Contact us:

Website:  http://www.ecotrons.com
Email:  info@ecotrons.com
        ev-support@ecotrons.com
CONTENTS

CHAPTER 1 GENERAL INFORMATION .................................................................................. 8

1.1 About EcoCoder ........................................................................................................ 8

1.2 System Requirements ................................................................................................. 10

1.3 MATLAB Installation Requirements .......................................................................... 10

1.4 Supported MATLAB Version ...................................................................................... 10

1.5 Developer’s Kit ............................................................................................................ 11

CHAPTER 2 DEVELOPMENT ENVIRONMENT WITH ECOCODER ......................... 12

2.1 Software Installation List ............................................................................................. 12

2.2 C++ Compiler Installation .......................................................................................... 12

2.2.1 Installation of Compiler for MATLAB 32-Bit ......................................................... 12

2.2.2 Compiler Selection for MATLAB 64-Bit ................................................................. 13

2.3 EcoCoder Installation .................................................................................................. 14

2.4 Link HighTec TriCore Tool Chain to EcoCoder ......................................................... 18

2.5 Activate EcoCoder ....................................................................................................... 19

2.5.1 Get Key File ............................................................................................................. 20

2.5.2 Activate EcoCoder by License (.dat) File ................................................................. 22

CHAPTER 3 QUICK START ON APPLICATION SOFTWARE .................................. 25

CHAPTER 4 ECOCODER LIBRARY ................................................................................. 27

4.1 EcoCoder Target Definition ....................................................................................... 28

4.2 ADC ............................................................................................................................. 29

4.2.1 Read ADC Value ....................................................................................................... 29
4.2.2 Read Fixed-Point ADC Volt ................................................................. 32
4.2.3 Read Float ADC Volt ............................................................................. 33

4.3 CAN Communication ............................................................................. 34
  4.3.1 CAN Channel Definition ....................................................................... 34
  4.3.2 CAN Wake-up Frame Definition ............................................................. 37
  4.3.3 Read Fixed-Point CAN Message ............................................................ 38
  4.3.4 Send Fixed-Point CAN Message ............................................................ 40
  4.3.5 Read/Send CAN Message ....................................................................... 42
  4.3.6 CAN Receive Counter ............................................................................ 42
  4.3.7 Set CAN Mode ....................................................................................... 43
  4.3.8 Send CAN Data ..................................................................................... 44
  4.3.9 Unpack Signals to CAN Data ................................................................. 45
  4.3.10 Pack Signals To CAN Data ................................................................. 46
  4.3.11 Receive CAN Message ......................................................................... 47
  4.3.12 Transmit CAN Message ....................................................................... 48

4.4 Serial Communication Interface (SCI) Block ........................................... 48
  4.4.1 SCI Definition ....................................................................................... 49
  4.4.2 Read SCI Data ..................................................................................... 49
  4.4.3 Send SCI Data ..................................................................................... 50

4.5 Digital I/O ............................................................................................... 50
  4.5.1 Switch Input ......................................................................................... 50
  4.5.2 KeyOn Input ......................................................................................... 51
  4.5.3 Switch Output ..................................................................................... 53
  4.5.4 IPM Read ............................................................................................. 54
  4.5.5 PWM Definition ................................................................................... 54
  4.5.6 PWM Output ....................................................................................... 56
  4.5.7 WakeUp Input ...................................................................................... 57
  4.5.8 H-bridge Definition ............................................................................. 58
  4.5.9 H-bridge Output .................................................................................. 60
  4.5.10 PWM IO Frequency Range Definition ............................................. 61
  4.5.11 IPWM Read ....................................................................................... 62

4.6 Task Scheduler ....................................................................................... 62
4.7 Non-Volatile Memory Blocks ........................................................................ 64
  4.7.1 Fixed NVM Definition ........................................................................... 64
  4.7.2 Read Fixed NVM ................................................................................ 65
  4.7.3 Write Fixed NVM ................................................................................ 66
  4.7.4 NVM Definition .................................................................................. 67
  4.7.5 NVM Variable Definition ..................................................................... 68
  4.7.6 Read NVM ................................................................................................ 69
  4.7.7 Write NVM ........................................................................................... 70
  4.7.8 Store All NVM Data ............................................................................. 71
  4.7.9 Restore All NVM Data .......................................................................... 72

4.8 Diagnostic Blocks .......................................................................................... 72
  4.8.1 Hardware Output DTC .......................................................................... 73
  4.8.2 DTC Parser .......................................................................................... 73
  4.8.3 Software Core Diagnostic .................................................................... 74
  4.8.4 Clear H-bridge DTC ............................................................................. 75

4.9 Calibration & Measurement .......................................................................... 76
  4.9.1 Calibration Definition .......................................................................... 76
  4.9.2 Read Calibration .................................................................................. 77
  4.9.3 Write Measurement ............................................................................. 78
  4.9.4 Write and Read Measurement ............................................................... 79
  4.9.5 Override Probe .................................................................................... 80
  4.9.6 1-D Lookup Table .............................................................................. 81
  4.9.7 2-D Lookup Table .............................................................................. 82
  4.9.8 Calibration Data Check ........................................................................ 84

4.10 System Management Blocks ....................................................................... 84
  4.10.1 Power Management Example .............................................................. 84
  4.10.2 Shutdown Power .................................................................................. 87
  4.10.3 Set ECU Mode .................................................................................... 88
  4.10.4 ECU Master Chip Wake-Up Definition .............................................. 88
  4.10.5 Watchdog Definition .......................................................................... 89
  4.10.6 Software Reset .................................................................................... 91
CHAPTER 5 CAN THEORY OF ECOTRONS

5.1 Introduction

5.2 CAN Implementation

5.2.1 Convert DBC to m File

5.2.2 EcoCoder CAN Blocks

5.2.3 Select m file

5.2.4 Select Message

5.2.5 Select Sample Time

CHAPTER 6 MEMORY MANAGEMENT
6.2 Storage device ........................................................................................................ 113

6.3 Data Storage ........................................................................................................ 113
  6.3.1 Calibration/Measurement Variable ................................................................. 113
  6.3.2 Non-Volatile Variable ..................................................................................... 114

CHAPTER 7 PROGRAMMING VCU WITH ECOFLASH ............................................. 115

CHAPTER 8 MEASUREMENT AND CALIBRATION WITH ECOCAL ..................... 116

CHAPTER 9 UNINSTALL ECOCODER ..................................................................... 117
  9.1 Uninstall EcoCoder from MATLAB ................................................................. 117
  9.2 Uninstall EcoCoder from Windows System ..................................................... 118

CHAPTER 10 FAQS .................................................................................................. 119

APPENDIX A - NONVOLATILE VARIABLES THEORY ........................................ 124
  Non-volatile Variables .......................................................................................... 124
  Fixed Non-volatile Variables .............................................................................. 124
  Battery Input ....................................................................................................... 125
Chapter 1 General Information

1.1 About EcoCoder

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

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

Plus, EcoCoder is only an add-on package on top of Simulink. It enables engineers to maximize the usage of Simulink generic library. It adds the necessary library blocks which bridge the gap between application software and the specific controller hardware.

Meaning the application software will not be dependent on the specific hardware, and you can port your models to any other hardware which supports the Simulink. In short, you are not stuck with EcoCoder by using it.

Features:

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

Benefits:
- Controls engineers could be freed from time-consuming learning curve of hardware, C programming, and specific microprocessor settings
- Responsive support services from Ecotrons
- Application software development is isolated from a specific hardware, and it has transparency and easy migration to other platforms.
1.2 System Requirements

<table>
<thead>
<tr>
<th>OS</th>
<th>Windows XP/Windows 7/Windows 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel CORE 2 Duo or higher</td>
</tr>
<tr>
<td>Memory</td>
<td>2 GB or higher</td>
</tr>
<tr>
<td>Hard drive</td>
<td>1 GB free hard disk space</td>
</tr>
</tbody>
</table>

1.3 MATLAB Installation Requirements

Mandatory Components:
- MATLAB
- Simulink
- Simulink Coder
- Embedded Coder

Optional Components:
- 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

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

1.5 Developer’s Kit

• VCU
• Test Harness*
• USB-CAN Adapter*

* Test harness is available from Ecotrons, however, users could also make their own by using recommended connector parts.

* USB-CAN adapter need to be compatible with CAN Calibration Protocol (CCP). Third party adaptor like Kvaser and PeakCAN are compatible with Ecotrons products.
Chapter 2 Development Environment with EcoCoder

2.1 Software Installation List

Please install software tools in the following order*:

1. CodeWarrior for MPC55xxMPC56xx v2.10.exe
2. EcoFlash Vxxxx Setup.exe
3. EcoCAL Vxxxx Setup.exe
4. Ecotrons USB-CAN Driver
5. C++ Compiler (only install when Stateflow is included in customer application)
6. EcoCoder 56xx Vxxxx Setup.msi

* Note: the above is an example tool chain for NXP MPC 56xx family microcontrollers. For Infineon Aurix family or NXP MPC 57xx family, or other family of microcontrollers, the tool chains will be different. But the installation process is pretty much same. Contact us for details on different tool chains.

2.2 C++ Compiler Installation

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

2.2.1 Installation of Compiler for MATLAB 32-Bit

1. Type ‘mex -setup’ at Matlab Command Window.
2. Type ‘y’ at Command Window.

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

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

3. Type ‘1’ at Command Window.

   Select a compiler:
   \[1\] lcc-win32 C 2.4.1 in C:\PROGRAM\MATLAB\R2010b\sys\lcc
   \[0\] None

   Compiler: \[1\]

4. Type ‘y’ at Command Window.

   Compiler: lcc-win32 C 2.4.1
   Location: C:\PROGRAM\MATLAB\R2010b\sys\lcc

   Are those correct [y]/[n]   \[y\]

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

   Trying to update options file: C:\Documents and Settings\Administrator\Application Data\MathWorks\MATLAB\R2010b\mexopts.bat
   From template: C:\PROGRAM\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:
   Building with the -largeArrayDims option enables the new API.
   **************************************************

### 2.2.2 Compiler Selection for MATLAB 64-Bit

1. Go to this official website of Mathworks

   https://www.mathworks.com/support/sysreq/previous_releases.html
2. Click ‘Details’ under ‘Supported Compilers’ of MATLAB version on customer PC.

3. For Windows 64-bit system, refer to page titled ‘Windows 64bit’.

4. After finishing the compiler installation, follow the steps in previous section to configure compiler for MATLAB.

### 2.3 EcoCoder Installation

---

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

---

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

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

3. Click ‘Next’.
4. Choose the version of MATLAB you want to install EcoCoder to, then select ‘Install EcoCoder to selected MATLAB version’, click ‘OK’. You could also install EcoCoder for all MATLABs on computer.

5. Click ‘Close’.

6. After the installation is successful, 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 start MATLAB, it will prompt message ‘EcoCoder has been installed successfully’ as shown in following red box. It indicates EcoCoder is successfully installed to MATLAB.
2.4  Link HighTec TriCore Tool Chain to EcoCoder

For EH2275A, the installation of HighTec TriCore tool chain is required.

After the installation of HighTec TriCore Tool Chain, users need to specify the makefile directory in EcoCoder Loader.

Open EcoCoder Loader, click Tools, then click ‘Select HighTec’.

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

For example, the full path could be: ‘C:\HIGHTEC\toolchains\tricore\v4.6.6.0-infineon-1.1\bin\make.exe’
2.5 Activate EcoCoder

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

*Note: Please close MATLAB for the activation process.*

1. Dongle

The hardware dongle released by Ecotrons could activate software once it is plugged in PC.

2. .dat file
‘.dat’ file is linked to Windows SN, meaning the ‘.dat’ file is bound to a specific PC and not allowed to be transferred to another PC. New ‘.dat’ file has to be issued if customer shift to new PC.

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

2.5.1 Get Key File

1. Double-click ‘EcoCoder Loader’ on the desktop.

2. Select Tools → Get EcoCoder Key.
3. Click ‘Export’.

4. Save the key file.
5. Please send the key file to EV-Support@ecotrons.com for license file.

2.5.2 Activate EcoCoder by License (.dat) File

1. Double-click ‘EcoCoder Loader’ at the Desktop.

2. Select Tools->Activate EcoCoder.
3. Click ‘Browse’.

5. Open license file, for example, select ‘EcoCoder_license.dat’, then select ‘Open’.
6. Click ‘OK’. The activation is successful if the pop-up window is displayed as following.

7. If following message shows up in MATLAB command window, EcoCoder is ready to use.
Chapter 3 Quick Start on Application Software

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

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

3. A model ‘DemoTest.mdl’, and a ‘.m’ file as shown in the following figure will be generated. In Simulink, by using shortcut key ‘Ctrl + B’ or click ‘Build Model’ button in Simulink task bar, ‘.a2l’, ‘.mot’, ‘.cal’ file would be generated.
If the ‘CodeWarrior’ window pops up, do not manually intervene, it will automatically compile and close when it is finished.

As shown in below picture, you can find the generated ‘mot’, ‘cal’, and ‘a2l’ files.
Chapter 4 EcoCoder Library

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

Folder: EcoCoder Blocks

Description:

Under the ‘General Parameters’ tab, this block defines the specific model of Ecotrons VCU hardware that you are using. Place this block in application model, usually at the top level, to select the VCU model for users’ application. The ‘Advanced Parameters’ tab enables the user to work with part of
the ROM and RAM memory addresses. If you would like to do the adjustment regarding all the addresses, please contact Ecotrons Tech Support, otherwise, please keep it as default.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Drop-down list</td>
<td>Pick target VCU</td>
</tr>
<tr>
<td>Enable all required</td>
<td>Check box</td>
<td>If enabled: All subsystems that are not assigned to tasking triggers would be assigned to L1ms trigger*</td>
</tr>
<tr>
<td>execution modules to be</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Parameters</td>
<td>Memory addresses</td>
<td>Contact our tech support for adjustments</td>
</tr>
</tbody>
</table>

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

4.2 ADC

4.2.1 Read ADC Value

Folder: EcoCoder Blocks/ADC

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

This EcoCoder block outputs values of the A/D converter channel connected to corresponded physical pin. The output value is a mapping of ratio of the voltage level of the analog source to the Reference Voltage (usually 5V) to a 10 or 12 bits binary value.

EcoCoder has predefined input voltage range and resolution of each channel, please refer to datasheet of the specific VCU.

See examples below:

<table>
<thead>
<tr>
<th>Channel</th>
<th>ADC Predefined Resolution (bits)</th>
<th>RAW ADC</th>
<th>Raw ADC (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>500</td>
<td>000111110100</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>500</td>
<td>00000111110100</td>
</tr>
</tbody>
</table>
**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog input channel</td>
<td>0 – n (determined by hardware resource of specific VCU)</td>
<td>Pick specific analog input channel</td>
</tr>
</tbody>
</table>
### 4.2.2 Read Fixed-Point ADC Volt

**Folder:** EcoCoder Blocks/ADC

**Description:**

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

**Block Parameters:**

1. *Analog input channel:* Choose analog channel.
2. *Input type:* Channel type of the voltage input, 4 types are supported: ‘0-5V’, ‘0-12V’, ‘0-24V’ and ‘Custom Voltage Ratio’.
3. *AD Resolution:* Please refer to the datasheet of VCU for resolution selection.
4. *Custom Voltage Ratio:* this option is available for fourth input type ‘Custom Voltage Ratio’ only. Previously, every voltage input type has fixed resistor divider, as a result, the user just need to select voltage type. Different resistor divider is introduced in new hardware, which explain the introduction of new input type and this input option.
5. *Reference Voltage:* By default, it will be set as 5V. Please do not change.
**Block Output:**

*Volt*: physical value of input voltage of specified channel; unit: V; ‘single’ data type.

(*) For fixed point toolbox advantages, refer to [https://www.mathworks.com/help/simulink/fixed-point.html](https://www.mathworks.com/help/simulink/fixed-point.html)

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

### 4.2.3 Read Float ADC Volt

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

**Block Parameters:**

1. *Analog input channel*: Choose analog channel.
2. *Input type*: Channel type of the voltage input, 4 types are supported: ‘0-5V’, ‘0-12V’, ‘0-24V’ and ‘Custom Voltage Ratio’.
3. **AD Resolution**: Please refer to the datasheet of VCU for resolution selection, since different VCU's have different AI channels setup.

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

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

**Block Output:**

*Volt*: physical value of input voltage of specified channel.

### 4.3 CAN Communication

Please Refer to Chapter 5, *CAN theory of Ecotrons* before using EcoCoder CAN blocks.

Chapter 5, combined with CAN bus communication protocol, will give the user preliminary knowledge of implementing CAN on Ecotrons VCU.

#### 4.3.1 CAN Channel Definition

**Folder**: EcoCoder Blocks/CAN

**Description:**
This block provides configuration interface for CAN lower level protocol parameters. It is recommended to read through and understand CAN lower level protocol prior to designing CAN related application software.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN_Channel</td>
<td>Drop-down list</td>
<td>Please refer to datasheet to select supported CAN channels. In some cases, CAN A is represented by CAN 0.</td>
</tr>
<tr>
<td>CAN_Enable</td>
<td>Check box</td>
<td>If checked: the selected CAN channel would be activated</td>
</tr>
</tbody>
</table>
### CAN ID Filter* Enable
- **Check box**
- **If checked:** message with ID in filter list would be filtered out by VCU on the selected CAN bus.

### CAN Extended
- **Check box**
- **If checked:** CAN ID input would be interpreted as extended format.

### CAN ID Mask (uint32 Hex)
- **Not Configurable**
- fixed at ‘7ff’, which means only IDs with lower 11 bits same as input of ‘CAN ID Filter’ will be accepted by VCU if ‘CAN ID Filter Enable’ checked.

### CAN ID Filter (uint32 Hex)
- **Numeric**
- Specify the filter

### CAN Baud Rate (bps)
- **Drop-down list**
- Specify baud rate

### CAN TxBuffer Size
- **Numeric**
- Input range: 1-30. It is used to specify software buffer size to help store the sequence of message to be sent.

### CAN RxBuffer Size
- **Numeric**
- Input range: 1-30. It is used to specify software buffer size to help store the sequence of incoming message.

*For principle of CAN bus filtering, please refer to some published literatures.*
4.3.2 CAN Wake-up Frame Definition

This block is used to define the VCU wake-up CAN message.

Parameters:

1. **CAN_Channel**: Selecting CAN channel number for this function.
2. **Wake-up Mode**: Setting the wake up mode, including Disable (disable CAN wake up function), All Frames (any frame on the specified bus can wake VCU up), and Specific Frames (User specify the frame that can wake up the VCU).
3. **Baud Rate**: CAN baud rate set up.
4. **ID Extended**: If checked, the specified VCU-waking-up message will have to use extended CAN ID. If not checked, the message have to use standard CAN ID.
5. **ID Setting**: Specify the ID here for the wake-up message.
6. **ID Mask**: The mask for VCU-waking-up message ID.

7. **Data Setting Enable**: If checked, not only a specific waking-up message ID is needed, but the user also need to specify the data in the message. Only message with matching ID and data can wake up the VCU. Data can be specified in the following blank.

8. **Data Length**: Set the wake-up message data length. Only when the data length of the wake-up message match this value, the message can wake up the VCU.

9. **Data Mask**: The mask for wake-up message data. The message data bitwise AND with this mask value, if one or more bit of the result of bitwise AND is (are) not 0, the message can wake up the VCU.

### 4.3.3 Read Fixed-Point CAN Message

**Folder**: EcoCoder Blocks/CAN

**Description:**

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

### Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select CAN Channel</td>
<td>Drop-down list</td>
<td>The CAN channel has to be defined before applied.</td>
</tr>
<tr>
<td>Select M File</td>
<td>Check box</td>
<td>If checked: please enter the name of m file in the blank space under check box.</td>
</tr>
<tr>
<td>Select Message</td>
<td>Drop-down list</td>
<td>Specify CAN message to be received and processed by the block.</td>
</tr>
<tr>
<td>Show Message Available Port</td>
<td>Check box</td>
<td>If checked: the block will provide a signal flag to help tell the availability of this CAN message.</td>
</tr>
<tr>
<td>Show Message Count Port</td>
<td>Check box</td>
<td>Message counter, if checked: every time the message is received, the counter increments by 1.</td>
</tr>
<tr>
<td>Show Signal Name</td>
<td>Check box</td>
<td>If checked: the names of the signals will be displayed.</td>
</tr>
<tr>
<td>Define Signal</td>
<td>Check box</td>
<td>If checked: signals parsed out from the block will be cast as measurement variables. ‘Show Signal Name’ must be checked before checking this item.</td>
</tr>
</tbody>
</table>
### Signal prefix
- **Alpha-numeric text**: Specify prefix to parsed out signal names, remember to use single quote.

### Output Data Type: Inherit
- **Check box**: If checked: the data type of the signal is inherited from input data type. If not checked: the signal type is automatically defined using fixed point tool.

### Sample time
- **Drop-down list**: Please refer to section 5.2.5

## 4.3.4 Send Fixed-Point CAN Message

**Folder: EcoCoder Blocks/CAN**

**Description:**

![Send Fixed-Point CAN Message Block Parameters](image-url)
<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select CAN Channel State</strong></td>
<td>Drop-down list</td>
<td>Connected: Message will be sent out from the CAN channel selected under ‘Select CAN Channel’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Connected: Message will be sent out from the CAN channel selected under ‘Select CAN Channel’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disconnected: User will have to assign the output port manually.</td>
</tr>
<tr>
<td><strong>Select CAN Channel</strong></td>
<td>Drop-down list</td>
<td>CAN channel selection</td>
</tr>
<tr>
<td><strong>Select M File</strong></td>
<td>Check box</td>
<td>If checked: please enter the name of m file in the blank space under check box.</td>
</tr>
<tr>
<td><strong>Select Message</strong></td>
<td>Drop-down list</td>
<td>Specify CAN message to be sent and processed by the block.</td>
</tr>
<tr>
<td><strong>Input Data Type: Inherit</strong></td>
<td>Check box</td>
<td>If checked: the data type of the signal is inherited from input data type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If not checked: the signal type is automatically defined using fixed point tool.</td>
</tr>
<tr>
<td><strong>Sample time</strong></td>
<td>Drop-down list</td>
<td>Please refer to section 5.2.5</td>
</tr>
</tbody>
</table>

**Block Inputs:**

Signals to be sent out.

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

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

4.3.5 Read/Send CAN Message

These two blocks are similar to previous two blocks which is generic version of CAN read/send for customer who did not purchase fixed point toolbox of Simulink. However, if the toolbox available, it is recommended to use fixed point version of read/send blocks.

4.3.6 CAN Receive Counter

Folder: EcoCoder Blocks/CAN

Description:

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

Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN channel</td>
<td>Drop-down list</td>
<td>Specify the channel to be</td>
</tr>
</tbody>
</table>
Block Output:

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

### 4.3.7 Set CAN Mode

Folder: EcoCoder Blocks/CAN

**Description:**

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger type</td>
<td>Drop-down list</td>
<td>Trigger type selection</td>
</tr>
<tr>
<td>CAN Channel</td>
<td>Drop-down list</td>
<td>Specify CAN channel to be controlled</td>
</tr>
<tr>
<td>CAN Mode</td>
<td>Drop-down list</td>
<td>Specify the CAN mode to be triggered by the block</td>
</tr>
</tbody>
</table>

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

4.3.8 Send CAN Data

![Send CAN Data Block](image)

**Block Parameters:**

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

**Block Inputs:**

- data: The message data to be sent.
4.3.9 Unpack Signals to CAN Data

**Block Parameter:**

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

**Block Inputs:**

Data: the message data to be unpacked.

**Block Outputs:**

Unpacked signals from the CAN data.
### 4.3.10 Pack Signals To CAN Data

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

#### Block Parameters:

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

#### Block Inputs:

Signals to be packed, values are in HEX.

#### Block Output:

Data: the packed CAN message data.
4.3.11 Receive CAN Message

Block Parameters:

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

Block Outputs:

2. Remote: Flag for frame type, 1 stands for remote frame. 0 stands for data frame.
3. Extended: Flag for frame type, 1 stands for extended frame. 0 stands for standard frame.
4. ID: Message ID.
5. Length: Message data length.
6. Data: Message data.
4.3.12 Transmit CAN Message

Block Parameters:

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

Block Inputs:

1. Remote: Flag for frame type, 1 stands for remote frame. 0 stands for data frame.
2. Extended: Flag for frame type, 1 stands for extended frame. 0 stands for standard frame.
3. ID: Message ID.
4. Length: Message data length.
5. Data: Message data.

4.4 Serial Communication Interface (SCI) Block

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

Block Parameters:

1. SCI Channel: Communication channel selection.
2. SCI_Enable: Enable selected channel.
3. SCI Baudrate: Channel baudrate setup.
4. SCI Parity Mode: Parity check mode setup.

4.4.2 Read SCI Data

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

Block Parameter:

SCI_Channel: SCI communication channel selection.

Block Outputs:

1. f(): Flag for receiving data. If data received, the flag will be 1. This signal could
be used as a trigger signal.

2. Data: Output received 8-bit data.

### 4.4.3 Send SCI Data

This block will send SCI data to assigned channel.

![Send SCI Data Block](image)

**Block Parameter:**

SCI_Channel: SCI channel selection.

**Block Input:**

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

### 4.5 Digital I/O

This block set is used for measuring digital input/output. Including Switch Input, Switch Output, PWM input and PWM output.

#### 4.5.1 Switch Input

**Folder:** EcoCoder Blocks/Digital I/O

**Description:**
This block reads the physical voltage level of digital input channels and output Boolean value to application layer.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch input channel</td>
<td>Drop-down list</td>
<td>Digital input channel selection</td>
</tr>
</tbody>
</table>

**4.5.2 KeyOn Input**

**Folder:** EcoCoder Blocks/Digital I/O

**Description:**
KeyON signal is recommended to be used for power up and shutting down the VCU. For different VCUs, KeyOn signal inputs are different (refer to the VCU datasheet to confirm the KeyOn signal input type) - if KeyOn signal is digital input, leave the configuration as default; If KeyOn signal is read through analog input channel, user will have to configure factor according to voltage divider parameter - for this parameter, please refer to VCU datasheet.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch input channel</td>
<td>Drop-down list</td>
<td>Only one channel selectable for KeyOn</td>
</tr>
<tr>
<td>Key AD2Volt Factor</td>
<td>Numeric</td>
<td>The voltage factor for KeyOn voltage detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(only valid when KeyOn is read from AI)</td>
</tr>
<tr>
<td>Key Off Threshold Volt</td>
<td>Numeric</td>
<td>If the input voltage is lower than this value, output is ‘0’.</td>
</tr>
<tr>
<td>Key On Hysteresis Volt</td>
<td>Numeric</td>
<td>The hysteresis value</td>
</tr>
</tbody>
</table>
between upper and lower threshold.
If the interpreted voltage is larger than the sum of ‘Key Off Threshold Volt’ value and this value, output is ‘1’.

4.5.3 Switch Output

Folder: EcoCoder Blocks/Digital I/O

Description:

Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch output channel</td>
<td>Drop-down list</td>
<td>Select switch channels to be controlled</td>
</tr>
<tr>
<td>Input</td>
<td>Numeric (bool)</td>
<td>0 or 1, switch control value</td>
</tr>
</tbody>
</table>
4.5.4 IPM Read

Folder: EcoCoder Blocks/Digital I/O

Description:

![Source Block Parameters: IPM Read](image)

This block measure the frequency of input PWM signal and returns the PWM signal period.

### Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPWM channel</td>
<td>Drop-down list</td>
<td>Select channel to measure PWM input</td>
</tr>
<tr>
<td>Period (output)</td>
<td>Numeric</td>
<td>PWM period, Unit is 0.1ms</td>
</tr>
</tbody>
</table>

4.5.5 PWM Definition

Folder: EcoCoder Blocks/Digital I/O

Description:
This block enables channels for PWM output, initializes the PWM output parameters for corresponding channels.

Channels with PWM output capability (H-bridge, LSO, HSO) could be found in the pinout table of VCU datasheet.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPWM channel</td>
<td>Drop-down list</td>
<td>Specify the channel for PWM output</td>
</tr>
<tr>
<td>OPWM Enable</td>
<td>Check box</td>
<td>If checked, enable PWM output function of specified channel</td>
</tr>
<tr>
<td>OPWM FRQ</td>
<td>Numeric</td>
<td>Recommended frequency range for perfect square wave output is 15 – 2000 Hz.</td>
</tr>
<tr>
<td>OPWM Duty</td>
<td>Numeric</td>
<td>The unit for input value is configurable in the block \textit{PWM IO Frequency Range Definition}. Control the duty cycle of the selected channel signal. Expected value is 0-10000, corresponding to 0-100%.</td>
</tr>
</tbody>
</table>

4.5.6 PWM Output

Folder: EcoCoder Blocks/Digital I/O

Description:

This block configures PWM outputs.
### Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPWM channel</td>
<td>Drop-down list</td>
<td>Specify the PWM output channel</td>
</tr>
<tr>
<td>freq (input)</td>
<td>Numeric</td>
<td>Recommended range for perfect square wave output is 15 – 2000 Hz.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For the input value unit, refer to PWM IO Frequency Range Definition.</td>
</tr>
<tr>
<td>duty (input)</td>
<td>Numeric</td>
<td>Control the duty cycle of the selected channel output, value 0-10000 corresponds to 0-100%.</td>
</tr>
</tbody>
</table>

#### 4.5.7 WakeUp Input

**Folder: EcoCoder Blocks/Digital I/O**

**Description:**

![S-Function mask and switch input channel](source-block-parameters-wakeup-input.png)
The block can read wake-up signal status.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch input channel</td>
<td>Drop-down list</td>
<td>Wakeup source selection</td>
</tr>
<tr>
<td>Output</td>
<td>Numeric (boolean)</td>
<td>‘1’ is active</td>
</tr>
</tbody>
</table>

### 4.5.8 H-bridge Definition

**Folder: EcoCoder Blocks/Digital I/O**

**Description:**

![H-bridge Definition](image)

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

**Block Parameters**
<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-bridge Channel</td>
<td>Drop-down list</td>
<td>Select H-bridge channel</td>
</tr>
<tr>
<td>H-bridge Enable</td>
<td>Check box</td>
<td>If checked: Enable H-bridge</td>
</tr>
<tr>
<td>H-bridge Current Direction A&gt;B</td>
<td>Check box</td>
<td>If checked: The current direction is from A&gt;B. If not checked: the current direction is B&gt;A. (A and B are the two outputs of H-bridge, see the VCU data sheet for more information)</td>
</tr>
<tr>
<td>H-bridge Initial Frequency</td>
<td>Numeric (Hz)</td>
<td>The theoretical range is 1-2000000 Hz. Recommended range for perfect square wave output is 15 – 2000 Hz. Input value unit is configurable in the PWM IO Frequency Range Definition block.</td>
</tr>
<tr>
<td>H-bridge Initial Duty Cycle</td>
<td>Numeric</td>
<td>0-10000 corresponds to 0-100%.</td>
</tr>
</tbody>
</table>
4.5.9 H-bridge Output

Folder: EcoCoder Blocks/Digital I/O

Description:

The block controls H-bridge output.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-bridge Channel</td>
<td>Drop-down list</td>
<td>Select H-bridge channel</td>
</tr>
<tr>
<td>en (input)</td>
<td>Numeric (bool)</td>
<td>‘1’ to enable H-bridge</td>
</tr>
<tr>
<td>curA2B (input)</td>
<td>Numeric (bool)</td>
<td>‘1’ : current flows from A to B; ‘0’ : currents flows from B to A.</td>
</tr>
<tr>
<td>frq (input)</td>
<td>Numeric</td>
<td>The theoretical range is 1-2000000 Hz. Recommended range for perfect square wave output is 15 – 2000 Hz. Input value unit is configurable in the PWM IO</td>
</tr>
</tbody>
</table>
Frequency Range Definition

| Duty (Input) | Numeric | 0-10000 corresponds to 0-100%. |

### 4.5.10 PWM IO Frequency Range Definition

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

**Block Parameters:**

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

2. Frequency Accuracy: Accuracy adjustment. Value in the box means the unit frequency for outputs/inputs of frequency related blocks. For example, if the Frequency Accuracy is 0.01 Hz, it means that when frequency related block outputs/inputs value is 5, the actual physical frequency value is $5 \times 0.01 \text{Hz} = 0.05 \text{Hz}$. 
4.5.11 IPWM Read

Block Parameter:
Select IPWM Channel: PWM inputs channel selection

Block Outputs:
1. freq: the input PWM frequency of the signal
2. duty: the input PWM signal duty cycle

4.6 Task Scheduler

4.6.1 Task Trigger

Folder: EcoCoder Blocks/Task Scheduler
Description:
This block is for task scheduling and prioritization.

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

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Task Time</td>
<td>Drop-down list</td>
<td>Task type and execution period selection*</td>
</tr>
</tbody>
</table>

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

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

* For CAN bus applications, it is recommended to run in 10ms tasks.
4.7 Non-Volatile Memory Blocks

Non-Volatile Memory (NVM) could be used to store critical data that is supposed not to be lost, or altered during VCUs’ service life, such as vehicle odometer values. NVM variables are stored in specified memory blocks and will be loaded to RAM when VCUs start running, and written back to memory blocks during VCUs power-off process.

For more information, please refer to Chapter 6 for VCU memory management. Chapter 6, combined with Appendix A, will give user preliminary knowledge of implementing NVM on Ecotrons VCU.

4.7.1 Fixed NVM Definition

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:

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

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

Block Parameters
Parameter Field | Value | Comments/Description
---|---|---
Select the m file | .m file | Select the .m file defining NVM variables.

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

```matlab
function NVMList=ecoder_getFixedNvmList()
NVMList=[...]
    struct ('name', {'From_double'}, 'type', {'double'}, 'size', 1, 'init', 0, 'value', 1), ...
    struct ('name', {'From_int8'}, 'type', {'int8'}, 'size', 2, 'init', 1, 'value', [1 2]), ...
    struct ('name', {'From_uint8'}, 'type', {'uint8'}, 'size', 1, 'init', 0, 'value', [1 2]), ...
    struct ('name', {'From_boolean'}, 'type', {'boolean'}, 'size', 4, 'init', 0, 'value', [1 2 3 4]), ...
    struct ('name', {'From_uint16'}, 'type', {'uint16'}, 'size', 2, 'init', 1, 'value', [1 2]), ...
    struct ('name', {'From_uint16'}, 'type', {'uint16'}, 'size', 2, 'init', 1, 'value', [1 2]), ...
    struct ('name', {'From_uint32'}, 'type', {'uint32'}, 'size', 4, 'init', 0, 'value', [1 2 3 4]), ...
];
end
```

The .m file needs to be added under MATLAB path. The ‘init’ in the .m file is the flag for NVM variable initialization.

init = 1: The corresponding NVM variable value(s) will be loaded from .mot file during the first time of VCU starting process, every time after .mot file being flashed into VCU.

init = 0: The corresponding NVM variable value(s) will be loaded from original VCU NVM memory block during the first time of VCU starting process every time after .mot file being flashed.

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

### 4.7.2 Read Fixed NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks
Description:

This module is used for reading fixed NVM variables.

Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed NVM Variable</td>
<td>Variable name</td>
<td>Specify the variable name to be read</td>
</tr>
</tbody>
</table>

Block Output:
The value of corresponding variables.

4.7.3 Write Fixed NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:
This module is used for writing fixed NVM variables.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed NVM Variable</td>
<td>Variable name</td>
<td>Specify the variable to be written.</td>
</tr>
</tbody>
</table>

**4.7.4 NVM Definition**

**Folder: EcoCoder Blocks/Non-volatile Memory Blocks**

**Description:**

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

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load NVM value in the following way</td>
<td>Drop-down list</td>
<td><strong>Load previous saved values in flash:</strong> The corresponding NVM variable value would be initialized from the NVM memory area, instead of .mot file.</td>
</tr>
</tbody>
</table>
4.7.5 NVM Variable Definition

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:

![NVM Variable Definition Block](image)

This block is used to define regular NVM variables.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable name</td>
<td>Alpha-numeric text</td>
<td>Variable name</td>
</tr>
<tr>
<td>Initial Value</td>
<td>Numeric</td>
<td>Initial value of the to-be defined variable</td>
</tr>
</tbody>
</table>
### 4.7.6 Read NVM

**Folder: EcoCoder Blocks/Non-volatile Memory Blocks**

**Description:**

This module is used for reading regular NVM variables.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable_name</td>
<td>Alpha-numeric text</td>
<td>Specify variable name</td>
</tr>
<tr>
<td>Variable_type</td>
<td>Drop-down list</td>
<td>Variable data type</td>
</tr>
</tbody>
</table>

**Block Output:**

The NVM variable value
4.7.7 Write NVM

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:

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

Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Name</td>
<td>Alpha-numeric text</td>
<td>NVM variable name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable Variable Definition</td>
<td>Check box</td>
<td>If checked:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Define and write NVM variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If not checked:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only write NVM</td>
</tr>
<tr>
<td>Initial Value</td>
<td>Numeric</td>
<td>NVM variable initial value (for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### NVM variable definition

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Type</td>
<td>Drop-down list</td>
<td>NVM variable type</td>
</tr>
<tr>
<td>Dimension</td>
<td>Numeric</td>
<td>Dimension of NVM variable</td>
</tr>
<tr>
<td>Unit</td>
<td>Alpha-numeric text</td>
<td>User-defined variable unit</td>
</tr>
<tr>
<td>Description</td>
<td>Alpha-numeric text</td>
<td>Memo</td>
</tr>
</tbody>
</table>

### 4.7.8 Store All NVM Data

**Folder:** EcoCoder Blocks/Non-volatile Memory Blocks

**Description:**

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

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

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
</table>
4.7.9 Restore All NVM Data

Folder: EcoCoder Blocks/Non-volatile Memory Blocks

Description:

The module reads NVM data from ROM (flash) back to RAM.
It is recommended to call this block when VCU powers on. This block could be triggered by ‘Task_ini’.

Block Parameter:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Trigger Type</td>
<td>Drop-down list</td>
<td>Select trigger type</td>
</tr>
</tbody>
</table>

4.8 Diagnostic Blocks

Diagnostic blocks are designed to realize VCU diagnostic functions.
4.8.1 Hardware Output DTC

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:

This block can realize the hardware diagnosis of supported LSO, HSO and H-bridge. Please refer to VCU datasheet for the channels that support diagnostic functions.

Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTC_Channel</td>
<td>Drop-down list</td>
<td>Select hardware channel</td>
</tr>
</tbody>
</table>

Block Output:

DTC: The diagnostic trouble code of the specified channel.

4.8.2 DTC Parser

Folder: EcoCoder Blocks/Diagnostic Blocks

Description:
This block could help parse out specific fault of DTC.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index of bit</td>
<td>Drop-down list</td>
<td>Select the fault to be analyzed</td>
</tr>
</tbody>
</table>

**Block Input:**

DTC: The diagnostic trouble code.

**Block Output:**

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

### 4.8.3 Software Core Diagnostic

**Folder:** EcoCoder Blocks/Diagnostic Blocks

**Description:**
This block provides memory/chip fault check.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTC_Channel</td>
<td>Drop-down list</td>
<td>Select the memory or chip to be diagnosed.</td>
</tr>
</tbody>
</table>

### 4.8.4 Clear H-bridge DTC

**Folder:** EcoCoder Blocks/Diagnostic Blocks

**Description:**

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

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
</table>
4.9 Calibration & Measurement

4.9.1 Calibration Definition

Folder: EcoCoder Blocks/Calibration & Measurement

Description:

This block could help define and initialize calibration variable.

Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>Alpha-numeric text</td>
<td>Calibration variable name</td>
</tr>
</tbody>
</table>
4.9.2 Read Calibration

Folder: EcoCoder Blocks/Calibration & Measurement

Description:

This block defines and reads calibration variables.

Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable_name</td>
<td>Alpha-numeric text</td>
<td>Calibration variable name</td>
</tr>
<tr>
<td>variable_value</td>
<td>Numeric</td>
<td>Calibration variable initial value</td>
</tr>
</tbody>
</table>
variable_type | Drop-down list | Calibration variable data type
---|---|---
variable_min | Numeric | Calibration variable lower limit
variable_max | Numeric | Calibration variable upper limit
Unit | Alpha-numeric text | User-defined calibration variable unit
Description | Alpha-numeric text | Memo

**Block Output:**

Calibration variable value.

### 4.9.3 Write Measurement

**Folder:** EcoCoder Blocks/Calibration & Measurement

**Description**

![Write Measurement Block](image)

This block could help define measurement variables.

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>demo_trq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>variable_type</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>Dimensions</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Block Input:

To-be-measured variable.

### 4.9.4 Write and Read Measurement

**Folder:** EcoCoder Blocks/Calibration & Measurement

**Description:**

![Function Block Parameters: Write and Read Measurement](image)

This block is an inline block, it helps read measurement variables.
Parameter Field | Value | Comments/Description
--- | --- | ---
variable_name | Alpha-numeric text | Measurement variable name
variable_type | Drop-down list | Variable data type
Dimensions | Numeric | The dimension of measurement variable
Unit | Alpha-numeric text | User-defined measurement variable unit
Description | Alpha-numeric text | Memo

**Block Input:**
To-be-measured variable.

**Block Output:**
Same as input. (This block is used for variable measurements, does not change variable values).

### 4.9.5 Override Probe

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

**Block Parameters:**

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

**Block Input:**

Variable to be overridden.

**Output:**

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

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

### 4.9.6 1-D Lookup Table

**Folder:** EcoCoder Blocks/Calibration & Measurement

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Alpha-numeric text</td>
<td>Variable name</td>
</tr>
<tr>
<td>Breakpoint type</td>
<td>Drop-down list</td>
<td>Variable type</td>
</tr>
<tr>
<td>Breakpoint data</td>
<td>Numeric (Matrix)</td>
<td>Breakpoint data</td>
</tr>
<tr>
<td>Table type</td>
<td>Drop-down list</td>
<td>Table variable type</td>
</tr>
<tr>
<td>Table data</td>
<td>Numeric (Matrix)</td>
<td>Table data</td>
</tr>
</tbody>
</table>

**4.9.7 2-D Lookup Table**

Folder: EcoCoder Blocks/Calibration & Measurement
Description:

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

Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Alpha-numeric text</td>
<td>2-D look-up table name</td>
</tr>
<tr>
<td>Breakpoints 1 type</td>
<td>Drop-down list</td>
<td>Breakpoints 1 variable data type</td>
</tr>
<tr>
<td>Breakpoints 1 data(Row)</td>
<td>Numeric (Matrix)</td>
<td>Breakpoints 1 variable data</td>
</tr>
<tr>
<td>Breakpoints 2 type</td>
<td>Drop-down list</td>
<td>Breakpoints 2 variable data type</td>
</tr>
<tr>
<td>Breakpoints 2 data(Column)</td>
<td>Numeric (Matrix)</td>
<td>Breakpoints 2 variable data</td>
</tr>
<tr>
<td>Table type</td>
<td>Drop-down list</td>
<td>Select table variable data type</td>
</tr>
<tr>
<td>Table data</td>
<td>Numeric (Matrix)</td>
<td>Initialize table data</td>
</tr>
</tbody>
</table>
4.9.8 Calibration Data Check

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

Block Parameters:

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

Output:

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

4.10 System Management Blocks

4.10.1 Power Management Example

Folder: EcoCoder Blocks/System Management Blocks

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Signal From Input</td>
<td>Check box</td>
<td>If checked, the key switch signal can be read from VCU input.</td>
</tr>
<tr>
<td>Power Off Delay(S)</td>
<td>Numeric</td>
<td>Power off delay time</td>
</tr>
<tr>
<td>Key Cycle Reset Enable</td>
<td>Numeric (Boolean)</td>
<td>If set to 0, VCU will not be reset if key switch turns back on before VCU power-off delay period ends.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Key Cycle Off Duration(S)</td>
<td>Numeric</td>
<td>If set to 0, VCU will be reset if key switch turns back on before VCU power-off delay ends.</td>
</tr>
<tr>
<td>Key Cycle On Duration(S)</td>
<td>Numeric</td>
<td>The duration (time threshold) after the key-off moment (KeyOn signal absent) to the time when VCU starts power-off process.</td>
</tr>
<tr>
<td>Key AD2Volt Factor</td>
<td>Numeric</td>
<td>VCU starts power-up process if KeyOn signal is detected for more than this time threshold.</td>
</tr>
<tr>
<td>Key Off Threshold Volt(V)</td>
<td>Numeric</td>
<td>The factor to be multiplied that convert AD to voltage, see section 4.2.1 for details.</td>
</tr>
<tr>
<td>Key Off Threshold Volt(V)</td>
<td>Numeric</td>
<td>If the input KeyOn voltage is less than this value, KeyOn signal is interpreted as ‘0’.</td>
</tr>
<tr>
<td>Key On Hysteresis Volt(V)</td>
<td>Numeric</td>
<td>If the KeyOn input voltage is larger than the sum of ‘Key Off Threshold Volt’ and this hysteresis value, KeyOn signal is ‘1’.</td>
</tr>
<tr>
<td>Power Off Fail Enable Reset</td>
<td>Numeric (Boolean)</td>
<td>If this setting is ‘1’, VCU would keep trying to power off at certain frequency when power-down process fails.</td>
</tr>
<tr>
<td>Enable NVM Store And Restore</td>
<td>Check box</td>
<td>If Checked: Enable NVM control option.</td>
</tr>
</tbody>
</table>
Set The Waiting Time(ms) | Numeric | Power-off delay time

### 4.10.2 Shutdown Power

**Folder: EcoCoder Blocks/System Management Blocks**

**Description:**

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Trigger Type</td>
<td>Drop-down list</td>
<td>Block trigger signal type selection.</td>
</tr>
<tr>
<td>Set The Waiting Time(ms)</td>
<td>Numeric</td>
<td>Set the time of power-off delay waiting time.</td>
</tr>
</tbody>
</table>
4.10.3 Set ECU Mode

Folder: EcoCoder Blocks/System Management Blocks

Description:

This module can set the working mode of ECU.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Type</td>
<td>Drop-down list</td>
<td>Block trigger signal type selection.</td>
</tr>
<tr>
<td>Select ECU Mode</td>
<td>Drop-down list</td>
<td>Work mode selection</td>
</tr>
</tbody>
</table>

4.10.4 ECU Master Chip Wake-Up Definition

Folder: EcoCoder Blocks/System Management Blocks

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

### Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Type</td>
<td>Drop-down list</td>
<td>Wake-up CAN channel selection</td>
</tr>
<tr>
<td>Wakeup Enable</td>
<td>Check box</td>
<td>If checked: the specified CAN channel can wake up VCU.</td>
</tr>
</tbody>
</table>

### 4.10.5 Watchdog Definition

**Folder:** EcoCoder Blocks/System Management Blocks

**Description:**
Settings for software watchdog and hardware watchdog.

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Watchdog Enable</td>
<td>Check box</td>
<td>If checked: Software watchdog is enabled.</td>
</tr>
<tr>
<td>Software Watchdog Wait Time(ms)</td>
<td>Numeric</td>
<td>The ‘feeding dog’ operation is executed at Task_L1ms, software will reset when timeout.</td>
</tr>
<tr>
<td>Hardware Watchdog Enable</td>
<td>Check box</td>
<td>If checked: Hardware watchdog enabled. (If this icon is greyed out, the specified VCU has no hardware watchdog built in)</td>
</tr>
</tbody>
</table>
4.10.6 Software Reset

Folder: EcoCoder Blocks/System Management Blocks

Description:

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

Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Trigger Type</td>
<td>Drop-down list</td>
<td>Block trigger signal type selection.</td>
</tr>
</tbody>
</table>

4.10.7 Read System Free Counter

Folder: EcoCoder Blocks/System Management Blocks

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

**Block Output:**

System free counter value.

### 4.10.8 Power Control Output

**Folder:** EcoCoder Blocks/System Management Blocks

**Description:**

![Power Control Output Block](image)

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Control Channel</td>
<td>Drop-down list</td>
<td>Power channel selection.</td>
</tr>
<tr>
<td>input</td>
<td>Boolean</td>
<td>1: turning on power for corresponding channel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: turning off power for corresponding channel.</td>
</tr>
</tbody>
</table>
4.10.9 Service Software Watchdog

Folder: EcoCoder Blocks/System Management Blocks

Description:

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

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

4.11 CCP

This block set includes CCP related implementations.

4.11.1 Fixed CCP Slave Definition

Folder: EcoCoder Blocks/CCP

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

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP Type</td>
<td>Drop-down list</td>
<td>Simple: Under this option, the only configurable parameter in this block is CAN channel. After building, users will get A2L, CAL and Mot (or Hex) file. To use EcoCAL, both A2L and CAL files need to be loaded. Configurable: Under this option, after building, users will get A2L and Mot (or Hex) file. To use EcoCAL, A2L and Mot (or Hex) file.</td>
</tr>
</tbody>
</table>
## EcoCoder Manual V4.8.2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP Enable</td>
<td>Check box</td>
<td>CCP enable</td>
</tr>
<tr>
<td>CAN Channel</td>
<td>Drop-down list</td>
<td>Specify CAN channel for CCP</td>
</tr>
<tr>
<td>Station Address (Intel)</td>
<td>Numeric</td>
<td>CCP Station Address</td>
</tr>
<tr>
<td>Command Receive Object (CRO) ID</td>
<td>Numeric</td>
<td>Specify Command Receive Object (CRO) ID (Master-&gt;Slave)</td>
</tr>
<tr>
<td>Data Transmit Object (DTO) ID</td>
<td>Numeric</td>
<td>Data Transmit Object (DTO) ID (Slave -&gt; Master)</td>
</tr>
<tr>
<td>Data Acquisition (DAQ) ID</td>
<td>Numeric</td>
<td>CCP DAQ ID</td>
</tr>
<tr>
<td>Data Acquisition (DAQ) PID</td>
<td>Numeric</td>
<td>The first PID in the DAQ list.</td>
</tr>
<tr>
<td>Data Acquisition (DAQ) Length</td>
<td>Numeric</td>
<td>DAQ list length</td>
</tr>
<tr>
<td>Data Acquisition (DAQ) Period(ms)</td>
<td>Numeric</td>
<td>CCP DAQ period</td>
</tr>
</tbody>
</table>

### 4.12 Programming Blocks

#### 4.12.1 Online Programming Definition

**Folder:** EcoCoder Blocks/CCP

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

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

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select CAN Channel</td>
<td>Drop-down list</td>
<td>Online programming CAN channel selection.</td>
</tr>
<tr>
<td>Enable KeyCycle Program</td>
<td>Check box</td>
<td>If checked: programming will require key cycle to start programming.</td>
</tr>
<tr>
<td>Enable Online Program</td>
<td>Check box</td>
<td>If checked: VCU programming will not require a key cycle.</td>
</tr>
<tr>
<td>Station Address (Intel)</td>
<td>Greyed out</td>
<td>This value cannot be changed for now.</td>
</tr>
</tbody>
</table>
### Command Receive Object (CRO) ID (11-bit)

Specify Command Receive Object (CRO) ID (Master->Slave)

It is recommended to remain as default.

### Data Transmit Object (DTO) ID (11-bit)

Data Transmit Object (DTO) ID (Slave -> Master)

It is recommended to remain as default.

---

### 4.13 Sensors Blocks

#### 4.13.1 Read Gyro Hex Value

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

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

#### 4.13.2 Read Gyro Phy Value

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

OTP  One-Time Programmable memory

4.14.1 Read OTP

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Position Offset</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Data Length</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Start Position Offset (Byte)</td>
<td>Numeric</td>
<td>Start address offset value</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Data Length (Byte)</td>
<td>Numeric</td>
<td>Data length (the number of bytes the data takes)</td>
</tr>
</tbody>
</table>

**Block Outputs:**

data: The data read from OTP area.

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

### 4.14.2 Read OTP (Input port)

![Diagram of Read OTP (Input port)](image)

**Function Block Parameters: Read OTP (Input port)**

- **Read OTP (mask) (link)**
- **Parameters**
  - **Data Length (Byte)**
    - 30

[Buttons: OK, Cancel, Help, Apply]
Block Parameters:

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Length (Byte)</td>
<td>Numeric</td>
<td>The number of bytes that the data takes in OTP area.</td>
</tr>
</tbody>
</table>

Block Input:
Offset (byte): the start position offset.

Block Output:
data: The data that has been read from OTP area.
st: Data reading status, 0 stands for data reading successfully.

4.14.3 Write OTP

![Write OTP Diagram]
Block Parameters:

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

![Diagram of Write OTP block](image)

**Function Block Parameters: Write OTP(Input port)**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Trigger Type</td>
<td>rising</td>
<td>Block trigger type selection.</td>
</tr>
<tr>
<td>Data Length (Byte)</td>
<td>30</td>
<td>The data length in byte.</td>
</tr>
</tbody>
</table>

**Block Parameters:**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Block Input:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start Position Offset (8Bytes)</td>
<td>Numeric</td>
<td>OTP writing start address offset, the unit is 8 bytes.</td>
</tr>
<tr>
<td>Select Trigger Type</td>
<td>Drop-down list</td>
<td></td>
</tr>
<tr>
<td>Data Length (Byte)</td>
<td>Numeric</td>
<td>The data length in byte.</td>
</tr>
<tr>
<td><strong>Block Input:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td>Data input</td>
</tr>
<tr>
<td><strong>Block Output:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>st</td>
<td>boolean</td>
<td>Data write status, 0 stands for writing successful; Non-0 value stands for writing</td>
</tr>
</tbody>
</table>
4.14.5 Read Data by Address

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

![Image of Read Data by Address block]

### Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start address</td>
<td>Numeric</td>
<td>Start address</td>
</tr>
<tr>
<td>Data Length (Byte)</td>
<td>Numeric</td>
<td>Data length</td>
</tr>
<tr>
<td><strong>Block Output:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td>Data output</td>
</tr>
</tbody>
</table>
4.14.6 Read Data by Address (Input port)

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

![Read Data By Address (Input port)](image)

**Block Parameters**

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start address (input)</td>
<td>Numeric</td>
<td>Start address</td>
</tr>
<tr>
<td>Data Length (Byte)</td>
<td>Numeric</td>
<td>Data read length</td>
</tr>
<tr>
<td><strong>Block Output:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td>Data output</td>
</tr>
</tbody>
</table>

4.14.7 Read String Value

This block could translate strings to corresponding ASCII numeric arrays.
## Block Parameters

<table>
<thead>
<tr>
<th>Parameter Field</th>
<th>Value</th>
<th>Comments/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Input</td>
<td>String</td>
<td>The string to be parsed</td>
</tr>
<tr>
<td>Data</td>
<td>Numeric</td>
<td>The numeric arrays of the string ASCII.</td>
</tr>
</tbody>
</table>
Chapter 5  CAN Theory of Ecotrons

5.1  Introduction

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

5.2  CAN Implementation

To use Ecotrons VCUss, the user need to convert DBC file into .m file and then use the .m file to define and initialize the CAN communications. The process is intuitive, user-friendly, and could give users more flexibility for CAN communication implementation.
5.2.1 Convert DBC to m File

User can convert DBC to .m file automatically using the software EcoCAN that could 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.

![Image showing a window with network nodes and a table of addresses and comments.]

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

![Image showing a file being saved with options for name, date modified, type, and size. A file named 'EV_Demo.m' is shown with a save button highlighted.]
5.2.2 EcoCoder CAN Blocks

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

5.2.3 Select m file

This m file could help parse out signals in messages. Users need to save the .m file under Matlab path.
5.2.4 Select Message

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

![Image of Source Block Parameters]

**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).
Interval \((t, -1)\) \((0,0.005)\) \([0.005,0.01)\) \([0.01,0.02)\) \([0.02,0.05)\) 
<table>
<thead>
<tr>
<th>Sample Time</th>
<th>Task_Inherit</th>
<th>Task_H1ms</th>
<th>Task_H5ms</th>
<th>Task_H10ms</th>
<th>Task_L20ms</th>
</tr>
</thead>
</table>
Interval \([0.05,0.1)\) \([0.1,0.2)\) \([0.2,0.5)\) \([0.5,1)\) \([1,10)\) 
| Sample Time        | Task_L50ms   | Task_L100ms| Task_L200ms| Task_L500ms| Task_L1000ms|
Chapter 6  Memory Management

6.1  Introduction

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

6.2  Storage device

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

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

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

6.3  Data Storage

6.3.1  Calibration/Measurement Variable

Please refer to section 4.9.1 to 4.9.4 for definition, initialization, reading and writing calibration and measurement variables.

The only special part is writing of calibration variable. It is achieved through EcoCAL ‘burn to’ function. EcoCAL is an advanced calibration tool developed by Ecotrons. ‘Save’ option could save calibration data to RAM, while ‘Burn to’ option could write calibration data to VCU flash. ‘Fetch from VCU’ could help extract existing calibration data in VCU flash.
6.3.2 Non-Volatile Variable

There are two types of NVM variables, Fixed NVM variables and NVM variables, for different application purposes. Ecotrons Non-Volatile memory theory is described in Appendix A.
Chapter 7  Programming VCU with EcoFlash

EcoFlash is a user-friendly GUI for programming VCU. Please refer to EcoFlash manual for operation of the software. The below picture provides a quick glance of EcoFlash.
Chapter 8  Measurement and Calibration with EcoCAL

EcoCAL is dedicated for data measurement, calibration, logging and analysis. It is a professional calibration tool developed by Ecotrons. EcoCAL is based on CCP/XCP protocols and uses the CAN bus for communication between master-slave stations. It provides great convenience for VCU in-vehicle testing and prototype development.
Chapter 9 Uninstall EcoCoder

9.1 Uninstall EcoCoder from MATLAB

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

1. Double-click ‘EcoCoder Loader’.

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

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

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

1. Click ‘Start’ and then click ‘settings’, follow steps in picture below
Chapter 10 FAQs

Q1. The m file exported from DBC by ‘EcoCAN’ can’t be used
A1. The name of the m file must match to the C Language variable naming requirement.
And it can’t be the name of the existing model or m file.

Q2. Model created by ‘EcoCoder_Prj’, emulation or code generation error
1. Check if your MATLAB has Fixed-Point Tool license. If not, the use of fixed-point blocks will trigger errors.
2. Make sure all support files are added to path.
3. Check whether necessary MATLAB components are installed.

Q3. ‘CAN’ module is blank after being configured
Please check whether the CAN definition .m file is added to Path.

Q4. EcoCoder Loader Pop-up error
You may have to register the ‘comdlg32.ocx’ to windows.

Q5. How to update application model to be compatible with updated EcoCoder
a) EcoCoder Target Definition
The model must include the EcoCoder Target Definition

b) Model configuration module

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

c) Resolve some disabled modules

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

1) Right-click on the disable block, and select Library link->Resolve Link
2) Restore all disable linked blocks.

Q6. Is there a way to get rid of popping up folder of generated file?
Yes. Please go through these steps, Configuration Parameters -> Code Generation -> EcoCoder Debug Options -> Compiled Action. Then you could configure.

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

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

OpenTarget: It will open folder which has generated files.

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

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

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

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

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

Removed: ‘XX_CWprj’ will be removed when it finishes generating files.
Appendix A - Nonvolatile Variables Theory

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

Non-volatile Variables

The NVM variables are stored in flash, which can maintain information even the VCU is powered off (unlike RAM, which would lose data after the VCU power off). Furthermore, the NVM variables will be kept the same even the VCU is programmed (unless it is required to be changed by user configuration), so critical data such as odometer data will not be lost even the VCU software update is performed.

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

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

Fixed Non-volatile Variables

The fixed NVM variables are stored in specific space of flash and arranged in the order defined by definition block, which means specific addresses in flash are reserved for specific variables.
If new variables need to be added to fixed NVM space, it is necessary to re-initialize by calling definition block.

**Battery Input**

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

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