User's Manual

Getting Started Document for P1x-C MCAL Driver

User's Manual

Version 1.0.3

Target Device:
RH850/P1x-C

Renesas Electronics
www.renesas.com

Rev.1.01 May 2017

Abbreviations and Acronyms

Abbreviation / Acronym	Description
ARXML/arxml	AUTOSAR xml
AUTOSAR	Automotive Open System Architecture
BSWMDT	Basic Software Module Description Template
<MSN>	Module Short Name
ECU	Electronic Control Unit
GUI	Graphical User Interface
MB	Mega Bytes
MHz	Mega Hertz
RAM	Random Access Memory
xml/XML	eXtensible Markup Language
<MICRO_VARIANT>	P1x-C
<MICRO_SUB_VARIANT>	P1H-C, P1M-C , P1H-CE
AUTOSAR_VERSION	4.0.3
DEVICE_NAME	Example :R7F701372EAFP

Definitions

Terminology	Description
.xml	XML File.
.one	Project Settings file.
.arxml	AUTOSAR XML File.
ECU Configuration Parameter Definition File	The ECU Configuration Parameter Definition is of type XML, which contains the definition for AUTOSAR software components i.e. definitions for Modules, Containers and Parameters. The format of the XML File will be compliant with AUTOSAR ECU specification standards.
ECU Configuration Description File	The ECU Configuration Description file in XML format, which contains the configured values for Parameters, Containers and Modules. ECU Configuration Description XML File format will be compliant with the AUTOSAR ECU specification standards.
BSWMDT File	The BSWMDT File in XML format, which is the template for the Basic Software Module Description. BSWMDT File format will be compliant with the AUTOSAR BSWMDT specification standards.
Configuration XML File	Configuration XML File is in XML format, which contains command line options and options for input/output file path.

Table Of Contents

List Of Figures

List Of Tables

Chapter 1 Introduction

This document describes the information about installation, usage of ECU Configuration Editor (ECU Spectrum), MCAL Code Generator Tool and references to sections in the Component User Manuals that helps the user reference for building the executable.

ECU Spectrum is a tool that dynamically generates GUI controls for specified AUTOSAR components according to ECU Configuration Parameter Definition File and generates ECU Configuration Description file complying with AUTOSAR standards.

MCAL Code Generator Tool is a command line tool that accepts ECU Configuration Description File(s), BSWMDT File and Configuration XML File as input and generates the C source and header files based on the configuration of the module.

Chapter 2 ECU Configuration Editor (ECU Spectrum)

2.1. Installation Of ECU Configuration Editor (ECU Spectrum)

The Following procedure is to be followed for proper installation of the software:

Copy all the files from the installation disk to a separate folder on to the hard disk of the computer on which the application is to be installed (recommended but not mandatory). Initialize the ‘setup.exe’ file (This can also be done by directly clicking on the same file in the installation disk).

An Install Shield application is invoked which guides the user throughout the installation of the software.

The ECU Spectrum installation Disk1 consists of the following files:

- data1.cab
- data1.hdr
- data2.cab
- engine32.cab
- layout.bin
- setup.bmp
- setup.exe
- setup.ibt
- setup.ini
- setup.inx
- setup.skin

The user is recommended to take backup of the installation disk before proceeding with the actual installation. Due to certain reasons if the installation procedure is aborted, the backup can be used.

The installation procedure is divided into ten steps. The details of all the steps are given below.

Step 1:

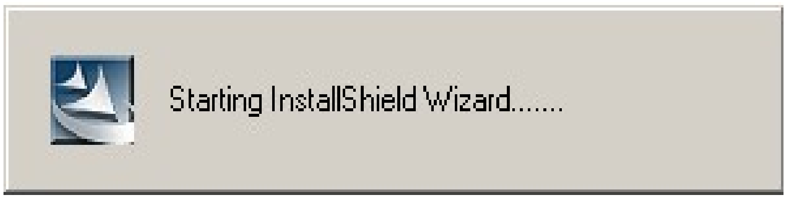

Starting InstallShield Wizard.......

Figure 2-1 Installation Initiation

Run ‘setup.exe’ by double clicking on the setup.exe icon. This operation shows the progress indication dialog as shown in the above Figure 2-1. After displaying above Figure 2-1, for few minutes ECU Spectrum splash screen will appear.

Figure 2-2 Splash Screen

After displaying splash screen for few seconds following 'Preparing Setup' dialog box appears (Refer Figure 2-3).

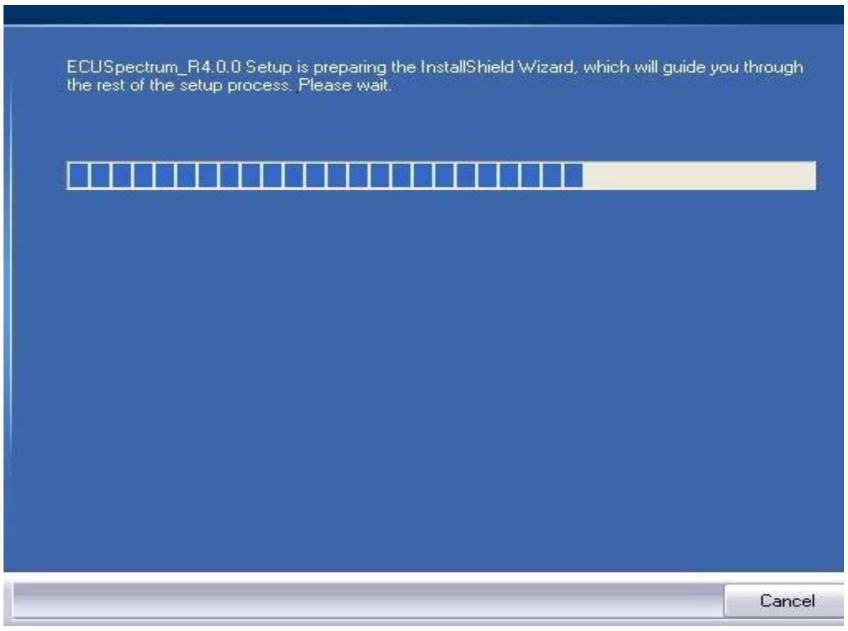
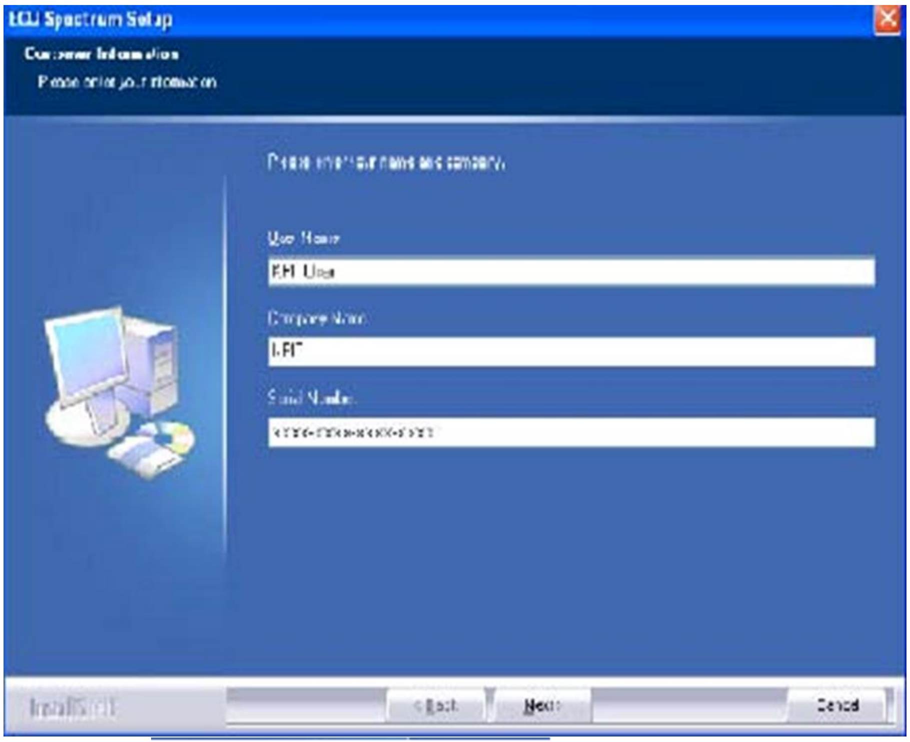

ECUSpectrum_R4.0.0 Setup is preparing the InstallShield Wizard, which will guide you through the rest of the setup process. Please wait.

Figure 2-3 ECU Spectrum installation step 1

Step 2:


ECU Spectrum Setup

Customer Information

Please enter your information

Please enter your name and company.

User Name

KPIT User

Company Name

KPIT

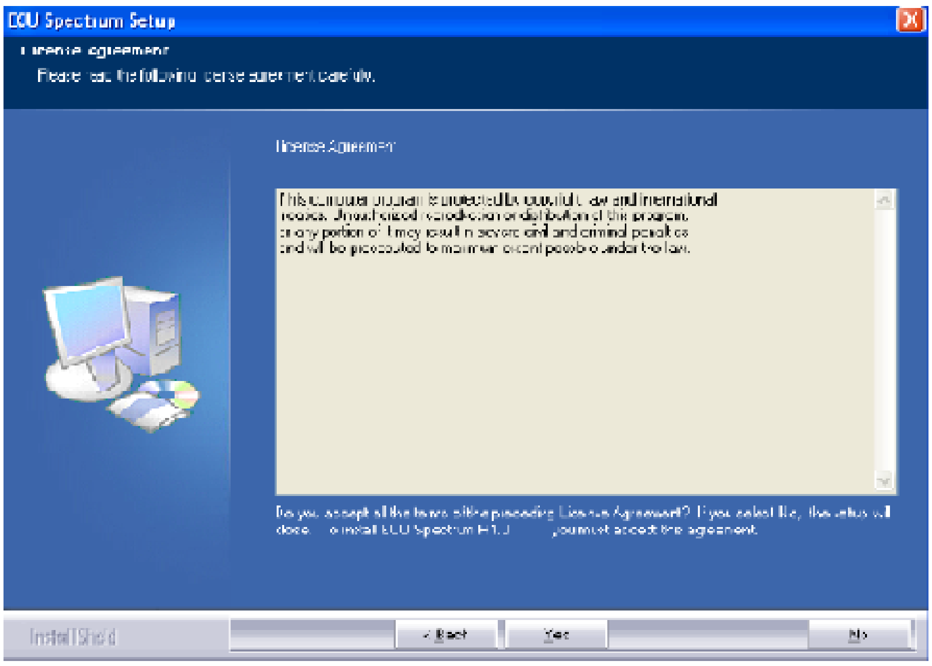
Serial Number

[illegible]

Figure 2-4 ECU Spectrum installation step 2

After completion of the above operation, another dialog box is displayed (Refer Figure 2-4) in order to get confirmation from the user to proceed with the installation. The user can cancel the installation of software by selecting 'Cancel' button'. To proceed with the installation select 'Next' button.

Step 3:



Please read the following license agreement carefully.

License Agreement

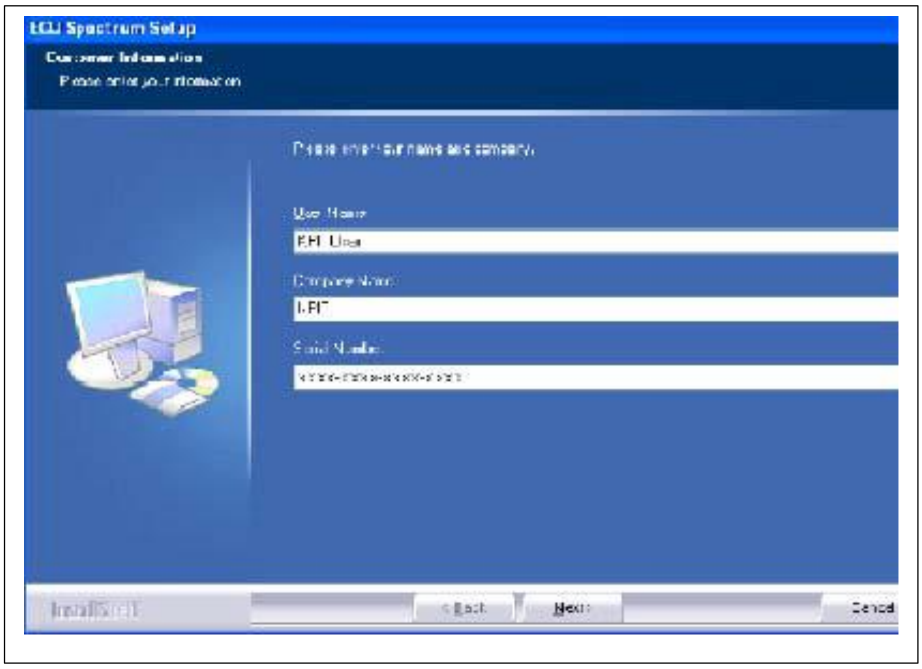
This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties and will be prosecuted to maximum extent possible under the law.

Do you accept all the terms of the preceding License Agreement? If you select No, the setup will close. To install ECU Spectrum H1.0 [illegible] you must accept this agreement.

Figure 2-5 ECU Spectrum installation step 3

On selecting ‘Next’ button in Step 2, a dialog box is invoked displaying the license agreement. If the terms and conditions of the agreement are acceptable then select ‘Yes’ button else select ‘No’ button to abort the installation. The user can select ‘Back’ button in order to modify previously made settings. (Refer Figure 2-5)

Step 4:



Please enter your information.

Please enter your name and company.

User Name: KPIT User

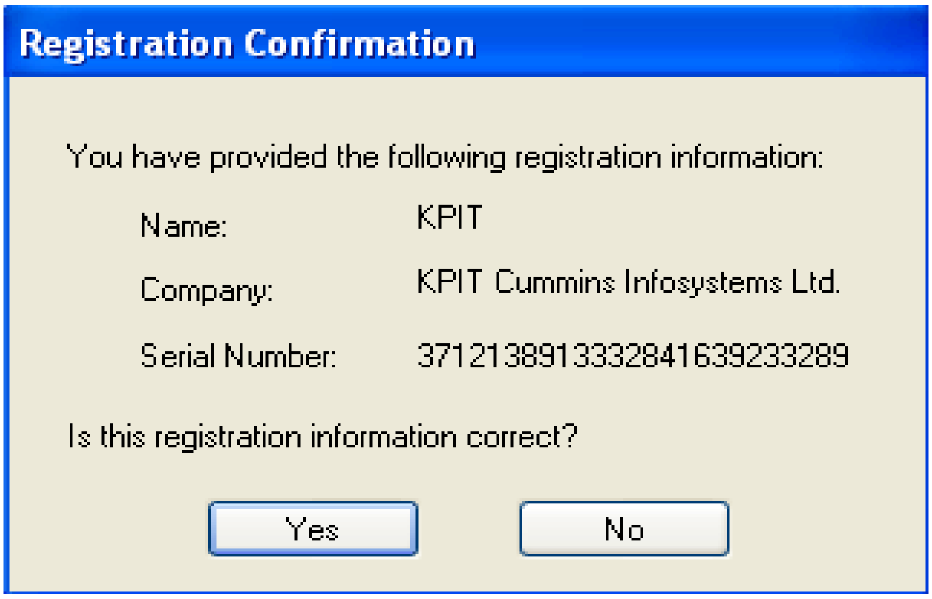
Company Name: KPIT

Serial Number: [illegible]

Figure 2-6 ECU Spectrum installation step 4

Step 5:

'Customer Information' dialog is displayed. Enter the User Name, Company Name and Serial Number and click on 'Next' button to proceed for further installation procedure. (Refer Figure 2-6)



Registration Confirmation

You have provided the following registration information:

Name: KPIT

Company: KPIT Cummins Infosystems Ltd.

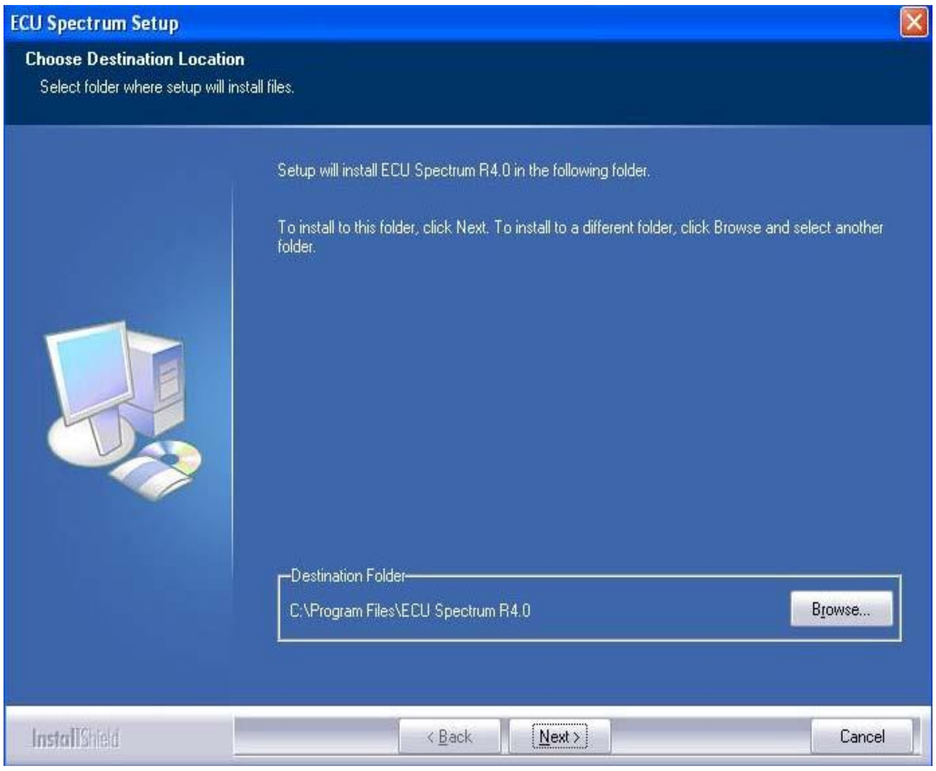
Serial Number: 3712138913332841639233289

Is this registration information correct?

Figure 2-7 ECU Spectrum installation step 5

Dialog box is displayed for user registration confirmation. Check the appeared user registration information, if yes click on 'Yes' button. (Refer Figure 2-7). If not click on 'No' and re-enter the user registration information.

Step 6:



ECU Spectrum Setup

Choose Destination Location

Select folder where setup will install files.

Setup will install ECU Spectrum R4.0 in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

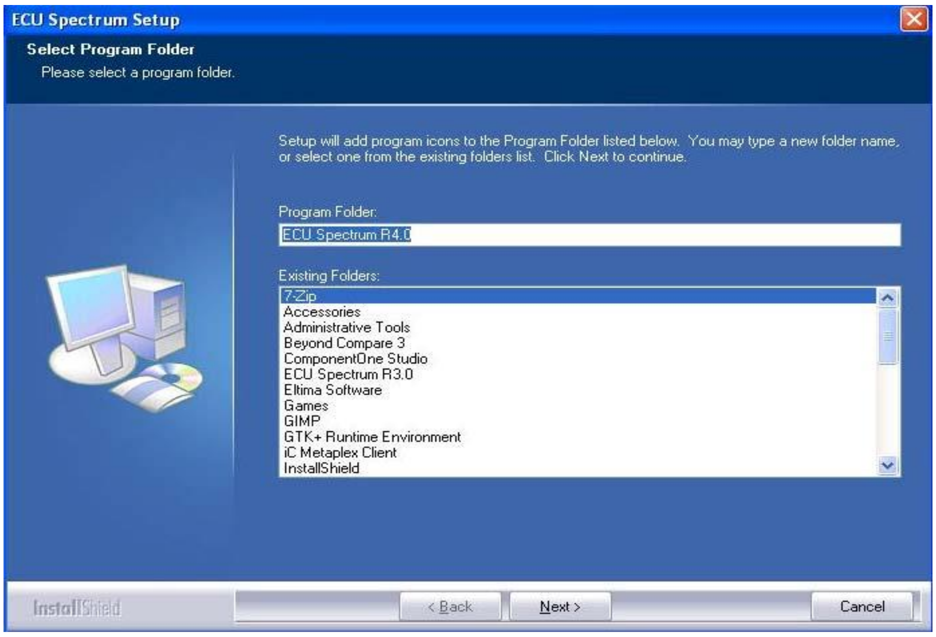
Destination Folder

C:\Program Files\ECU Spectrum R4.0

Figure 2-8 ECU Spectrum installation step 6

Next, a dialog box allowing the user to select the destination folder is displayed (Refer Figure 2-8). The default directory for installation selected by the Install shield is C:\Program Files\ECU Spectrum R3.0. However the user can select the folder for installation of his/her choice using the 'Browse' button. The user can cancel the installation of software by selecting 'Cancel' button and click on 'Next' button to proceed for further installation procedure.

Step 7:



Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing folders list. Click Next to continue.

Program Folder:
ECU Spectrum R4.0

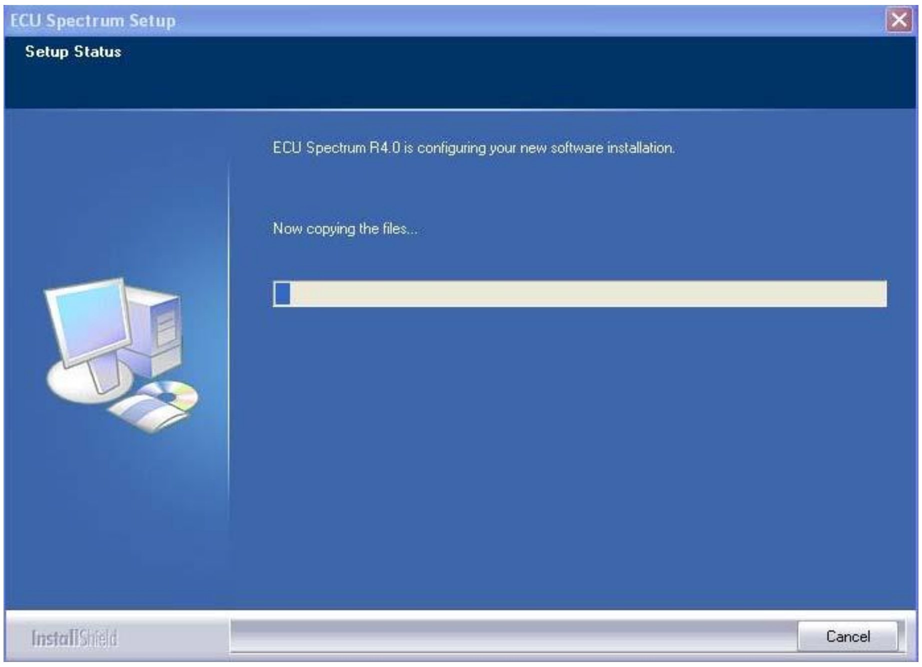
Existing Folders:
7-Zip
Accessories
Administrative Tools
Beyond Compare 3
ComponentOne Studio
ECU Spectrum R3.0
Eltima Software
Games
GIMP
GTK+ Runtime Environment
iC Metaplex Client
InstallShield

Figure 2-9 ECU Spectrum installation step 7

Next, a dialog box allowing the user to select the program folder is displayed. By default, the name of the folder is 'ECU Spectrum R3.0' and the user is allowed to change the folder name. (Refer Figure 2-9)

Select 'Next' button to continue the installation and 'Cancel' button to abort the installation.

Step 8:


ECU Spectrum Setup

Setup Status

ECU Spectrum R4.0 is configuring your new software installation.

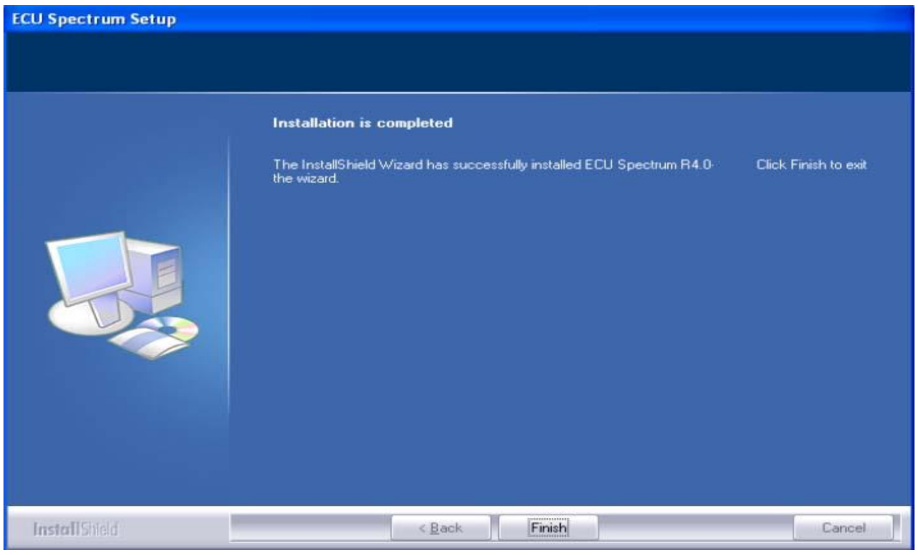
Now copying the files...

Figure 2-10 ECU Spectrum installation step 8

After selecting the appropriate folder for installation, the install wizard will display a dialog box displaying the status of the files being copied. (Refer Figure 2-10).

The user is allowed to abort the installation by pressing ‘Cancel’ button.

Step 9:


ECU Spectrum Setup

Installation is completed

The InstallShield Wizard has successfully installed ECU Spectrum R4.0. Click Finish to exit the wizard.

Figure 2-11 ECU Spectrum installation step 9

After confirmation from the user for copying the files mentioned in Step 8, the install wizard will automatically install the ECU Spectrum application, based on the selections made by the user. After completion of the installation procedure, a dialog gets displayed to intimating the user about completion of the installation process. (Refer Figure 2-11).

Step 10: Click on ‘Finish’ button to complete the installation process.

2.2. ECU Spectrum Input and Output Files

ECU Spectrum takes ECU Configuration Parameter Definition File(s) as input. It reads the definitions, provides a generic interface to edit values for all the configuration parameters and generates the ECU Configuration Description file(s) in XML format. It resolves relatively simple dependencies explicitly defined in the ECU Configuration Parameter Definition file. On the Plug-In side, the user can choose the MCAL Code Generator Tool executable for the individual components that takes ECU Configuration Description File as input and generates the required ‘C’ source and header files.

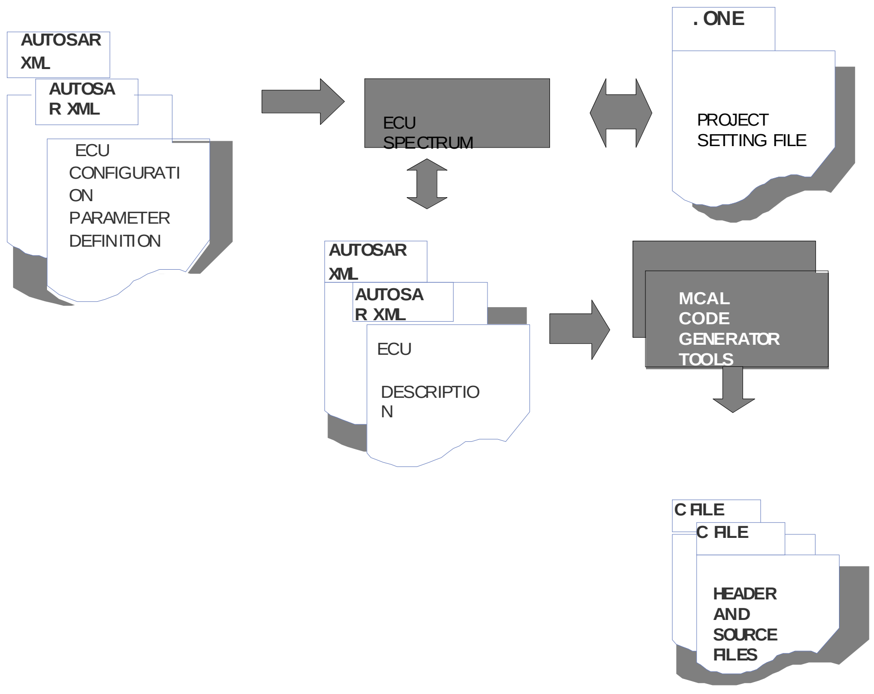


Figure 2-12 ECU Spectrum Overview

Chapter 3 Configuration Using ECU Configuration Editor (ECU Spectrum)

This section gives details about procedure for creating a new project, configuring the parameters, saving a project, validating the current GUI configuration and generating ECU Configuration Description file of ECU Spectrum.

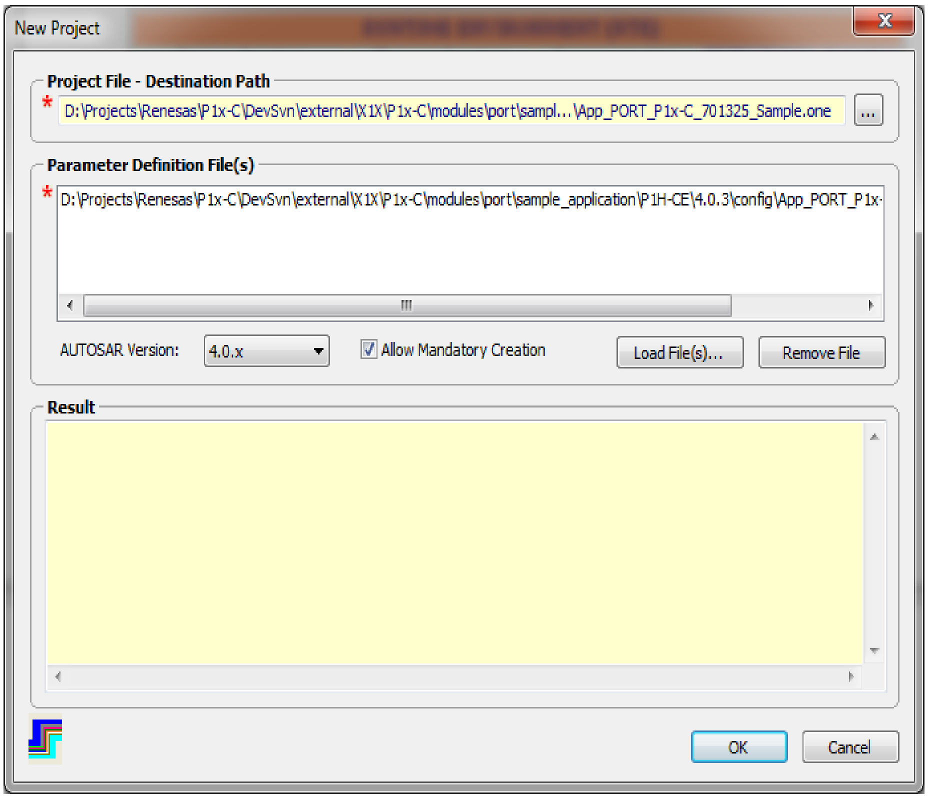
3.1. Creating New Project

Creating a ‘New Project’ loads the modules from specified ECU Configuration Parameter Definition File into the Software. New Project is created using “File -> New Project” from the main menu.

Steps to be followed:

1. Select “File -> New Project”.
2. Select the AUTOSAR Version. Default AUTOSAR version is 4.0.x.
3. Browse a valid Project Settings file name (of type ‘.one’).
4. Browse a valid ECU Configuration Parameter Definition File name (of type ‘*.xml /*.arxml’).
5. Click on ‘OK’ button.
6. Follow step 4 to load more than one definition file.

The modules available in the selected files get loaded into the software and it is saved in the specified Project Settings file location. Specified Project Settings File name is displayed on the title bar of the ECU Spectrum along with the respective AUTOSAR version.



D:\Projects\Renesas\P1x-C\DevSvn\external\X1X\P1x-C\modules\port\sampl...\App_PORT_P1x-C_701325_Sample.one

D:\Projects\Renesas\P1x-C\DevSvn\external\X1X\P1x-C\modules\port\sample_application\P1H-CE\4.0.3\config\App_PORT_P1x-

AUTOSAR Version: 4.0.x

[✓] Allow Mandatory Creation

Figure 3-1 Creating New Project

The modules available in the selected files get loaded into the Software and it is saved in the specified Project Settings location. Specified Project Settings file name is displayed on the Title bar of the Software.

3.2. Configuration

Right click on the module in the 'Left Selection View', a popup menu having 'New Module' option is displayed. On selecting this option, the instance of the module is created. The ECU Spectrum will assign a short name to the newly created module automatically. On selecting the newly created module, controls are displayed in the 'Right Configuration View' for configuration. Edit the data and validate the current GUI configuration. Errors/Warnings/Messages is displayed in the 'Message Info' window, if any.

The user can configure any number of modules, containers, parameters, and references depending on the Multiplicity specified in the ECU Configuration Parameter Definition File.

Right clicking on the instance of the module in 'Left Selection View', a popup menu having 'Insert Copy' ,'Delete' ,'Expand' and 'Collapse' option is displayed. Using 'Insert Copy', the copy of selected element with configured values is inserted. 'Insert Copy' option is displayed in the pop up menu based on the multiplicity. Using 'Delete', user can delete the selected element. 'Expand' is used to expand the selected element and 'Collapse' is used to collapse the selected element.

Existing Project Settings can be configured in following ways:

1. Right Click on the module and add an instance of the module.

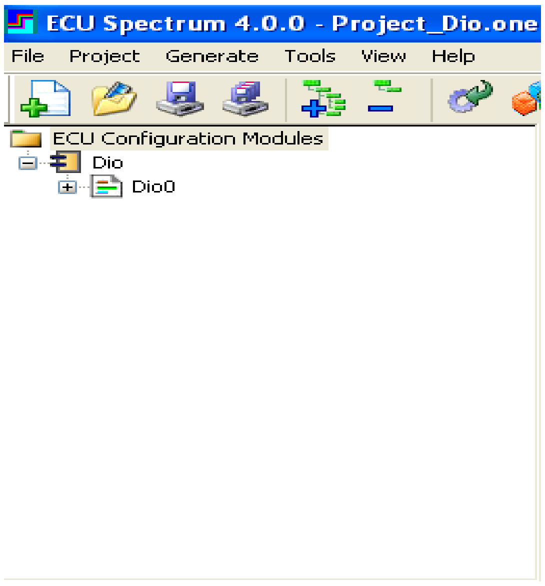
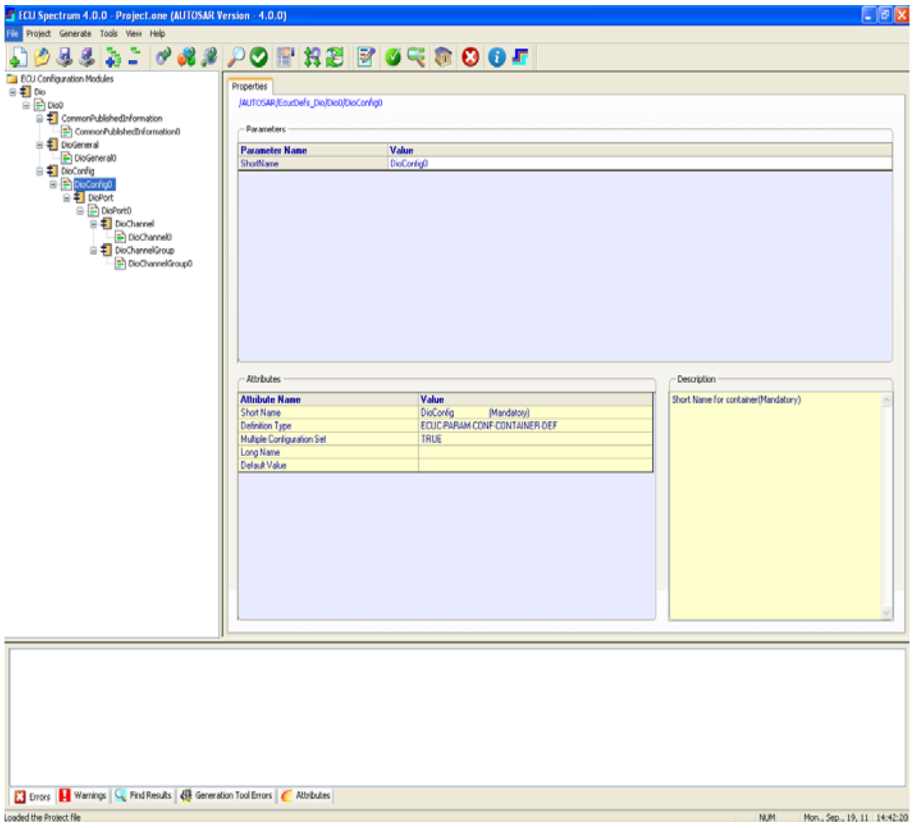

ECU Spectrum 4.0.0 - Project_Dio.one
File Project Generate Tools View Help
ECU Configuration Modules
Dio
Dio0

Figure 3-2 Adding New Module Instance

2.Click on the instance of the module, user will find a grid on right view for configuration.



/AUTOSAR/EcucDefs_Dio/Dio0/DioConfig0

Parameter Name	Value
ShortName	DioConfig0

Attribute Name	Value
Short Name	DioConfig (Mandatory)
Definition Type	ECUC-PARAM-CONF-CONTAINER-DEF
Multiple Configuration Set	TRUE
Long Name	
Default Value	

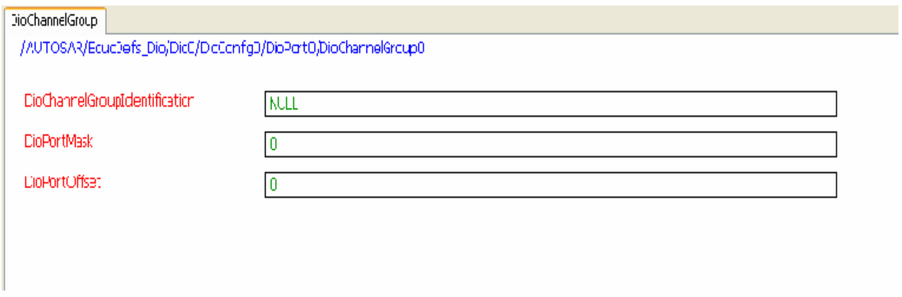
Short Name for container(Mandatory)

Loaded the Project file NUM Mon., Sep., 19, 11 14:42:20

Figure 3-3 GUI for Configuration

3.2.1. Parameter Configuration

Short Name, Definition Type, Lower multiplicity, Upper multiplicity, Minimum value, Maximum value, Implementation config class, Implementation config variant, Default value and Long Name are displayed in ‘Attributes’ grid and Description of the parameter is displayed in the ’Description’ display area on click of the parameter in the ‘Right Configuration View’ as shown in the following figure. Configure the parameter and press ‘ENTER’ button. Following are the types of the parameters:



DioChannelGroup

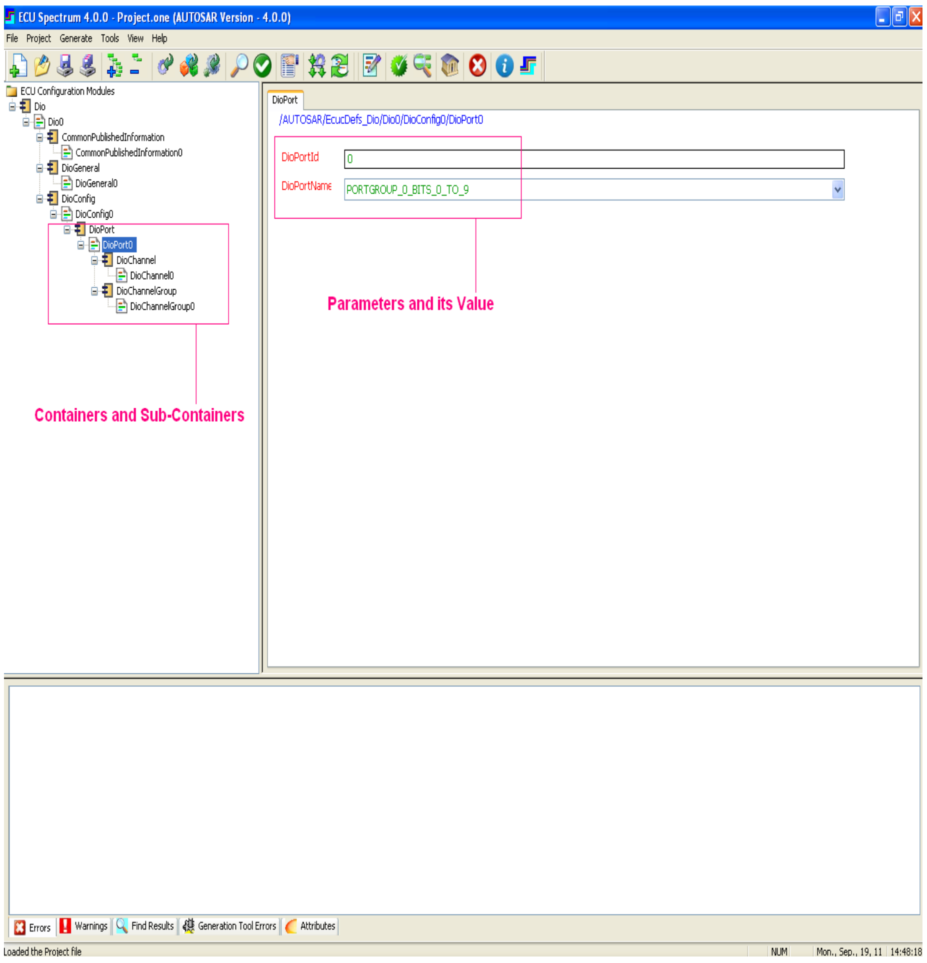
/AUTOSAR/EcucDefs_Dio/Dio0/DioConfig0/DioPort0/DioChannelGroup0

DioChannelGroupIdentification	NULL
DioPortMask	0
DioPortOffset	0

Figure 3-4 Parameter Configuration

3.2.2. Distinguish Between Containers

On selecting the newly created module in the ‘Left Selection View’, controls are displayed in the 'Right Configuration View' for configuration. Name of the containers is displayed at the top of the ‘Right Configuration View’ as shown in the following figure. On selecting the container, Parameters and sub-containers is displayed in the grid control as shown in the following figure.



/AUTOSAR/EcucDefs_Dio/Dio0/DioConfig0/DioPort0

DioPortId	0
DioPortName	PORTGROUP_0_BITS_0_TO_9

Loaded the Project file

Mon., Sep., 19, 11 14:48:18

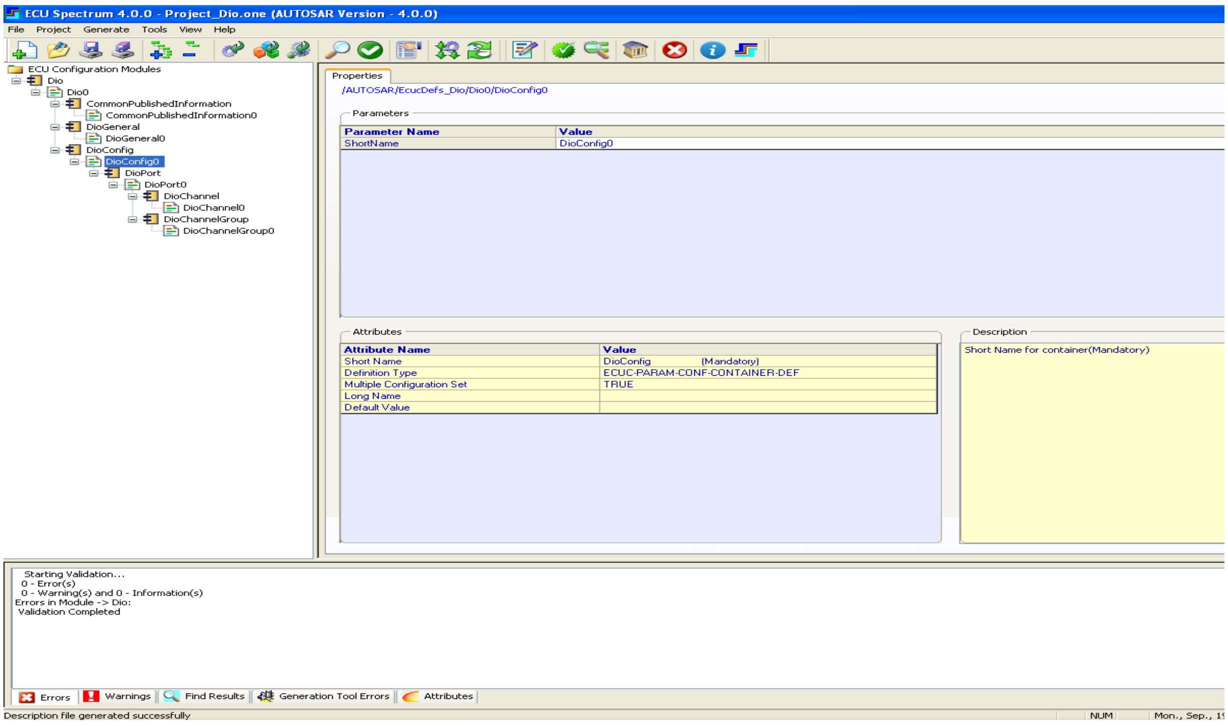
Figure 3-5 Distinguish Between Containers

3.2.3. Save Project

Using “File-> Save Project” menu item, current GUI configuration can be saved with specified Project Settings file name.

3.2.4. Validation

The modules configuration can be checked for correctness and completeness in validation. Before generation of ECU Configuration Description, validate the configured values of the modules. Select “Project -> Validate” or press ‘F8’ key, a current GUI configuration is validated and list of Errors/Warnings/Messages is displayed in the 'Message Info' window, if any.



/AUTOSAR/EcucDefs_Dio/Dio0/DioConfig0

Parameter Name	Value
ShortName	DioConfig0

Attribute Name	Value
Short Name	DioConfig (Mandatory)
Definition Type	ECUC-PARAM-CONF-CONTAINER-DEF
Multiple Configuration Set	TRUE
Long Name	
Default Value	

Short Name for container(Mandatory)

```
 Starting Validation...
 0 - Error(s)
 0 - Warning(s) and 0 - Information(s)
Errors in Module -> Dio:
 Validation Completed
```

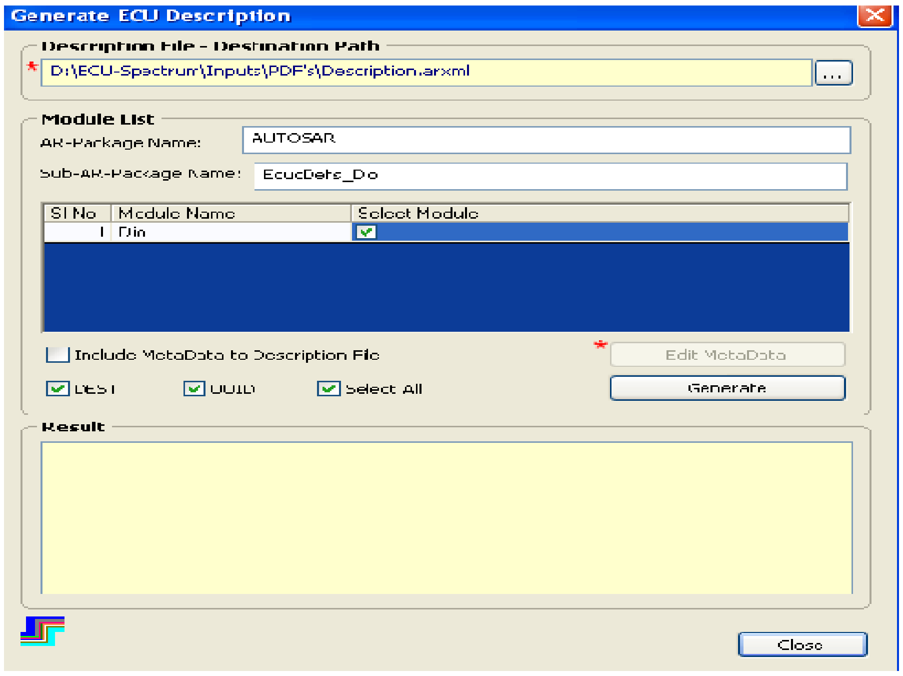
Description file generated successfully

Figure 3-6 Validation

3.3. Generate ECU Configuration Description

This generates the ECU Configuration Description File, which contains the configured values for Parameters, Containers and Modules. The generated ECU Configuration Description File format will be compliant with the AUTOSAR ECU specification standards. After validation of the configured values, to generate the ECU Configuration Description follow the below steps:

1. Select "Generate -> ECU Configuration Description".
2. Check the 'Select all' Check box.
3. Specify the ECU Configuration Description File name (of type '*.xml/ *.arxml').
4. Click 'Generate' button


Generate ECU Description

Description File - Destination Path: D:\ECU-Spectrum\Inputs\PDF's\Description.arxml

Module List

AR-Package Name: AUTOSAR

Sub-AR-Package Name: EcucDefs_Do

Sl No	Module Name	Select Module
1	Dio	✓

Include MetaData to Description File [ ]

DEST [✓] UUID [✓] Select All [✓]

Figure 3-7 Description File Generation

Successful generation message is displayed in the 'Result' display area. The ECU Configuration Description data for all modules is generated at the specified location.

Chapter 4 Generation Tool

The MCAL Generator Tool is a template based code generator. The template language is Velocity Scripting. The template parsing is done by Apache Velocity engine. MCAL Generator Tool will generate the configuration source files for the module and their respective variant specified along with the command line arguments.

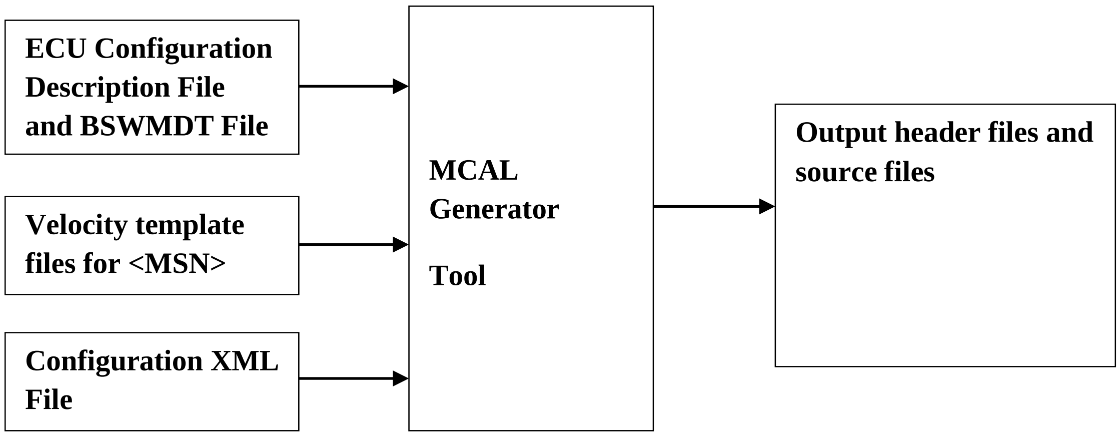

ECU Configuration Description File and BSWMDT File
Velocity template files for <MSN>
Configuration XML File
MCAL Generator Tool
Output header files and source files

Figure 4-1 Generation Tool Overview

4.1. ECU Configuration Description File

This file will contain WDG Driver specific configuration information. This file should be generated by AUTOSAR specified Configuration Editor.

4.2. Velocity template files

They are interpreted by the MCAL Generator Tool in order to provide user input validation and generate the final output file needed by the AUTOSAR configuration chain. They are the "logic" of the MCAL Code Generator Tool.

4.3. Configuration XML File

This file is used to specify which velocity template to use and their location and the name of the output file generated.

4.4. Usage

This section provides the information regarding usage of the Generation Tool. It also provides the syntax of the command line arguments (input filenames and options).

Generation Tool executable is invoked as shown below.

MCALGenerator.exe<space><Description_File_Path><space><Module_Name><space><AR_Package><space><Template_Path><space>–o<space>[Output_Path]<space>–e<space>[Ecu_Variant]<space>–ref<space>[Reference_Module]<space>[Reference_Description_XML_File]

Where,

1. Description_File_Path: The path to the description ARXML file.
2. Module_Name: The name of the module for which the code has to be generated.
3. AR_Package: The AR-Package name as specified in the Description file. While specifying the AR_Package, the full path of the AR_Package is to be specified. See the example for more clarity.
4. Template_Path: The path to the module configuration files for the specified module. This path should also contain the code generation templates.
5. The "Output_Path" is an optional argument to specify the location for the generated source files. The argument should be preceded with optional argument identifier –o
   For example: –o configcode this will generate the code in a folder named "configcode".

   If the output path is not specified, the source files will be generated in the folder where the MCALGenerator.exe is kept.
6. The "Ecu_Variant" is an optional argument to specify the variant model. The argument should be preceded with optional argument identifier –e.
   For example: –e 701372.This will generate the code for the variant 701372
7. The "Reference_Module" and "Reference_Description_XML_File" is an optional argument to specify the reference module and its corresponding xml file. As per our earlier concept, we had the module whose configuration files have to be generated and its reference module in the same XML file. But now, based on our new mechanism, we can have the module whose configuration files has to be generated and its reference module in the same XML file or we can have the reference modules in a different XML file. Both the description file as well as the reference file should have the same ARPACKAGE name. The argument should be preceded with an optional argument identifier -ref.
   For example: -ref Dem0"..\..external\X1X\common_platform\generic\stubs\4.0.3\Dem\config\Dem_<Msn>.arxml"

4.5. Tool Installation Requirements

The minimum hardware and software requirements for proper installation of Module Specific Generation Tool is listed below. This ensures optimal performance of the Tool.

4.5.1. Hardware Requirements

Processor	Pentium/equivalent processor @ 500 MHz or greater
Memory	RAM 64MB or greater
Hard Disk Drive	500 MB or greater storage capacity

4.5.2. Software Requirements

Operating System	Microsoft Windows Platform

4.5.3. Limitations

Command Line characters are limited to 128 depending upon the operating system.

4.6. Tool Installation

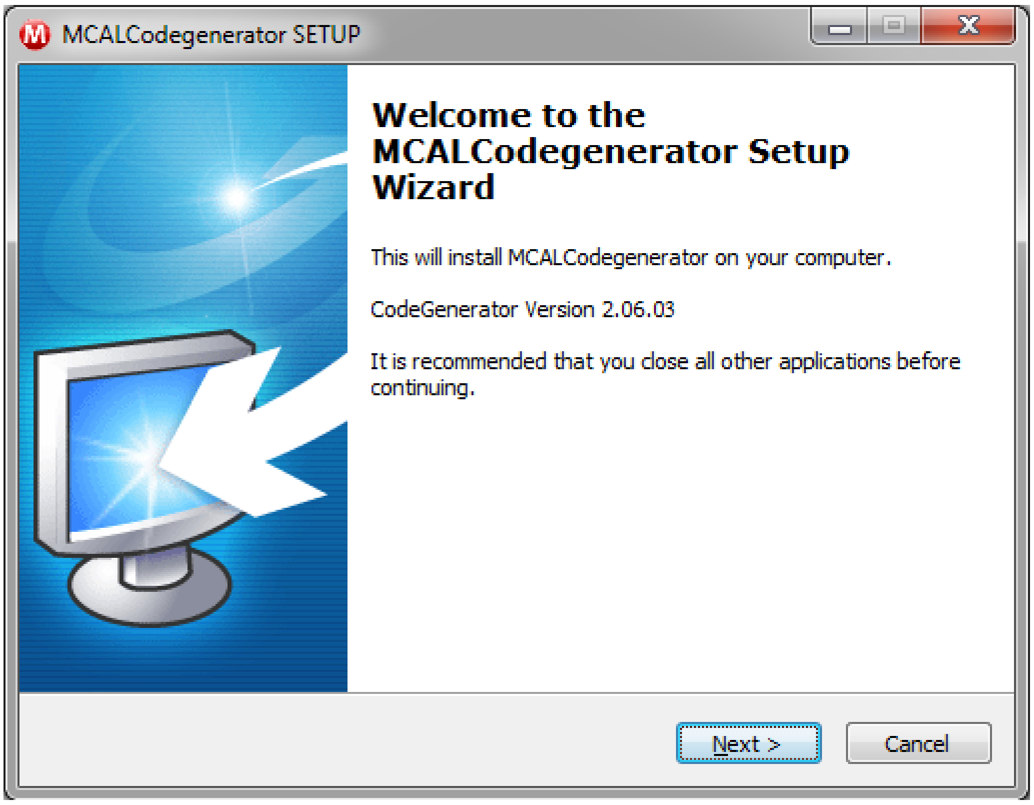
The installation procedure of MCAL Generation Tool is provided in the section below

4.6.1. Pre Requisite

Module Specific Generation Tool executable runs on Windows platforms only.

4.6.2. Installation Steps

Run 'MCALCodegenerator_Installer' by double clicking on the MCALCodegenerator_Installer.exe icon.



Welcome to the MCALCodegenerator Setup Wizard

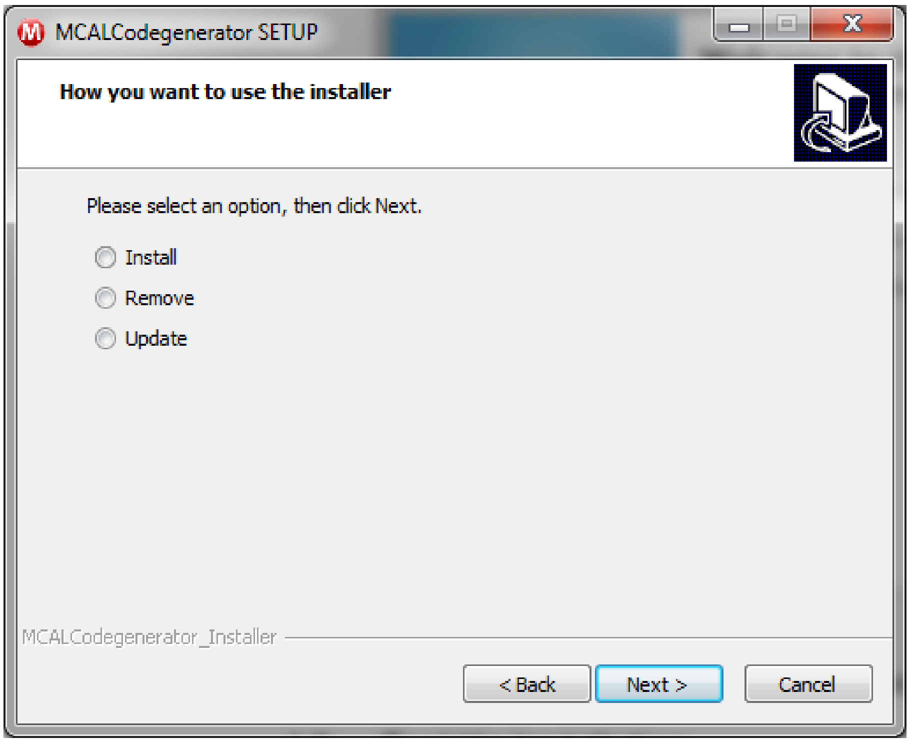
This will install MCALCodegenerator on your computer.

CodeGenerator Version 2.06.03

It is recommended that you close all other applications before continuing.

Figure 4-2 MCAL Generator installation step 1

Select the option to select install or remove or update the software



MCALCodegenerator SETUP

How you want to use the installer

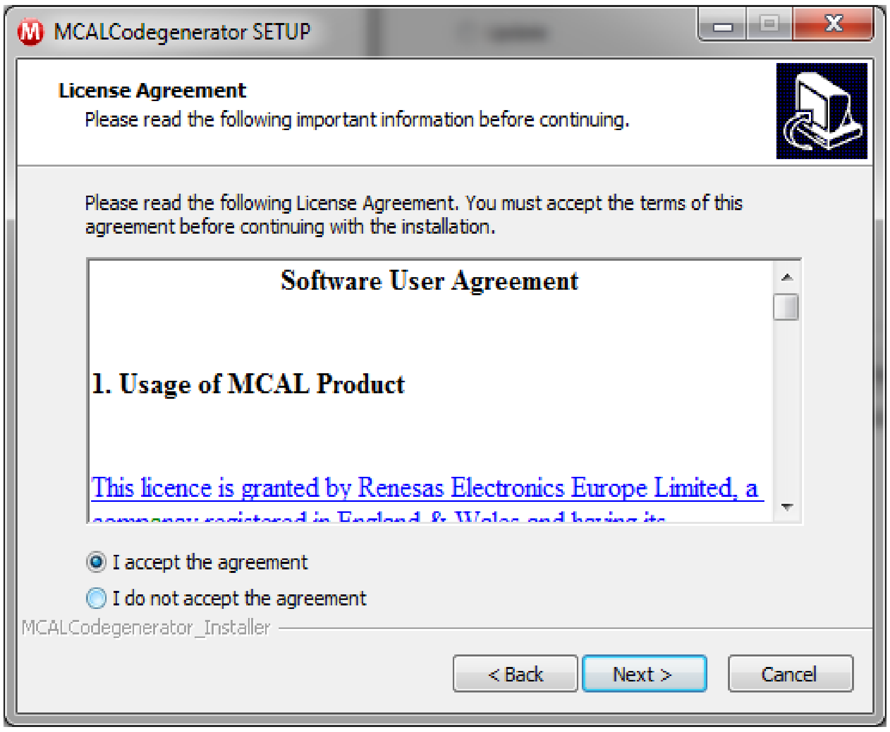
Please select an option, then click Next.

- Install
- Remove
- Update

MCALCodegenerator_Installer

Figure 4-3 MCAL Generator installation step 2

Accept the software User Agreement



MCALCodegenerator SETUP

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

Software User Agreement

1. Usage of MCAL Product

This licence is granted by Renesas Electronics Europe Limited, a

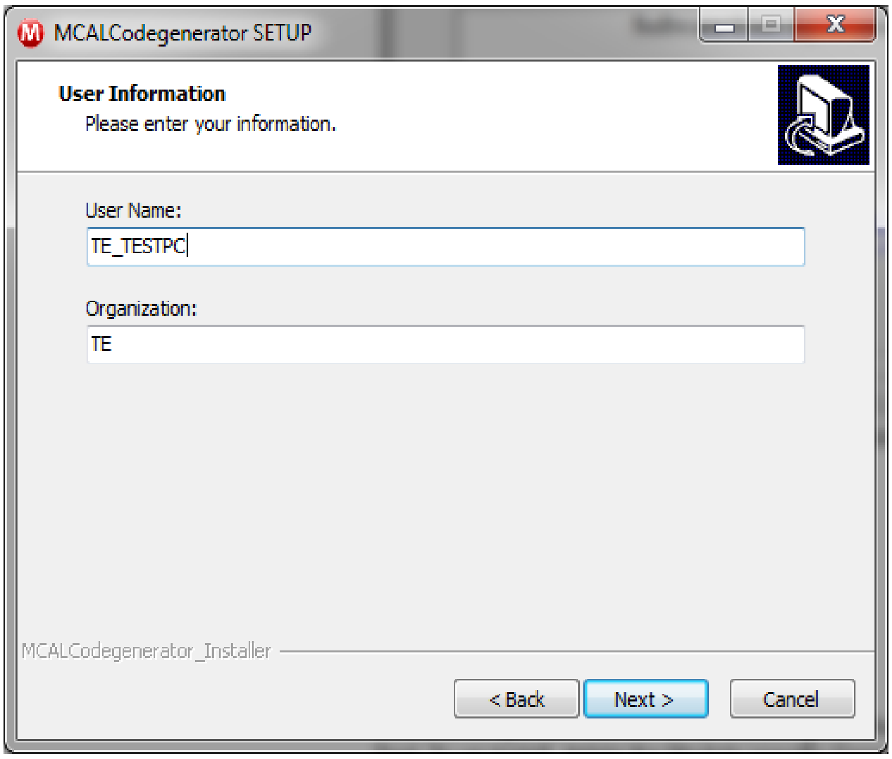
[✓ I accept the agreement]

I do not accept the agreement

MCALCodegenerator_Installer

Figure 4-4 MCAL Generator installation step 3

Fill the User Information


MCALCodegenerator SETUP

User Information

Please enter your information.

User Name:

TE_TESTPC

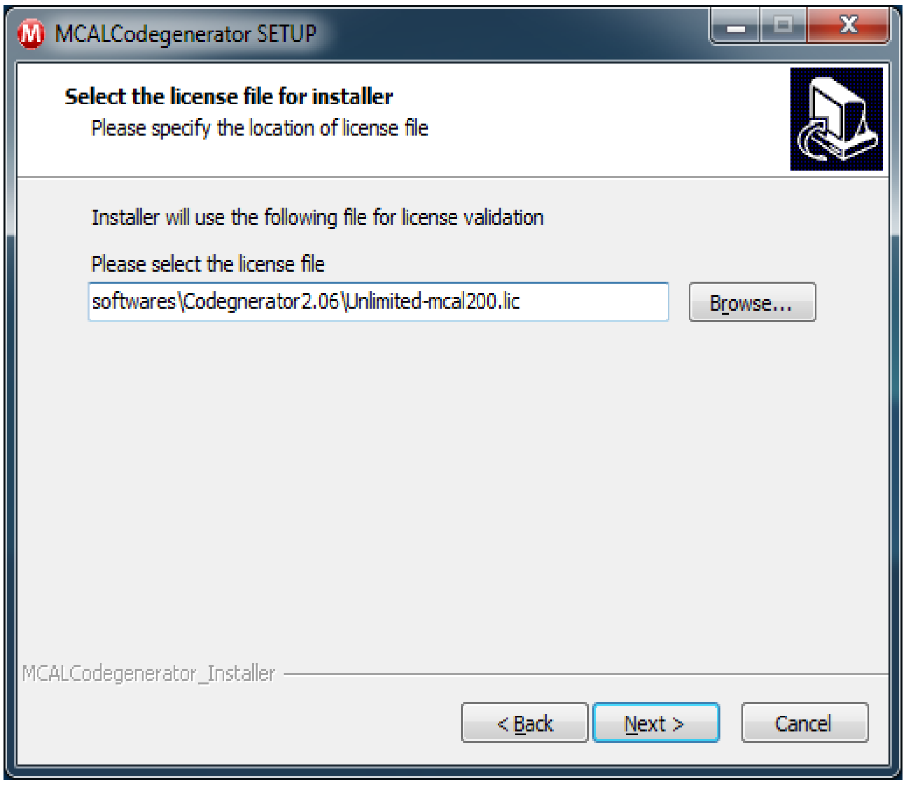
Organization:

TE

MCALCodegenerator_Installer

Figure 4-5 MCAL Generator installation step 4

Select the license file


MCALCodegenerator SETUP

Select the license file for installer

Please specify the location of license file

Installer will use the following file for license validation

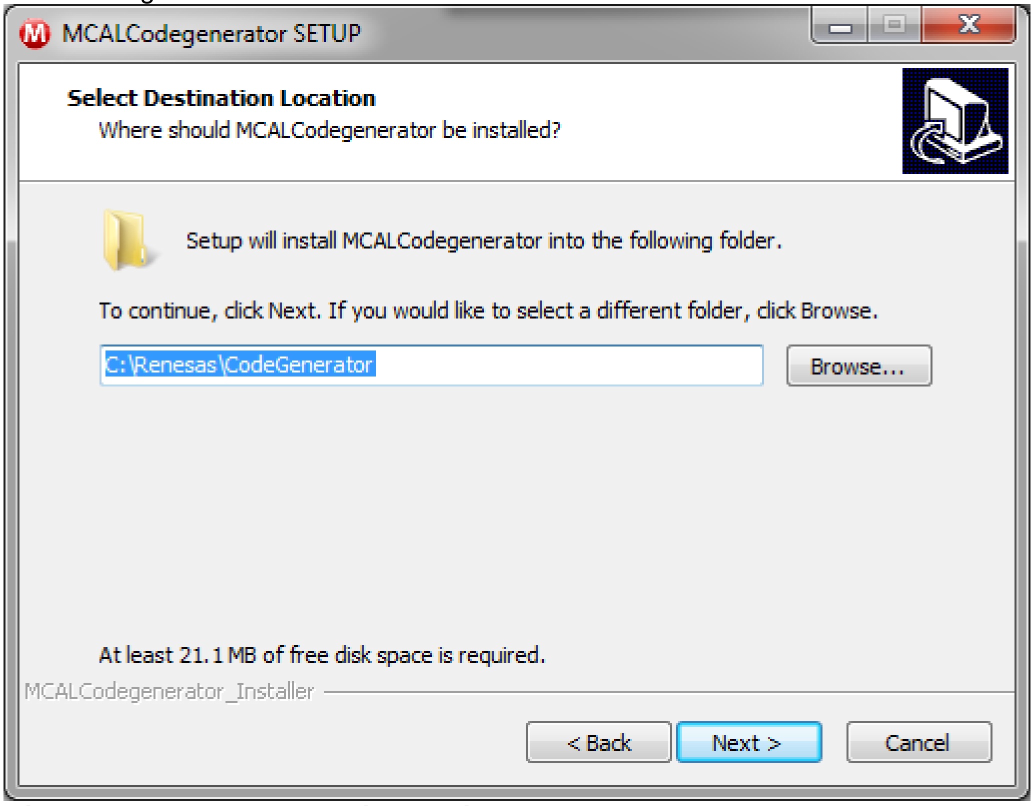
Please select the license file

softwares\Codegnerator2.06\Unlimited-mcal200.lic

MCALCodegenerator_Installer

Figure 4-6 MCAL Generator installation step 5

The default directory for installation selected by the Install shield is C:\Renesas\CodeGenerator. However, the user can select the folder for installation of choice using the ‘Browse’ button.



MCALCodegenerator SETUP

Select Destination Location
Where should MCALCodegenerator be installed?

Setup will install MCALCodegenerator into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

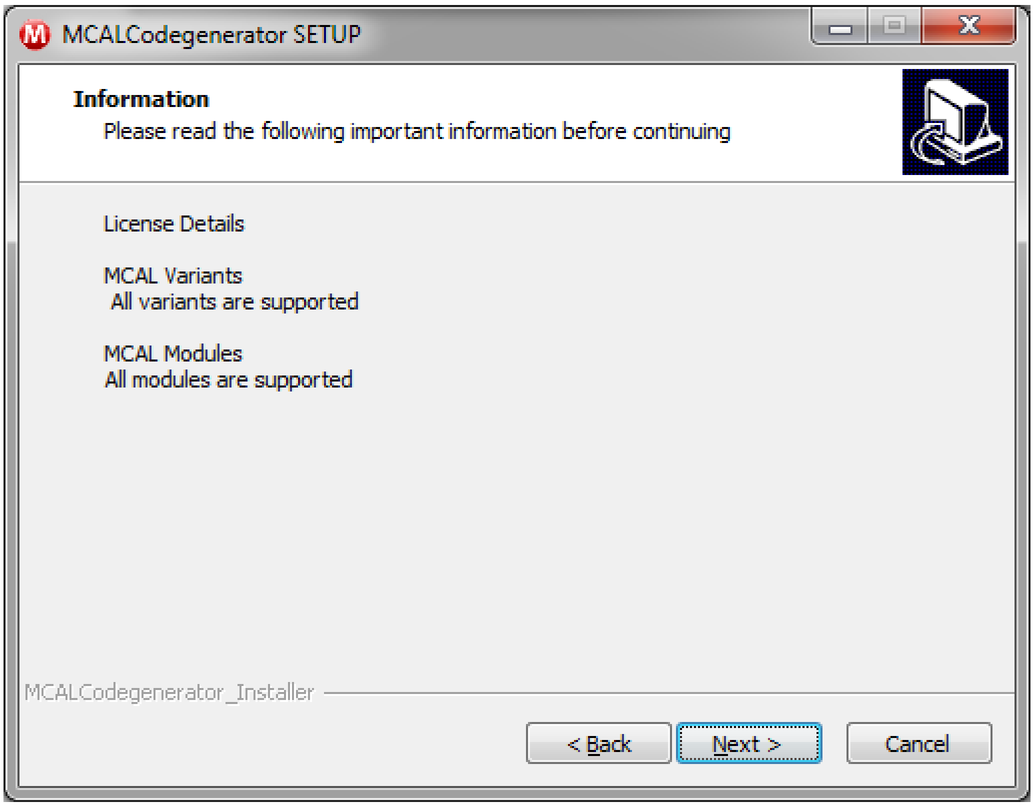
C:\Renesas\CodeGenerator

At least 21.1 MB of free disk space is required.

MCALCodegenerator_Installer

Figure 4-7 MCAL Generator installation step 6

The license information based on the selected license file will be displayed.



MCALCodegenerator SETUP

Information
Please read the following important information before continuing

License Details

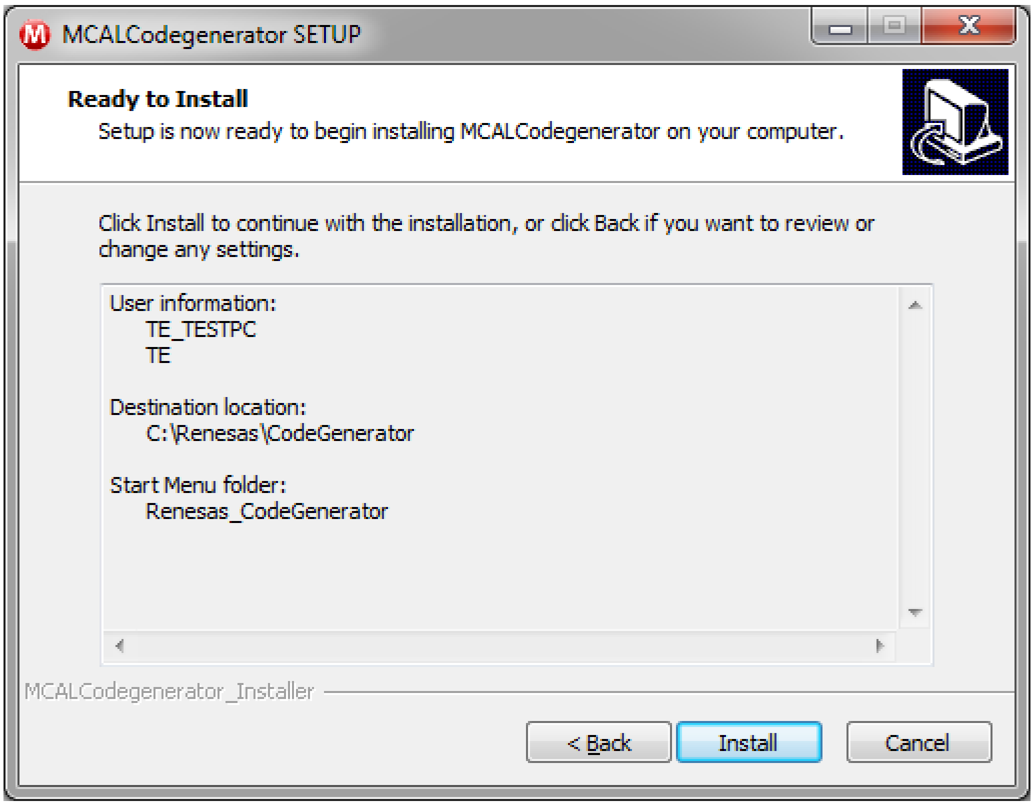
MCAL Variants
All variants are supported

MCAL Modules
All modules are supported

MCALCodegenerator_Installer

Figure 4-8 MCAL Generator installation step 7

Click Install



Ready to Install

Setup is now ready to begin installing MCALCodegenerator on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

User information:
TE_TESTPC
TE

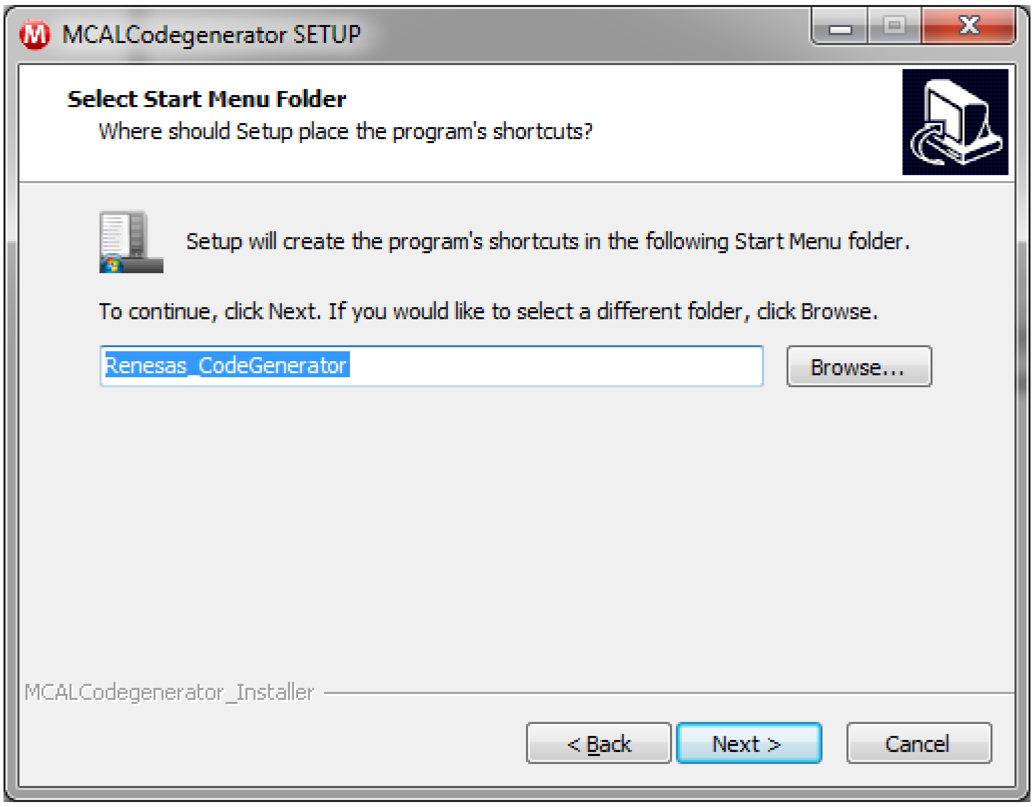
Destination location:
C:\Renesas\CodeGenerator

Start Menu folder:
Renesas_CodeGenerator

MCALCodegenerator_Installer

Figure 4-9 MCAL Generator installation step 8

Select the folder where the program shortcut to be placed



Select Start Menu Folder

Where should Setup place the program's shortcuts?

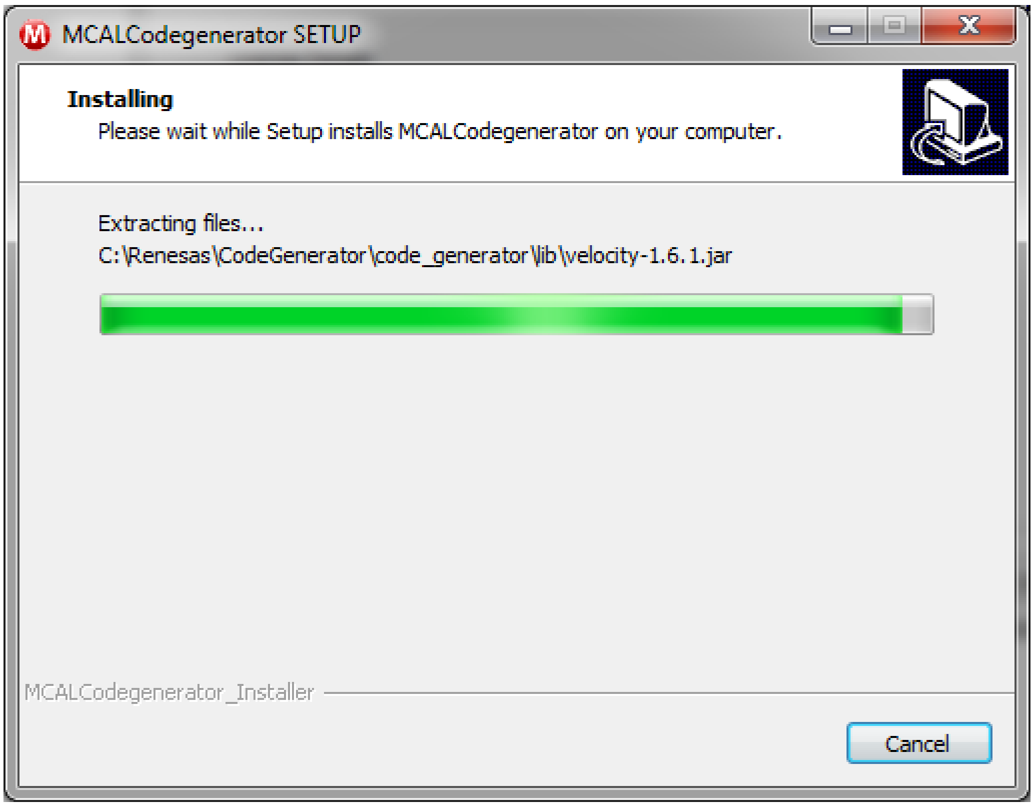
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Renesas_CodeGenerator

MCALCodegenerator_Installer

Figure 4-10 MCAL Generator installation step 9


MCALCodegenerator SETUP

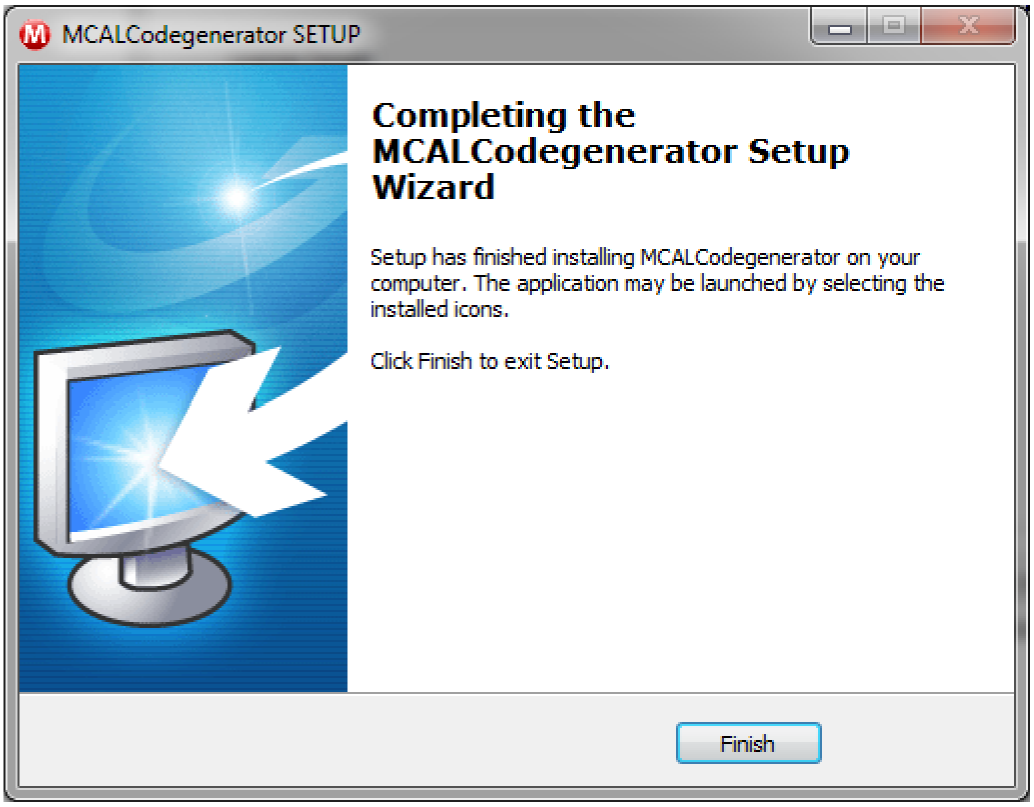
Installing

Please wait while Setup installs MCALCodegenerator on your computer.

Extracting files...
C:\Renesas\CodeGenerator\code_generator\lib\velocity-1.6.1.jar

MCALCodegenerator_Installer

Figure 4-11 MCAL Generator installation step 10


MCALCodegenerator SETUP

Completing the MCALCodegenerator Setup Wizard

Setup has finished installing MCALCodegenerator on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

Figure 4-12 MCAL Generator installation step 11

Click on ‘Finish’ button to complete the installation process.

4.7. Tool Un-Installation

To un-install, use the remove option in step 2

4.8. Common Messages

This section contains the list of error/warning/information messages

which is common for AUTOSAR Renesas R3.2.2 and R4.0.3 X1x MCAL Driver module that is will be generated by the Generation Tool.

4.8.1. Error Messages

ERR000001: File <File_Name> does not exist.

This error occurs, if the input <File_Name> is not found.

ERR000002: Name of the Generation Tool Configuration XML File is not given along with <-C/-CONFIG> option.

This error occurs, if the name of the Generation Tool Configuration XML File is not given along with <-C/-CONFIG> option.

ERR000003: File <File name> is not as per XML standard.

This error occurs, if the input <File name> is not as per XML standard.

ERR000004: Cannot open the <Log file name> file.

This error will occur, if unable to open the <Log file name> file.

ERR000005: Name of output directory is not given along with <-O/-OUTPUT> option.

This error will occur, if the output directory name is not given along with <-O/-OUTPUT> option.

ERR000006: Name of output directory is not given in OUTPUT-PATH tag in <File name>.

This error will occur, if the output directory is not given in OUTPUT-PATH tag in configuration file.

ERR000007: The Generation Tool expects inputs.

This error occurs, if the no option is provided in the command line and none of the option in the configuration file is set.

ERR000008: The option <option> is not supported by the Generation Tool. The Generation Tool supports < -O/-OUTPUT, -Osrc , -Oinc, -H/-HELP, -L/-LOG, -C/-CONFIGFILE and -D/-DRYRUN>" options.

This error will occur, if the invalid <option> is provided to the tool.

ERR000009: Invalid output directory name <output directory name> as the file with same name exists.

This error will occur, if the <output directory name> already exists.

ERR000010: Invalid output directory name <output directory name> Directory name should not contain any of *\?\"\|\: characters.

This error will occur, if the <output directory name> path contains junk character.

ERR000011: ECU Configuration Description File is not provided as input to the Generation Tool.

This error will occur, if the ECU Configuration Description File is not given in the command line or in configuration file.

ERR000012: The input <File name> is not as per XML standard. Provide the ECU Configuration Description File as input on the command line.

This error will occur, if the ECU Configuration Description File is not as per XML standard.

ERR000015: The 'device_name' tag should be present as child of 'TRANSLATION-FILE-PATH'' tag in <File name>.

This error occurs, if the device mentioned in ECU Configuration Description File is not present in

'TRANSLATION-FILE-PATH’ tag in the <File name>.

ERR000016: ‘DEVICE-FILE-PATH’ tag in <File name> is empty.

This error occurs, if the translation file <File name> doesn’t have ‘DEVICE-FILE-PATH’ tags.

ERR000017: The 'device_name’ tag should be present as child of ‘DEVICE-FILE-PATH' tag in <File name>.

This error occurs, if the device mentioned in ECU Configuration Description File is not present in

‘DEVICE-FILE-PATH’ tag.

ERR000018: Cannot create directory <output directory name>.

This error occurs, if unable to create output directory <output directory name>.

ERR000019: Cannot open <File name>.

This error occurs, if unable to open <File name>.

ERR000020: The macro label <macro label> should be unique in < translation file name> translation C Header File.

This error occurs, if macro label is not unique in translation C Header File.

ERR000021: The macro definition for <macro label> macro is not found in <translation file name> translation C Header File. The macro label format should be <label format>.

This error occurs, if macro definition is not found in translation C Header File.

ERR000022: The macro value for <macro label> macro is empty in <translation file name> translation C Header File.

This error occurs, if macro label value is empty in translation C Header File.

ERR000023: The macro definition for <macro value> macro is not found in input device specific C Header File(s).

This error occurs, if macro definition is not found in input device specific C

Header File(s).

ERR000024: The macro value for <macro value> macro is empty in input device specific C Header File(s).

This error occurs, if macro value is empty in input device specific C Header File(s).

ERR000025: Path <Configured Reference Path> provided for Bsw Module is incorrect.

This error occurs, if the reference provided for Bsw Module Component is incorrect.

ERR000026: BSWMDT content is not present in the input file(s) for '<Module Name>' module.

This error occurs, if the module specific BSWMDT content is not present in the input files.

ERR000027: <MSN> BSWMDT File of either AUTOSAR R3.2 or R4.0 should be given as input.

This error occurs, if the both R3.2 and R4.0 BSWMDT file given to the input to the generation tool.

ERR000028: 'MODULE-DESCRIPTION-REF' element should be present in the description file of '<Module Name>' module.

This error occurs, if the MODULE-DESCRIPTION-REF element is not present module specific description file.

ERR000029: AUTOSAR version of BSWMDT File and Module Description File is different.

This error occurs, if the AUTOSAR version of the BSWMDT File and module description file is different.

4.8.2. Warning Messages

WRN000001: As per AUTOSAR ECU Configuration Description File naming convention, the file extension should be '.arxml' for file.

This warning occurs, if ECU Configuration Description file having an extension other than '.arxml'.

4.8.3. Information Messages

INF000001: Tool Version:

This is to display Tool Version for each execution of the tool.

INF000002: Command line arguments:

This is to display the command line arguments for each execution of the tool.

INF000003: The valid inputs are provided below.

This information occurs, if the command line option is not given.

INF000004: Opened file <filename> at <time>.

This information occurs, during opening the file.

INF000005: Error(s) and Warning(s) detected.

This information displays the number of errors and warnings.

INF000006: Execution completed successfully.

This information occurs, if the execution completed successfully.

INF000007: Execution completed successfully with warnings.

This information occurs, if the execution completed successfully with warnings.

INF000008: Execution terminated due to command line errors.

This information occurs, if the execution terminated due to command line errors.

INF000009: Execution terminated due to error in the input file.

This information occurs, if the execution terminated due to error in the input file.

INF000010: Execution terminated due to error, during the structure generation in the output file.

This information occurs, if the execution terminated during structure generation in output file.

4.9. BSWMDT File

The BSWMDT File is the template for the Basic Software Module Description. Module specific Generation Tool uses “Common Published Information” from module specific BSWMDT file. BSWMDT file should not be updated manually since it is “Static Configuration” file.

The required elements from BSWMDT File by module specific Generation Tool is as follows:

BSW-MODULE-DESCRIPTION

- MODULE-ID
- BSW-IMPLEMENTATION
- SW-VERSION
- VENDOR-ID
- AR-RELEASE-VERSION
- VENDOR-API-INFIX

In case of multiple driver support implementation, VENDOR-API-INFIX is mandatory. In case of single driver support implementation, VENDOR-API- INFIX is not used.

4.10. User Environment Settings

Edit and update user settings to SetEnv.bat file located at @ external\X1X\P1x-C\common_family\Sample_Application\ghs\SetEnv.bat

Example

Set MCAL Generator path

```
SETMCAL_GEN_EXEC=
C:\Renesas\CodeGenerator\code_generator\MCALGenerator.exe
```

Set root folder

```
 SET ROOT_FOLDER=U:
```

Set GNU Make Path

```
SET GNU_PATH="C:\Program Files (x86)\GnuWin32\bin"
```

Set Compiler install directory

```
SET COMPILER_INSTALL_DIR="C:\ghs\ comp_201517"
```

Chapter 5 Application Example

5.1. Folder Structure

Refer Section “Integration and Build Process” in the respective component User Manuals.

5.2. Compiler, Linker and Assembler

This section provides information about the details of Compiler, Linker and Assembler used for building the MCAL Driver Component.

Table 5-1 Compiler, Linker And Assembler Version Details

Details	Version
Workbench Version	Green Hills Multi V6.1.6
Compiler Version	GreenHills C V6.1.6 Compiler Version V2015.1.7
Linker Version	ELXR Version 2015.1.7
Assembler Version	EAST Version 2015.1.7

In the batch file X1X/<MICRO_VARIANT>/common_family/Sample_Application/<compiler>/SetEnv.bat set the COMPILER_INSTALL_DIR="C:\ghs\comp_201517" according to the GHS installation path.

5.2.1. Compiler

The compiler switches used for building the MCAL Driver Component using Green Hills C Compiler versions are provided.

Table 5-2 Compiler Options

Option	Meaning
-cpu=rh850g3m	Specifies a particular rh850g3m core target processor
-prepare_dispose	Instructs compiler to use the prepare and dispose instructions for function prologue and epilogue
-no_callt	Prevents the compiler from using the ‘callt’ instruction even when compiling for V850E
-sda=all	Small data memory model allows access to 64KB RAM and 64KB ROM
-reserve_r2	Reserves r2 register for use by the user
-gsize	Controls use of the gsize utility to determine the
-Osize	Enables optimization for size
-g	Generates source level debugging information
-Wundef	Enables the warning that is issued for undefined symbols in preprocessor expressions

Option	Meaning
-c	Produces object file for each source file
--short_enum	Store enumerations in the smallest possible type.
-Wshadow	Controls a warning that is issued if the declaration of a local variable *shadows* the declaration of a variable of the same name declared at the global scope, or at an outer scope.
-nofloatio	Controls the use of floating-point in stdio operations. Off (-nofloatio)
--prototype_errors	Controls the treatment of functions that are referenced or called when no prototype has been provided. Errors (--prototype_errors)
--diag_error 193	Sets the specified diagnostics message to the level of error
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-inline_prologue	Forces the compiler to use inline code sequences.This option may adversely impact the size of the generated code, so it should only be used when necessary (for example, when the routines may not exist in memory yet).
-ignore_callt_state_in_interrupts	Do not save the CTPSW and CTPC registers in interrupt routines generated by the compiler.
-large_sda	Generate 23-bit SDA relocations for load/store instructions- Increases the size of the SDA to 8 MB.
-shorten_loads	Convert 23-bit SDA relocations to 16-bit in load/store instructions when possible
-shorten_moves	Convert 32 and 48-bit move relocations to 16-bit in move instructions when possible
-delete	Controls the removal from the executable of functions that are unused and unreferenced

5.2.2. Linker

The linker switches used for building the MCAL Driver Component using Green Hills Linker are provided. The memory placements can be done using the linker file.

Table 5-3 Linker Options

Option	Meaning
Same as Compiler options	-

5.2.3. Assembler

The Assembler settings used for WDG Driver component is as follows:

Table 5-4 Assembler Options

Option	Meaning
Same as Compiler options apart -c	-

5.3. Batchfile Description

Batch file is available in the folder
X1X/<MICRO_VARIANT>/common_family/Sample_Application/<compiler>
The file is:

• SampleApp.bat

Usage:
SampleApp.bat Msn Variant Compile_Option
Msn - Module Short Name to be generated e.g. Port, Can, Adc, All..
Variant - Device variant e.g. 701372
Compile_Option - clean/build/generate.
clean for clean
build for incremental build
generate for code generation
Note: If Compile_Option is left blank, then batch will clean, generate and compile files.

5.4. Makefile Description

Makefile is available in the folder "X1X\< MICRO_VARIANT >\modules\<msn>\sample_application\make\<compiler>".
The Makefiles with the guidelines provided in AUTOSAR BSW Makefile Interface Specification, which enables easy integration with other components and the application.

The files is:

• App_<Msn>_<MICRO_VARIANT>Sample.mak
(Contains the device specific instructions).

5.4.1. App_<Msn>_<variant>_Sample.mak

```
#################################################################
# Makefile to compile and build the Sample application with the AUTOSAR
<MSN> #
# Driver Component (For Test purposes only)                    #
# Compatible with GNU Make 3.81 for Win32.                     #
#################################################################
```

```
#######################################################################
# Definitions of global environment variables                        #
#######################################################################
#Get name of the current application
CURRENT_APPL = App_<Msn>


# Get the project directory into variable "PROJECT_ROOT"

PROJECT_ROOT = $(shell cd)\..\..\..\..\..\..

COMMON_SAMPLE_CORE_PATH =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\common_platform
                    \modules\<Msn>\sample_application


# Get the current working directory into variable "SAMPLE_ROOT_PATH"
SAMPLE_ROOT_PATH =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\$(MICRO_VARIANT)\modules
<Msn>\sample_application\$(MICRO_SUB_VARIANT)

# Get the current working directory into variable "STUBS"
STUBS_PATH =
$(PROJECT_ROOT)\$(MICRO_FAMILY)
                    \common_platform\generic
                    \stubs\$(AUTOSAR_VERSION)

# Get current configuration path
<MSN>_CONFIG_PATH =
$(SAMPLE_ROOT_PATH)\$(AUTOSAR_VERSION)


# Get ARXML path
ARXML_CONFIG_PATH =  $(PROJECT_ROOT)
                     \$(MICRO_FAMILY)\$(MICRO_VARIANT)
                     \common_family\generator

# Get BSWMDT path
<MSN>_BSWMDT_CONFIG_PATH = $(PROJECT_ROOT)
                           \$(MICRO_FAMILY)
                           \$(MICRO_VARIANT)
                           \modules\<Msn>
                           \generator

# Get current configuration file path
<MSN>_CONFIG_FILE = $(MSN_CONFIG_PATH) \config
\App_<MSN>_$(MICRO_SUB_VARIANT)_
$(DEVICE_NAME)_Sample.arxml

# Path to ECUM Configuration File which is required for this module
```

```
ECUM_CONFIG_PATH = $(STUBS_PATH)\EcuM

ECUM_CONFIG_FILE = "$(ECUM_CONFIG_PATH)\xml\EcuM_Icu.arxml"
endif

# Path to BSWMDT Configuration File which is required for MSN Sample
Application

ifeq ($(AUTOSAR_VERSION), 3.2.2)
MSN_BSWMDT_CONFIG_FILE =
"$(MSN_BSWMDT_CONFIG_PATH)\R322_$(MODULE_NAME)_$(MICRO_V
ARIANT)_BSWMDT.arxml"
else
ICU_BSWMDT_CONFIG_FILE =
"$(MSN_BSWMDT_CONFIG_PATH)\R403_$(MODULE_NAME)_$(MICRO_V
ARIANT)_BSWMDT.arxml"
endif

# Version check for inter modules required
MSN_VERSION_CHECK_REQ = yes

# Database to be linked together with the current application

# Define 'no' to isolate database from the application

<MSN>_DBASE_REQ = yes


# Get the name of the SRECORD file


CURRENT_APPL_SRECORD =
$(CURRENT_APPL)_$(MICRO_SUB_VARIANT)_Sample

# Name of the database if generated separately

<MSN>_DB = <Msn>_PBcfg


#################################################################

# Final executable                                             #

#################################################################

EXE = $(CURRENT_APPL)_ MICRO_


<SUB_VARIANT>_Sample.$(EXE_FILE_SUFFIX)


LIBRARIES_TO_BUILD =


OBJECTS_LINK_ONLY =

OBJECT_OUTPUT_PATH  = $(SAMPLE_ROOT_PATH)\obj\ghs


GENERATED_SOURCE_FILES =


CC_FILES_TO_BUILD =

CPP_FILES_TO_BUILD =
ASM_FILES_TO_BUILD =

CC_INCLUDE_PATH =
CPP_INCLUDE_PATH =
ASM_INCLUDE_PATH =

PREPROCESSOR_DEFINES =

LIBRARIES_LINK_ONLY =
DIRECTORIES_TO_CREATE =
DEPEND_GCC_OPTS =

MAKE_CLEAN_RULES =
MAKE_GENERATE_RULES =
MAKE_COMPILE_RULES =
MAKE_DEBUG_RULES =
MAKE_CONFIG_RULES =
MAKE_ADD_RULES =


MAKE_DEBUG_RULES =
MAKE_ CONFIG_RULES =
MAKE_ADD_RULES =

MAKE_DEBUG_RULES += debug_base_make

STD_LIBRARY =
LNKFILE =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\$(MICRO_VARIANT)\modules\<msn
>\sample_application\make\ghs\$(CURRENT_APPL)_$(MICRO_SUB_VARIA
NT)_$(DEVICE_NAME)_Sample.ld
LNKFILE_DB =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\$(MICRO_VARIANT)\modules\<ms
n>\sample_application)\make\ghs\$(CURRENT_APPL)_$(MICRO_SUB_VARI
ANT)_$(DEVICE_NAME)_Sample_db.ld


.PHONY: MAKE_CLEAN_RULES MAKE_GENERATE_RULES
MAKE_COMPILE_RULES \
MAKE_DEBUG_RULES MAKE_CONFIG_RULES MAKE_ADD_RULES

#####################################################################
# Modules to be included in the project                             #
#####################################################################

#####################################################################
# Sample Application
```

```
#
include
$(COMMON_SAMPLE_CORE_PATH)\make\$(CURRENT_APPL)_Common_S
ample_Defs.mak
include
$(COMMON_SAMPLE_CORE_PATH)\make\$(CURRENT_APPL)_Common_S
ample_rules.mak


SAMPLE_CORE_PATH = $(SAMPLE_ROOT_PATH)
include
$(SAMPLE_CORE_PATH)\make\$(CURRENT_APPL
_$(MICRO_SUB_VARIANT)_Sample_defs.mak
include
$(SAMPLE_CORE_PATH)\make\$(CURRENT_APPL
_$(MICRO_SUB_VARIANT)_Sample_rules.mak

##################################################################

##################################################################

##################################################################

# DET Module Core Path
#
#DET_CORE_PATH = $(STUBS_PATH)\Det
#include $(DET_CORE_PATH)\make\det_defs.mak
#include $(DET_CORE_PATH)\make\det_rules.mak
##################################################################

##################################################################


# OS Module Core Path
#
OS_CORE_PATH = $(STUBS_PATH)\os
 include $(OS_CORE_PATH)\make\os_defs.mak
 include $(OS_CORE_PATH)\make\ os_rules.mak
 ##################################################################



 ##################################################################
 # ECUM Module Core Path
 #
 ECUM_CORE_PATH = $(STUBS_PATH)\EcuM
 include $(ECUM_CORE_PATH)\make\ecum_defs.mak
 include $(ECUM_CORE_PATH)\make\ecum_rules.mak
 ##################################################################

 ##################################################################
 # Scheduler Manager Module Core Path
 #
 ifeq ($(AUTOSAR_VERSION), 3.2.2)
 SCHM_CORE_PATH = $(STUBS_PATH)\SchM
 include $(SCHM_CORE_PATH)\make\schm_defs.mak
 else
 RTE_CORE_PATH = $(STUBS_PATH)\SchM
 include $(RTE_CORE_PATH)\make\rte_defs.mak
 endif
 ###############################################################

 # <MSN> Driver Component

 #

<MSN>_CORE_PATH =
 $(PROJECT_ROOT \$(MICRO_FAMILY)\  common_platform

 \modules\<msn>

 include $(<MSN>_CORE_PATH)\make\renesas_<msn>_defs.mak

 include $(<MSN>_CORE_PATH)\make\renesas _<msn>_check.mak

 include $(<MSN>_CORE_PATH)\make\renesas_<msn>_rules.mak


 ###############################################################

################################################################

# Command to generate standalone database                          #

$(MSN_DB).$(S_RECORD_SUFFIX):$(MSN_DB).$(OBJ_FILE_SUFFIX)
$(LNKFILE_DB)
@echo    *********************************************************************
@echo Building the standalone database ...
$(DBLINKER) $(LNKFILE_DB) \
"$(OBJECT_OUTPUT_PATH)\$(ICU_DB).$(OBJ_FILE_SUFFIX)" \
-map="$(OBJECT_OUTPUT_PATH)\$(ICU_DB).$(MAP_FILE_SUFFIX)" \
-o "$(OBJECT_OUTPUT_PATH)\$(MSN_DB).$(EXE_FILE_SUFFIX)"
@echo Generating Motorola S-Record file...
$(CONVERTER) $(SFLAGS)

"$(OBJECT_OUTPUT_PATH)\$(ICU_DB).$(EXE_FILE_SUFFIX)" \

-o "$(OBJECT_OUTPUT_PATH)\$(ICU_DB).$(S_RECORD_SUFFIX)"
 @echo Done ...

###############################################################
##################

$(<MSN>_DB).$(S_RECORD_SUFFIX):$(<MSN>_DB).$(OBJ_FILE_SUFFIX
) $(LNKFILE_DB)

@echo *********************************************************************

@echo Building the standalone database ...

$(DBLINKER) $(LNKFILE_DB) \

"$(OBJECT_OUTPUT_PATH)\$(<MSN>_DB).$(OBJ_FILE_SUFFIX)" \

-map="$(OBJECT_OUTPUT_PATH)\$(<MSN>_DB).$(MAP_FILE_SUFFIX)" \

-o "$(OBJECT_OUTPUT_PATH)\$(<MSN>_DB).$(EXE_FILE_SUFFIX)"

@echo Generating Motorola S-Record file...
```

```
$(CONVERTER) $(SFLAGS)
"$(OBJECT_OUTPUT_PATH)\$(<MSN>_DB).$(EXE_FILE_SUFFIX)" \
-o "$(OBJECT_OUTPUT_PATH)\$(<MSN>_DB).$(S_RECORD_SUFFIX)"
@echo Done ...


#####################################################################
# End of the Base Make script                                       #
#####################################################################
```

5.5. Integrating The <MSN> Driver Component With Other Components

This section explains the procedure to integrate the <MSN>Driver Component with other BSW components and the application.

Depending on the various configurations, the following modules are required to be integrated with the <MSN>Driver Component:

- <MSN>Interface (Folder 'Sample_Application' where the sample application for <MSN> exists. The variable '<MSN>_CORE_PATH' and the corresponding module Makefile names must be suitably changed in the base Makefile)

- Development Error Tracer (Folder 'Det' where the DET module files exist. The variable 'DET_CORE_PATH' and the corresponding module Makefile names must be suitably changed in the base Makefile)

- Scheduler Manager (Folder 'SchM' where the SCHM module exists. The variable 'RTE_CORE_PATH' and the corresponding module Makefile names must be suitably changed in the base Makefile)

- MCU Interface (Folder 'Mcu' in the give example. The variables 'MCU_CONFIG_PATH' and 'MCU_CONFIG_FILE' must be suitably changed in the module Makefile (Software_Source_Code\ssc\mak\renesas_<MSN>_rules.mak) and the base Makefile).

All the above folders are given only as examples and they have to be replaced with actual component folders. It is assumed that every component has the corresponding module Makefiles.

Apart from the above BSW components, few other folders are provided as mentioned below:

- AUTOSAR Type definition Files (Folder 'common\include', where the header files containing standard definitions that are common to all components are placed. The variable 'STUB_CORE_PATH' and the corresponding module Makefile names must be suitably changed in the base Makefile)

- RH850 specific Files (Folder 'X1X\common_platform\generic\include',where the header files that are common to all components but specific to Renesas

V850 microcontroller are placed. The variable ‘ GENERIC_PLATFORM_PATH’ and the corresponding module Makefile names must be suitably changed in the base Makefile)

Compiler specific Files (Folder ‘compiler’, where the header files that are common to all components but specific to GreenHills Compiler are placed. The variable ‘COMPILER_PATH’ and the corresponding module Makefile names must be suitably changed in the base Makefile).

5.6. Building The <MSN> Driver Component

This section explains the procedure to build the <MSN>Driver Component for any given configuration.

The <MSN> Driver Configuration Description file (.arxml) has to be given as input to the <MSN> Driver Generation Tool. The tool generates output files namely <Msn>_Lcfg.c, <Msn>_PBcfg.c, <Msn>_Cbk.h and <Msn>_Cfg.h.

Following variables must be defined in the base Makefile described in Section 5.2.1 (Makefile Description)

- PROJECT_ROOT: Root directory where the projects for all components exist.
- SPECIFIC_APPL_ROOT_PATH: Directory where the <MSN> sample application exists.
- OBJECT_OUTPUT_PATH: Directory where the module specific output files are generated.
- STARTUP_<family>_CORE_PATH: Core path for the variant specific statup files exist.
- STUB_CORE_PATH: Core path for the stub files exist.
- COMPILER_PATH: Directory where the compiler files exist.
- DEVICE_FILE_PATH: Directory where the device files exists.
- MSN_CORE_PATH: Core path for the <MSN> Driver Component folder.
- MSN_TOOL_PATH: Directory where the module specific tool exe exist.
- CC_INCLUDE_PATH: Path variable where all the header files can be found by the compiler.
- CC_FILES_TO_BUILD: Variable that contains the list of source files, to be compiled and linked.
- <MSN>_clean_generated_files: This target can be used to clean the configuration source and header files generated by the <MSN> Driver Generation Tool.
- debug_<MSN>_makefile: This target can be used to print the debug information such as paths used by <MSN> Driver Component.
- generate_<MSN>_config: This target can be used to invoke the <MSN> Driver Generation Tool which in turn takes the ECU Configuration Description files (App_<MSN>_<DEVICE_NAME>_Sample.arxml) as an input and generates the configuration source and header files.

Following variables must be defined in the Module Makefile described in Section 5.2.1 (Makefile Description):

- PROJECT_ROOT: Root directory where the projects for all components exist.
- MSN_CONFIG_PATH: Configuration path for description file of the <MSN> Driver Component.
- MSN_CONFIG_FILE: Name of the <MSN> Driver Component description file.
- STUB_CONFIG_PATH: Configuration path for description file of the stub.
- MCU_CONFIG_FILE: Name of the MCU Driver Component description file.
- ARXML_CONFIG_PATH: ARXML Configuration file path used for the <MSN> Driver Component.
- ARXML_CONFIG_FILE: ARXML Configuration file used for the <MSN> Driver Component.
- BSWMDT_CONFIG_PATH: Path for <MSN> BSWMDT file.
- BSWMDT_CONFIG_FILE: Name of the <MSN> BSWMDT file.
- GENERIC_STUB_PATH: Directory where the generic stub exist.
- GENERIC_PLATFORM_PATH: Directory where the generic platform files exist.
- CC_INCLUDE_PATH: Path variable where all the header files can be found by the compiler.
- CC_FILES_TO_BUILD: Variable that contains the list of source files, to be compiled and linked.
- <MSN>_DB: Name of the Post-build configuration file.

The above mentioned variables must be used by the user to build the base Makefile.

A sample base Makefile (App_<MSN>_ <MICRO_SUB_VARIANT> _Device_Sample.mak) has been provided with the product for reference. This file can be modified to suit the developer's needs.

The targets that are supported in the base Makefile enable the user in build and cleanup activities during/after the build process. They are listed below:

5.6.1. Targets Supported By The Sample Base Makefile

5.6.1.1. debug_base_make

Invoking the Make utility and passing "debug_base_make" as a parameter prints all the variables that are used in the base Makefile. This can be used to print various paths and file names to see if they are correct.

5.6.1.2. clean_objs

Invoking the Make utility and passing "clean_objs" as a parameter deletes all the object files from the output folder ("X1X\<MICRO_VARIANT>\modules\<msn>\Sample_application\< MICRO_SUB_VARIANT >\obj" in this case).

5.6.1.3. clean

Invoking the Make utility and passing “clean” as a parameter deletes tool generated files in the configuration output folders (“X1X\<MICRO_VARIANT>\modules\<msn>\sample_application\<MICRO_SUB_VARIANT>\src” and

“X1X\<MICRO_VARIANT>\modules\<msn>\Sample_application\<MICRO_SUB_VARIANT>\include”in this case)

.

5.6.1.4. clean_all

Invoking the Make utility and passing “clean_all” as a parameter deletes all files such as object file, list files and map files from the output folder (“X1X\< MICRO_VARIANT >
\modules\<msn>\sample_application\< MICRO_SUB_VARIANT >\obj\<compiler>” in this case).

5.6.1.5. generate_<msn>_config

Invoking the Make utility and passing “generate_<MSN>_config” as a parameter invokes the <MSN> Driver Generation Tool. The tool takes the ECU Configuration Description File(s) (“X1X\< MICRO_VARIANT >\modules\<msn>\Sample_application\<MICRO_SUB_VARIANT>\ AUTOSAR_VERSION

config\<MSN>_Sample_ <MICRO_SUB_VARIANT>\.arxml” as input and generates the output files in folders

“X1X\< MICRO_VARIANT >\modules\<msn>\Sample_application\< MICRO_SUB_VARIANT>\ AUTOSAR_VERSION \src” and

“X1X\< MICRO_VARIANT >1\modules\<msn>\Sample_application\< MICRO_SUB_VARIANT>\ AUTOSAR_VERSION \include”).

5.6.1.6. App_<MSN>_< MICRO_SUB_VARIANT >_Sample.out

Invoking the Make utility and passing “Sample.out” as a parameter invokes the compiler and linker sequentially. Then it generates the executable “App_<MSN>_<MICRO_SUB_VARIANT>_Sample.out”.

5.6.1.7. <Msn>_PBcfg.s37

Invoking the Make utility and passing “<Msn>_PBcfg.s37” as a parameter invokes
the compiler and linker sequentially and generates the Motorola S-Record file “<Msn>_PBcfg.s37” in the output folder.
This scenario typically arises when post-build parameters are modified and only the database needs to be flashed into the device without disturbing the other ROM contents.

Chapter 6 Support For Different Interrupt Categories

The <MSN> Driver supports CAT1 and CAT2 interrupt categories:

CAT1

In CAT1 type of interrupt ISR does not use an operating system service. In practice, the OS does not handle these interrupts, and the interrupt handler is implemented in the driver code, with the only restriction that OS services cannot be called. Typically, these are the fastest highest priority interrupts.

CAT2

In CAT2 type of interrupt wherein the ISR is handled by the system and OS calls can be called from the handler.

For CAT1 and CAT2, the selection of interrupt category is for each interrupt in the module. Individual MCAL module does not contain any option for interrupt category configuration. The user has to configure the ISR category in OS and also to use the right MCAL specified name and MCAL expects "ISR(INTERRUPT_NAME)" keyword defined in OS in case of CAT2.

Notes

1. The understanding is Os module does not publish short name handles for CAT1 OsIsr container. But it should be defined in the interrupt vector table by the OS.
2. The understanding is that Os module should publish short name handles for CAT2 OsIsr container according to ecuc_sws_2108 requirement by adding the Os_" pefix to the configured interrupt name.

Reference between the <MSN> module and OS:

<Msn> module's <Module>_Irq.c/h files include "Os.h" header file to obtain the interrupt category information configured in the OS. Therefore following pre-processor definitions are expected to be published in Os.h file by the OS in case of CAT2 or to be used in the interrupt vector table in case of CAT1.

Table 6-1 CAT1 and CAT2 Naming Convention

Interrupt Category	Naming Convention
CAT1	<MCAL_INTERRUPT_NAME>_ ISR
CAT2	<MCAL_INTERRUPT_NAME>_CAT2_ISR
CAT2 (In case the handles of the OsIsr container are generated without 'Os_' prefix by Os generation tool)	Os_<MCAL_INTERRUPT_NAME>_CAT2_ISR

MCAL in Stand Alone:

In case if the MCAL modules are to be used stand alone without having standard Autosar Os module, the user has to prepare an Os.h stub file with the published handles only for those interrupt names which are to be used as CAT2.

Table 6-2 List of ISR Names that need to be configured and published in Os.h (CAT2) or used in the interrupt vector table (CAT1) for <MSN> Driver

Sl. No.	CAT1	CAT2	CAT2(In case the handles of the OsIsr container are generated without 'Os_' prefix by Os generation tool)
1	<MSN>n_SGm_ISR	<MSN>n_SGm_CAT2_ISR	Os_<MSN>n_SGm_CAT2_ISR
2	<MSN>_DMA_CHxy_ISR	<MSN>_DMA_CHxy_CAT2_ISR	Os_<MSN>_DMA_CHxy_CAT2_ISR

Where

'n' indicate HW Unit number

'm' indicate SG Unit number

'xy' indicate DMA channel Id number

Chapter 7 GNU MAKE Environment

Every component is delivered with the necessary Make scripts that are required to integrate the component with the application. The scripts are compatible with GNU Make version 3.81.

All the delivered Makefiles have to be included in the project level base Makefile in order to build the component together with the application. Refer section “**Integration and Build Process**” of the respective component User Manuals for more information on the Makefile variables and their usage.

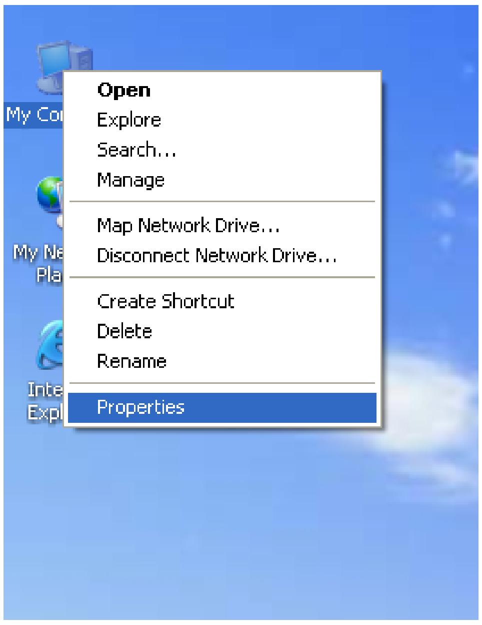
7.1. Build Process With GNUMAKE

When the batch file of certain application is built, the GNU Make utility will be searched by batch file. The GNU Make utility should be present in the default path specified by GNUMAKE\PATH variable. By making use of the GNU Make utility the batch file will be compiled.

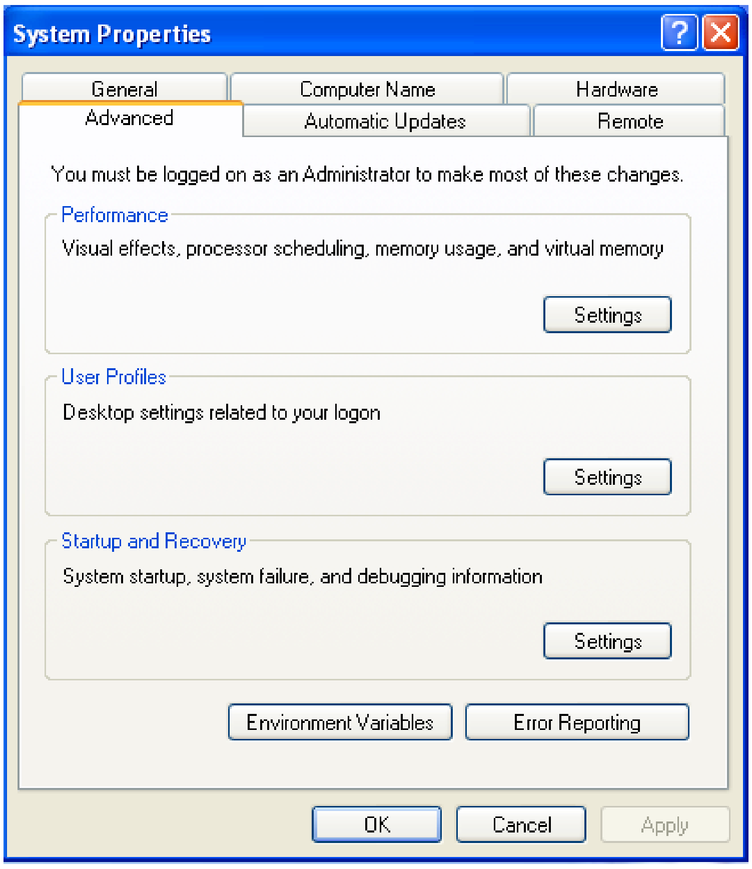
7.2. Build Process Without GNUMAKE

If GNU Make utility is not present at the default path or present in some other directory the following procedure is followed to set the Environmental variable GNUMAKE\PATH.

1. Right click on “My Computer” select properties, user will find System Properties.



2. In System Properties select “Advanced” option, user will find “Environmental Variables” at the bottom side of window.



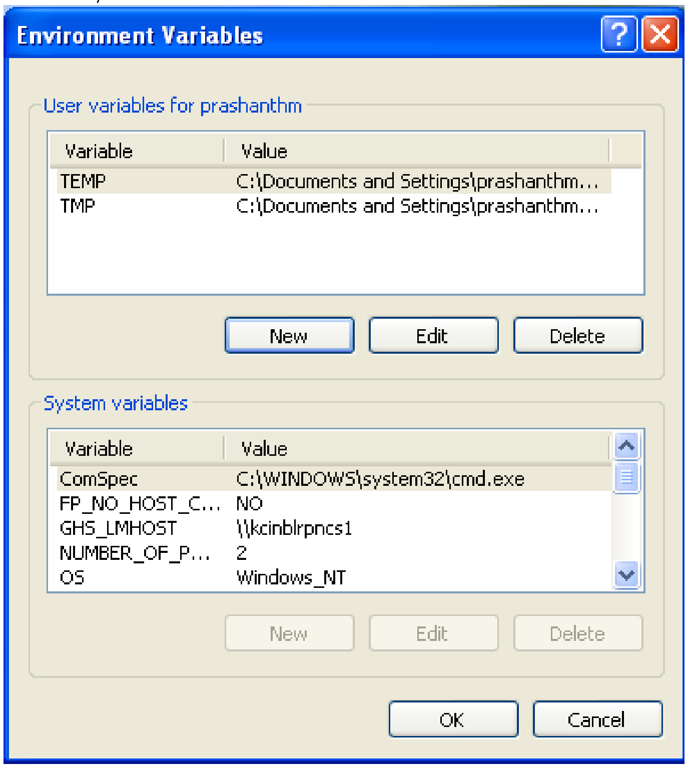
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

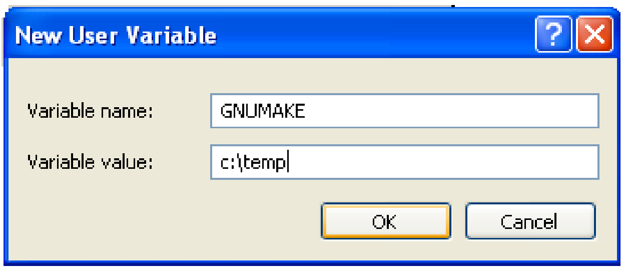
3. Click on “Environmental Variables”, user will find “Environment Variables” window in that, select “New”.



Variable	Value
TEMP	C:\Documents and Settings\prashanthm...
TMP	C:\Documents and Settings\prashanthm...

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
GHS_LMHOST	\\kcinblrpncs1
NUMBER_OF_P...	2
OS	Windows_NT

4. After step 3, user can find “New User Variable” window with “Variable name” and “Variable path” options which needs to be set, Variable name will be set as GNUMAKE and Variable path is the path of the directory where GNU Make utility is present and click ok.



Variable name: GNUMAKE
Variable value: c:\temp

5. After step 4, in “System Properties” window click “Apply” and then “Ok”.

Remark GNU Make utility version 3.81 must be separately downloaded and installed to use the Makefiles delivered along with the component. More information on the utility can be found at http://www.gnu.org/

Chapter 8 Load Binaries

Once the Executable or S-Record is generated using the project level base Makefile, it needs to be downloaded into the target using a Flash programmer.

The user has to read the instructions provided in the Flash programmer's User Manual thoroughly before using it.

Chapter 9 Appendix

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	15-Sep-2014
2.	The following changes are made : 1. Added section 5.2. Compiler, Linker and Assembler in Chapter 5. 2. Added section 5.3. Batch file Description in Chapter 5. 3. Supported device name changed throughout the document. 4. Added R-number to the document.	1.0.1	25-Apr-2016
3	The following changes are made: 1. Added Section 4.10 – User Environment settings. 2. Removed Description of Translation XML file from Chapter 9 3. Updated Copyright year 4. Added example for reference in section 4.4 5. In Chapter 4 Figure 4-2 has been updated as per version change 6. Compiler Option Table updated 7. In Chapter 5 Section 5.4.1 Trxml changed to Arxml format 8. Chapter 9 Section 9.1 Configuration XML file remove since not relevant 9. Document name corrected in second last and last page. 10. Compiler options updated in the section 5.2	1.0.2	17-Feb-2017
4	The following changes are made: 1. Updated R-number of the document. 2. Notice and copyright are updated.	1.0.3	09-May-2017

Getting Started Document for P1x-C MCAL Driver User' Manual Version 1.0.3

Publication Date: Rev.1.01, May 09, 2017

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Getting Started Document for P1x-C MCAL Driver

User's Manual

Renesas Electronics Corporation

R20UT3828EJ0101