information.

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



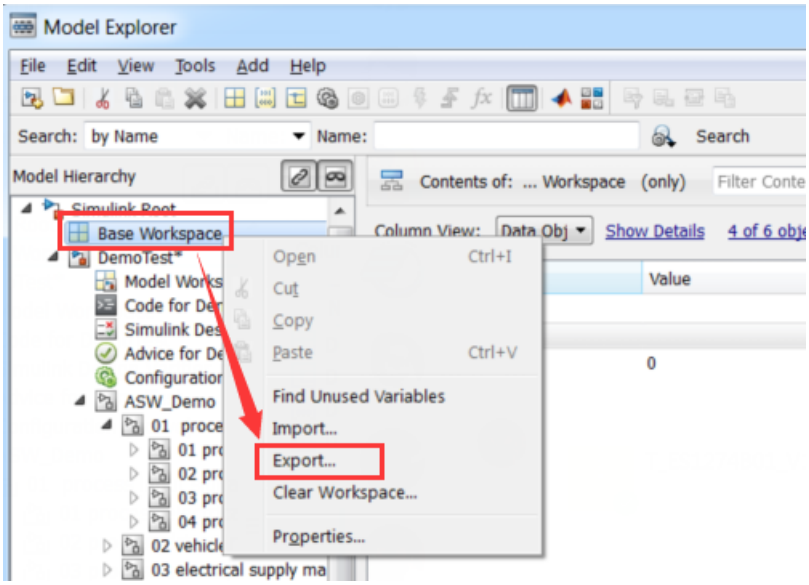

2. Set the properties through the window on the right. To define NVM variables, “MeasurementNvm (Custom)” must be chosen in “Storage Class”.



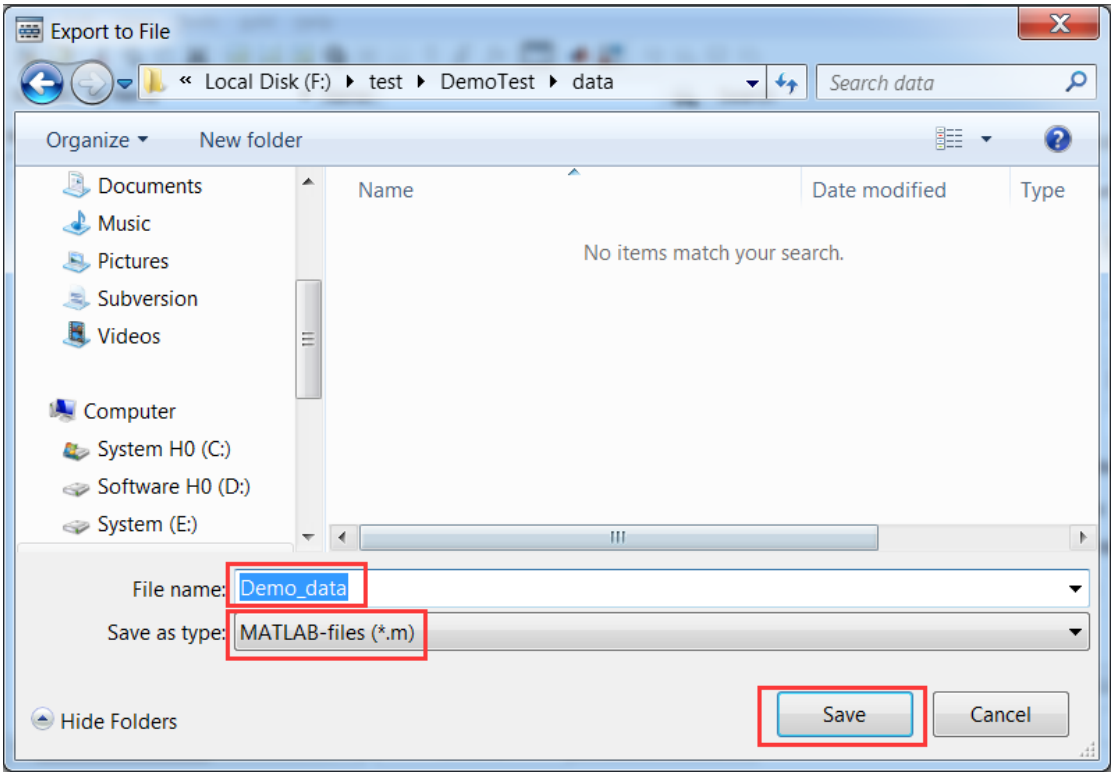
## 5.6 Save the Variables to M file

After "Base Workspace" defined variables, if you close MATLAB, the data will be lost, so the data should be saved to the m file in time after the data is defined.

1. Base Workspace > Export…

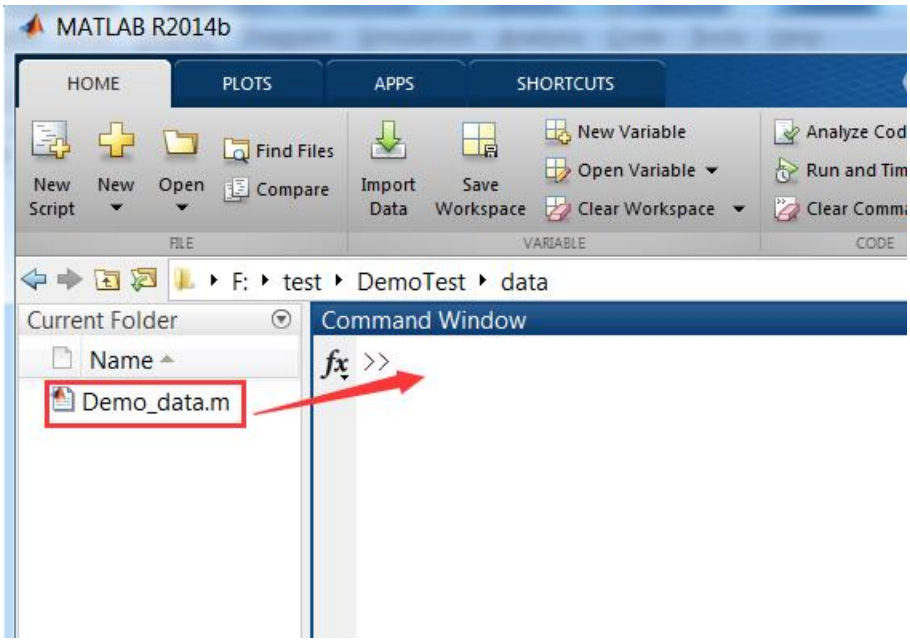



2. Click “Export…”, as shown below, save the file to “Demo_data.m”.

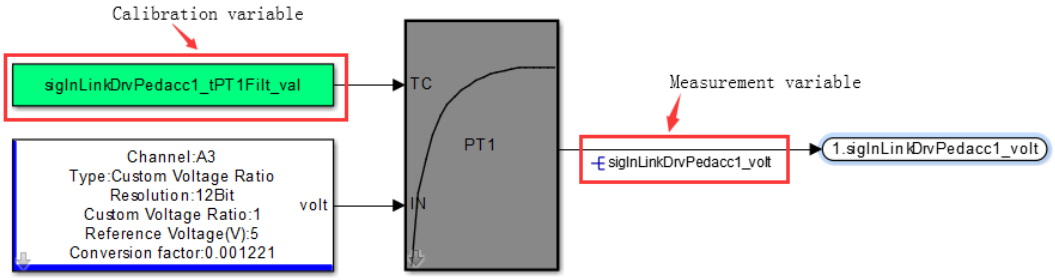


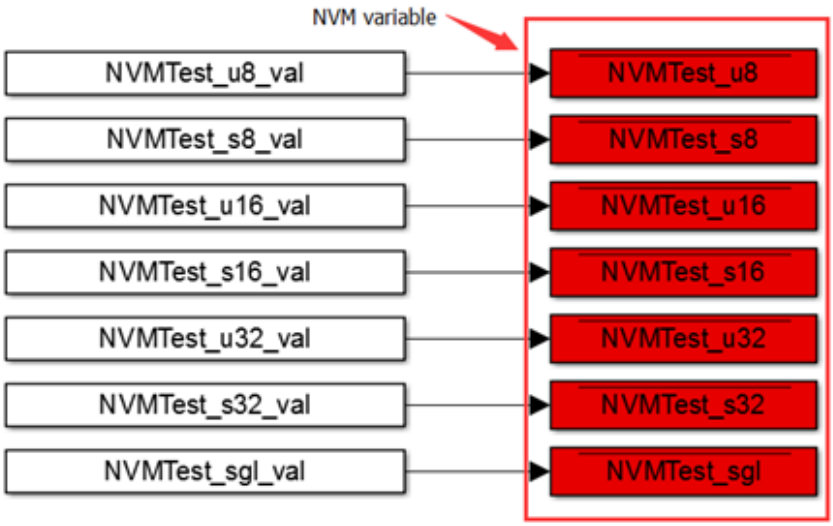

## 5.7 Load M file to Workspace

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



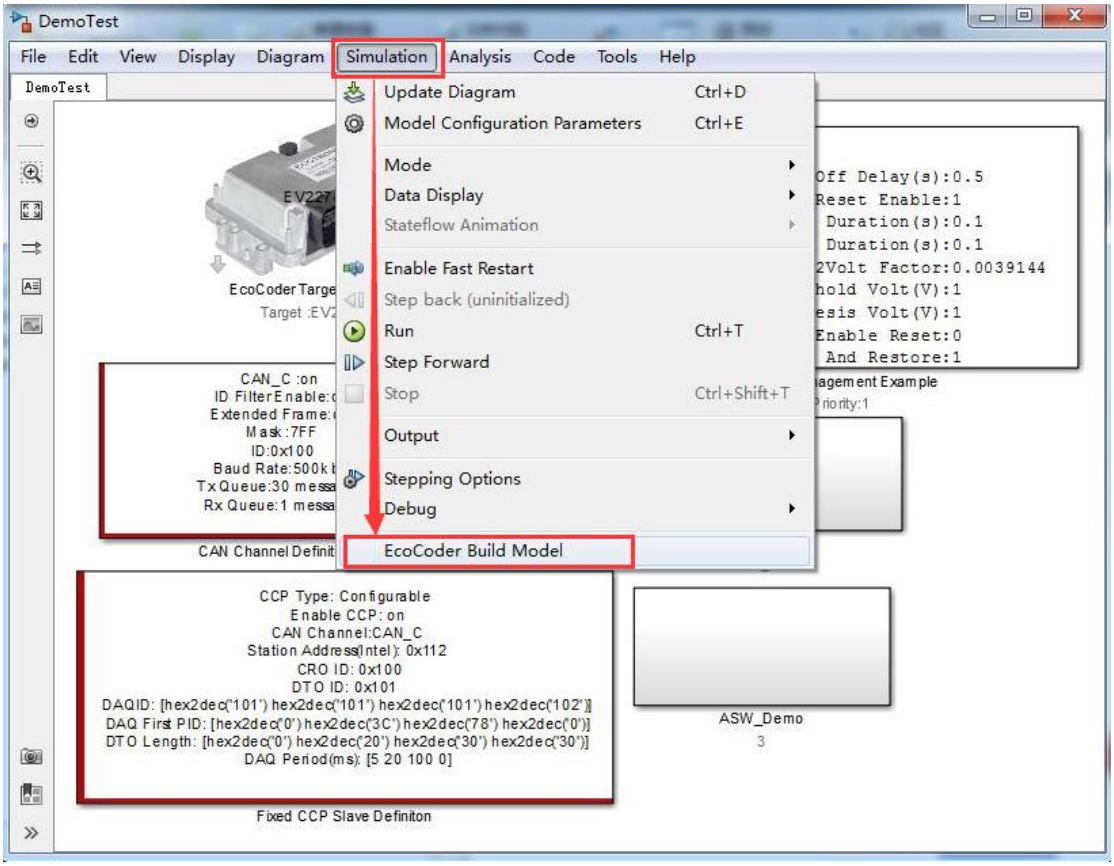
## 5.8 Model Example



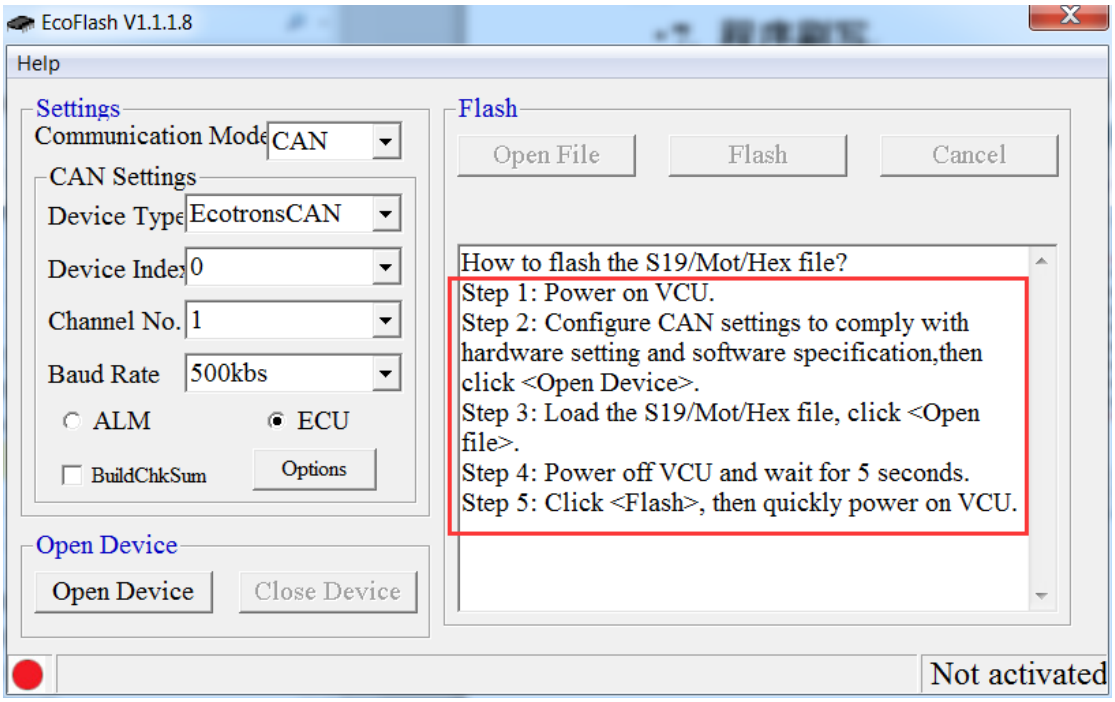
# Chapter 6 Automatically code generation

After the model simulation is finished, click the menu "Simulation->EcoCoder Build Model" to generate the executable file with one click.



# Chapter 7 Programming VCU with EcoFlash

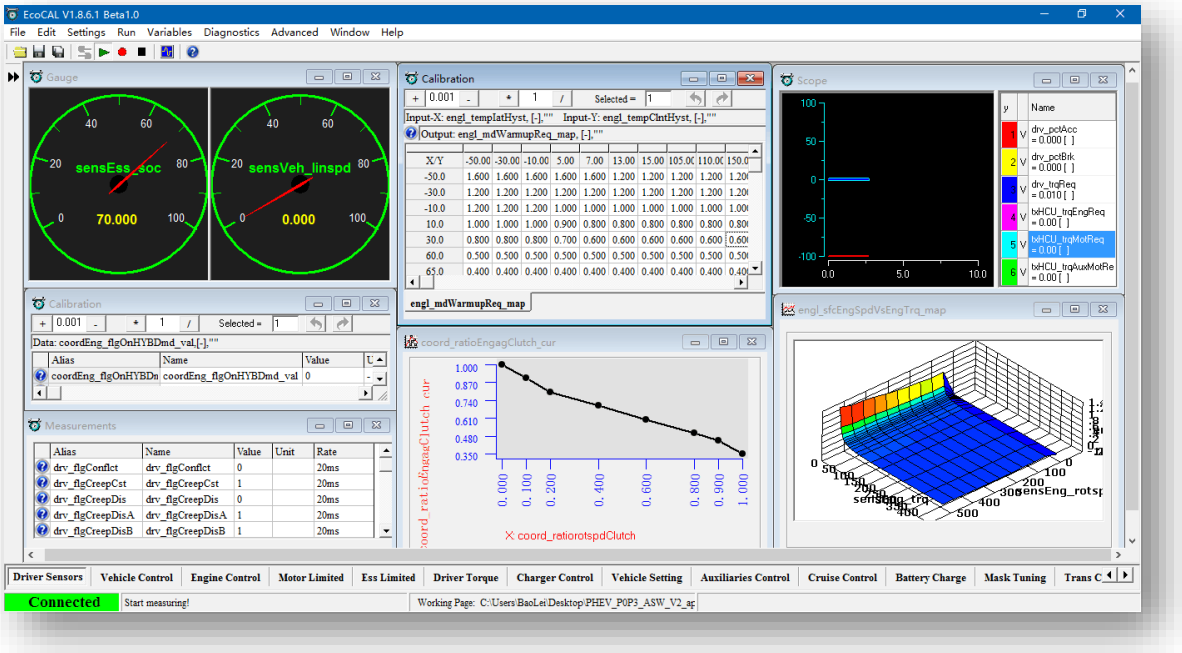
Flashing executable file through EcoFlash. Please refer to the tips in the red box when using.



# Chapter 8 Measurement and Calibration with EcoCAL

EcoCAL is a professional calibration and measurement tool with real-time measurement, real-time recording, online calibration offline flashing and other useful functions.

EcoCAL follows the CCP/XCP standard communication protocol, reads standard A2L files, and manages calibration data in CAL file format. It supports a wide range of communication devices, including CAN bus, USB, SCI (COM), and Ethernet. It has functions such as data playback, OBD online fault diagnosis, real-time reading of fault code, etc.



# Chapter 9 Uninstall EcoCoder

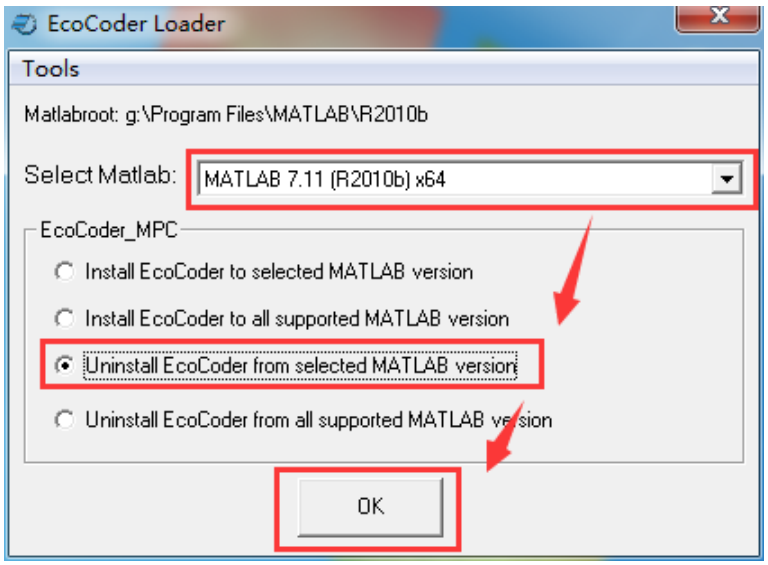
**Note: You have to close all MATLAB applications before uninstalling.**

## 9.1 Uninstall EcoCoder from MATLAB

Uninstalling EcoCoder from MATLAB is to let selected MATLAB to no longer start the EcoCoder Toolbox. The specific uninstall steps are as follows:

1. Double-click 'EcoCoder Loader'.

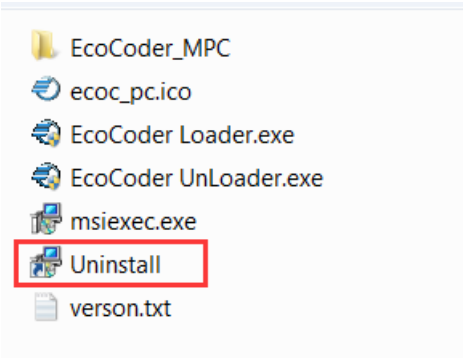

2. Choose MATLAB version, and select 'Uninstall EcoCoder from selected MATLAB version', then click 'OK'.



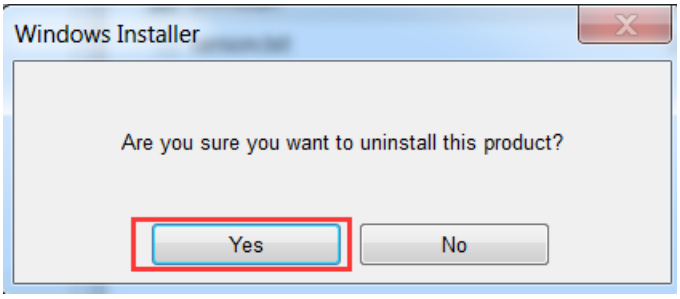
## 9.2 Uninstall EcoCoder from Windows System

Uninstalling EcoCoder from the operating system will not only make all MATLAB no longer start the EcoCoder Toolbox, but will also completely remove EcoCoder from the computer, as follows:

1) Uninstall via "Uninstall" in the EcoCoder installation directory.




2) Select "Yes" to uninstall.



# Chapter 10 Manually add the registry information

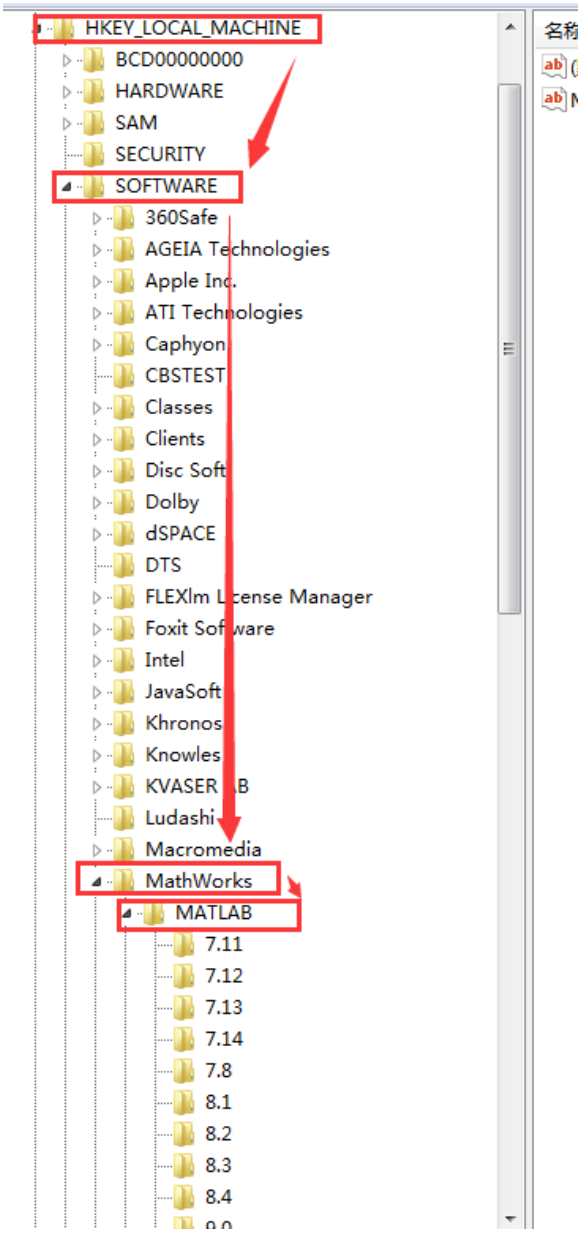
If the current MATLAB is an none-installation version, or if the computer has reinstalled the system before, then it will have no information in the MATLAB registry, and the existing MATLAB version cannot be found in the EcoCoder Loader after the installation is complete. In this case, you need to manually add MATLAB registry information, as follows:

1. Enter regedit into the command window to open registry editor.

2. Add MATLAB 64bit registry information under a 64-bit system is as follows:

Information needs to be added under

HKEY_LOCAL_MACHINE\SOFTWARE\MathWorks\MATLAB




For example, to add MATLAB R2019a, you need to right-click at MATLAB to create a new item 9.6. The version of MATLAB needs to be found by customer themselves.

Then right-click in item 9.6 to create a new string value,

Name the string value MATLABROOT and the value is the path to MTLAB 2019a

3. Add MATLAB 32bit registry information under a 64-bit system as follows:
   The method is the same as #2, but only add new item under
   HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\MathWorks\MATLAB

Although you can add the registry manually, it is recommended to reinstall MATLAB if you can reinstall it.

# Chapter 11 FAQs

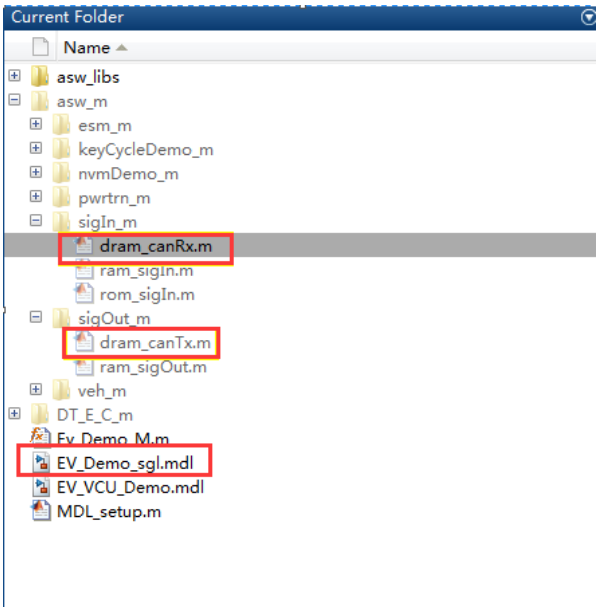
## 11.1 Q1. The m file exported from DBC by 'EcoCAN' can't be used

Why m files exported from DBC with "EcoCAN" cannot work?

Check whether the name of the m file follows the c language variable naming rules, and cannot be the name of an existing model or an existing m file. For example, "0-. Demo1.2" or "EcoCanM_Demo" is illegal.

## 11.2 Q2. Model created by 'EcoCoder_Prj', simulation 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. Task scheduling priority

H tasks have higher priority than L tasks. For all H tasks, the shorter period, the higher priority. All L tasks have the same priority.

## 11.5 Q5. EcoCoder Loader Pop-up error


When using ecoCoder Loader and pop up the above window, you need to register the control "comdlg32.ocx". The easiest way to do this is to install it according to the "2.1 Software Installation Order" section, or install EcoFlash before installing EcoCoder.

## 11.6 Q6. Cancel window pops up after the code is generated

Is there a way not to pop up the folder prompt window after the one-click code is generated?

You can set this up by following these steps:

1) Open Model Configuration Parameters.

2) Click EcoCoder Debug Options in the Code Generation directory.

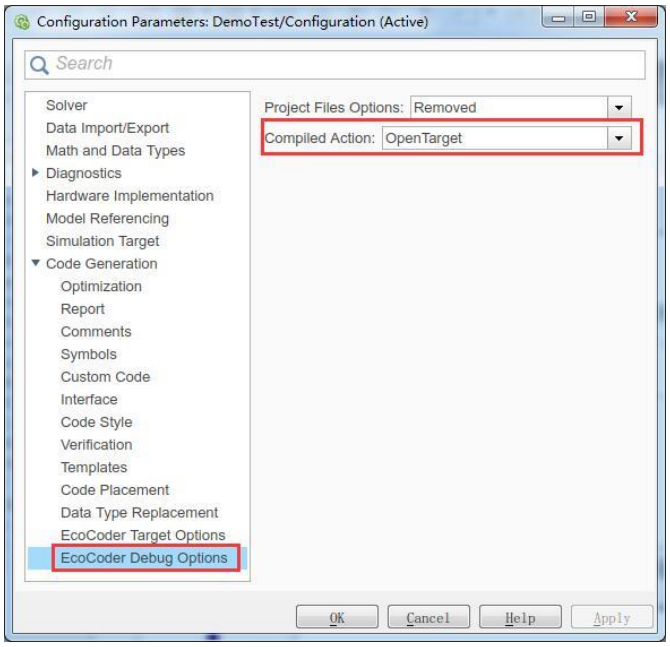
3) Set Compiled Action menu.

The Compiled Action menu contains 3 options, and the meanings of the different options are as follows:

1) No Prompt: One-click code is generated without any prompt.

2) OpenTarget: The generated folder is automatically opened after the one-click code is generated.

3) PopupBox: After the one-click code is generated, only the prompt box "Software has been compiled successfully!" appears.




## 11.7 Q7. How to keep a C code project

You can set this up by following these steps:

1) Open Model Configuration Parameters.

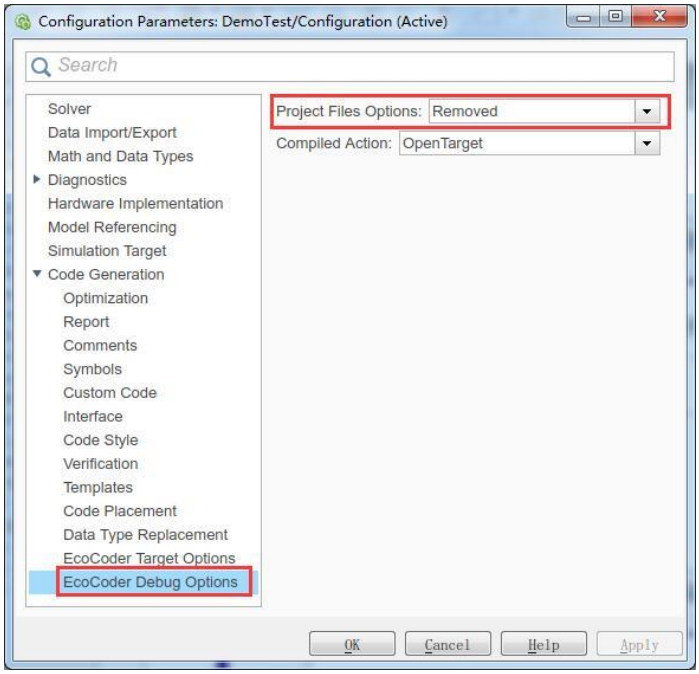
2) Click EcoCoder Debug Options in the Code Generation directory.

3) Set Project Files Options menu.

Project Files Options menu contains 2 options, and the meanings of the different options are as follows:

1) Reserved: After one-click file generation, the 'XX_CWPrj' folder is retained, and the user can freely access the generated code.

2) Removed: After generating a file with one click, the 'XX_CWPrj' file is deleted.




## 11.8 Q8. How to update application model to be compatible with updated EcoCoder

How do I update the user-built application model to be compatible with the latest version of EcoCoder?

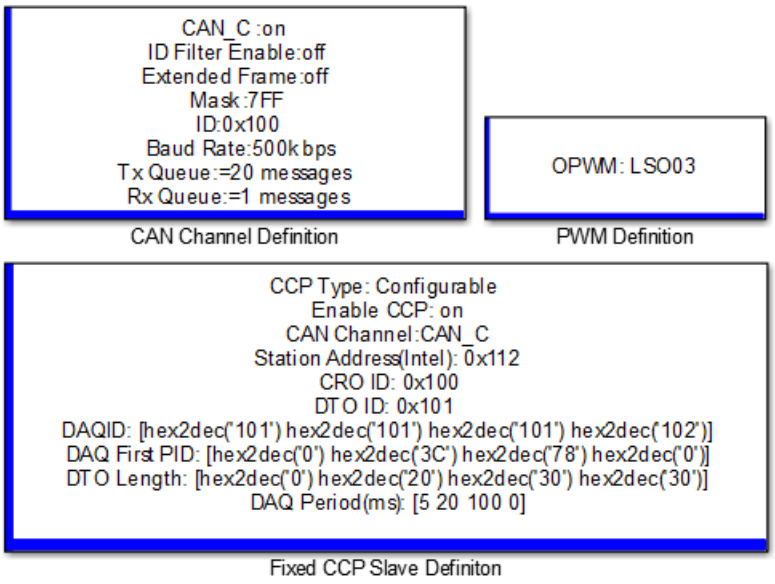
You can complete the model-compatible installation package by following these steps:

1) Add EcoCoder Target Definition, the model must contain EcoCoder Target Definition.


EcoCoder Target Definition

Target :EV2274A

2) Model configuration. The new version requires independent model configuration modules, such as CAN, OPWM, CCP are added as required.




CAN Channel Definition

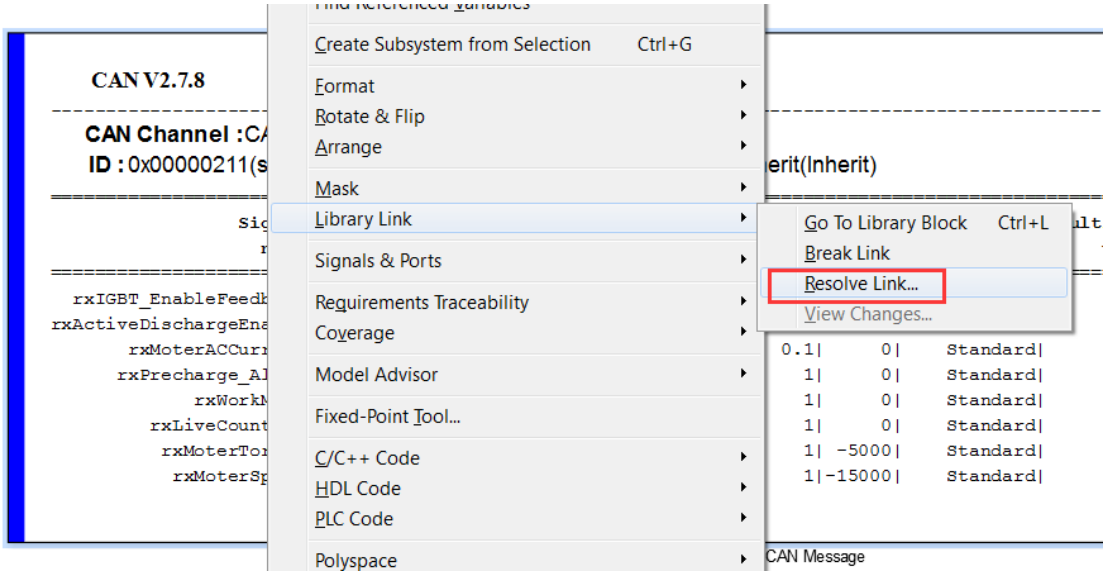
PWM Definition

Fixed CCP Slave Definiton

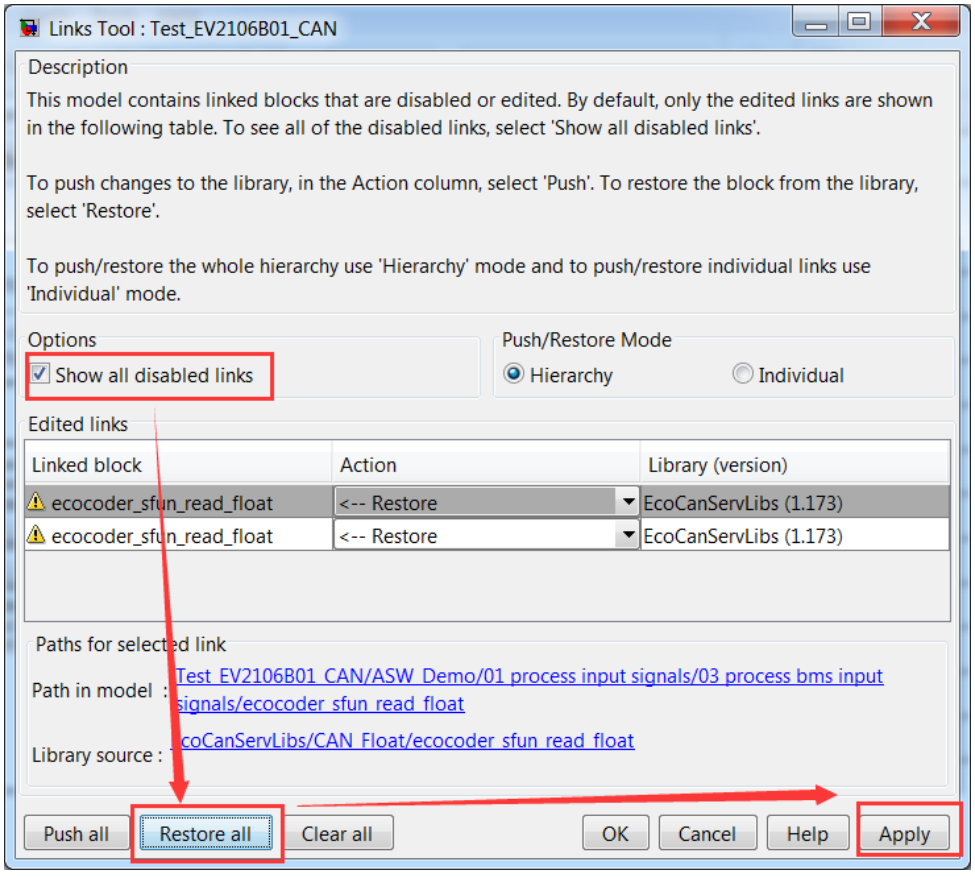
3) Resolve some Disable modules.

If the CAN module library used in the model is disable, you need to restore all the CAN modules first, save the model and then upgrade EcoCoder, otherwise the original model will be stuck after the upgrade. The upgrade steps are as follows:

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



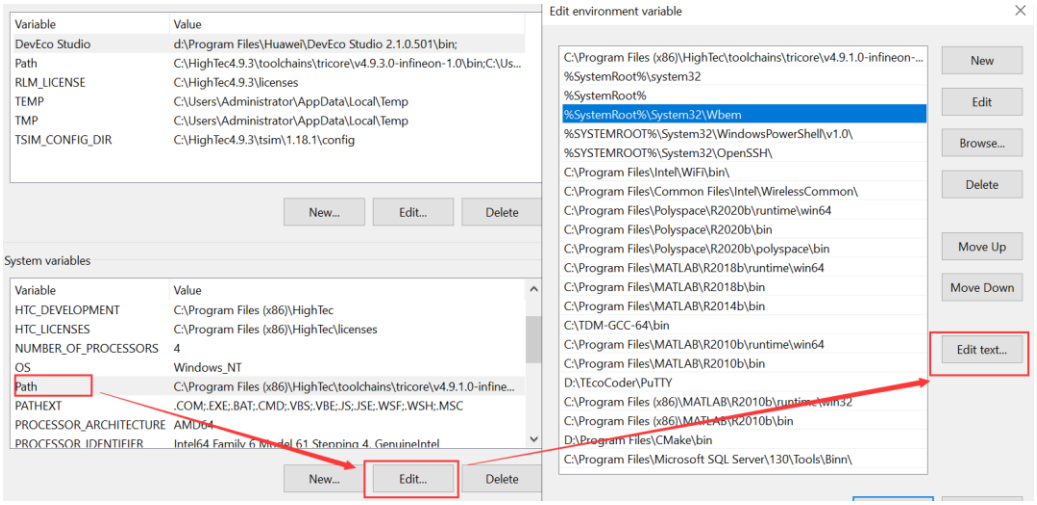

2) Restore all disable linked blocks.



## 11.9 Q9. Compilation error FAQ

### 11.9.1 Win10 system failed to add environment variable

WIN10 system add PATH environment variable needs to follow the steps in dialog box in the manual, not in list, as shown in the following:




### 11.9.2 After adding the environment variable, need to restart MATLAB

### 11.9.3 Added an environment variable with splitter error

Note The splitter for adding environment variables must be a semicolon in English.

### 11.9.4 EcoCoder Loader selects the sysroot path incorrectly

Note that using EcoCoder Loader selects the path of the compiler's sysroot correctly.

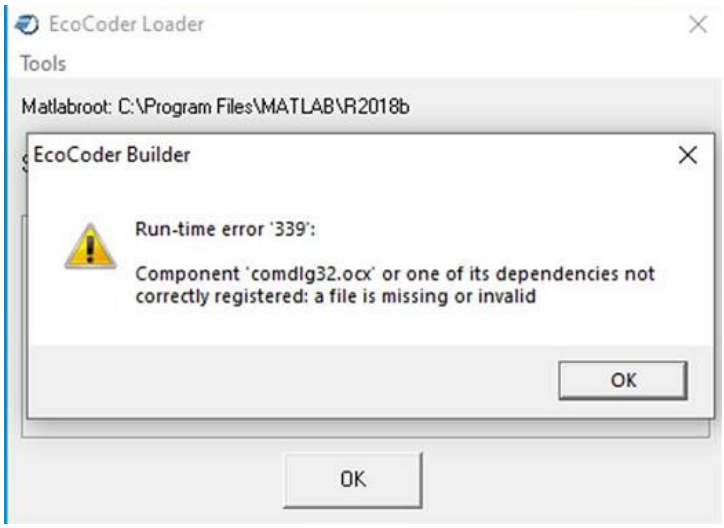
### 11.9.5 Current model path has none-English, spaces, or special characters

When there is a "No such file or directory" prompt in the Dialog Viewer prompt, it is generally the reason for the none-English, space, or special character in the path, please replace it with a path without none-English characters.

## 11.10 Q10-EcoFlash default DLL

The bootloader of the controller whose main chip is Infineon TC27x/TC29xx/TC39xx is CCP protocol, the factory default DLL file is PG_EH2175A.dll, and the other controllers are PG_Default.dll. The file name can be directly searched under the EcoCoder installation directory.

## 11.11 Q11-EcoCoder Loader indicates that COMDLG32.OCX is missing




At present, three solutions are recommended to try to solve them

1. Install EcoFlash. Since EcoFlash will automatically register COMDLG32.OCX

2. In the installation directory after installing EcoCoder, find the file oxcFiles .zip, and run regsvr32oxc .bat after extracting it

3. Search on internet for COMDLG32.OCX and find how to register.

# Chapter 12 Appendix

Table 1

| Status number | Status Description |
|---|---|
| 0 | Operation successful |
| 1 | Not enough space, free area block is smaller than the setting active block |
| 2 | Flash operation failed |
| 3 | Block operation failed |
| 4 | Block detect failed |
| 5 | Not enough space to write |
| 6 | Need erase |
| 7 | Block status abnormal |
| 8 | Parameter error |
| 9 | Record not found |
| 10 | Record type not matched |
| 11 | Record has been deleted |
| 12 | Record copy successfully |
| 13 | Recoding writing |
| 14 | Doing swap operation |

| | |
|---|---|
| 15 | Record needs to be written into new action block |
| 16 | Need erase |
| 17 | Read length not matched |

Table 2

| Status number | Status Description |
|---|---|
| 0 | Bus normal |
| 1 | Parameter error |
| 6 | Bus busy |