User's Manual

Getting Started Document for X1x MCAL Driver

User's Manual

Version 1.0.5

Target Device:
RH850/X1x

Renesas Electronics
www.renesas.com

Rev.0.01 Aug 2014

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

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>	F1x, R1x, P1x, E1x etc.
<MICRO_SUB_VARIANT>	F1L, R1L, P1L, E1L, E1MS etc.
AUTOSAR_VERSION	3.2.2 or 4.0.3
DEVICE_NAME	Example :701205EAFP

Definitions

Terminology	Description
.xml	XML File.
.one	Project Settings file.
.arxml	AUTOSAR XML File.
.trxml	Translation 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.
Translation XML File	Translation XML File is in XML format which contains translation and device specific header file path.
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

The document describes the information about installation, usage of ECU Configuration Editor (ECU Spectrum), Generation Tool and references to Sections in the Component User Manuals that the user needs to refer to build 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.

Generation Tool is a command line tool that accepts ECU Configuration Description File(s), BSWMDT File, Translation XML 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:

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).

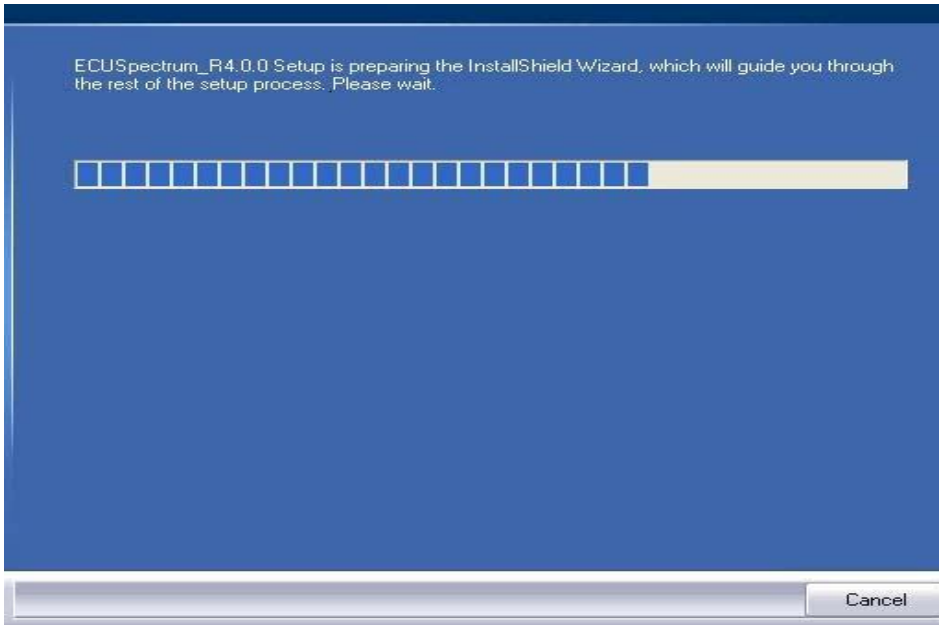


Figure 2-3 ECU Spectrum installation step 1

Step 2:

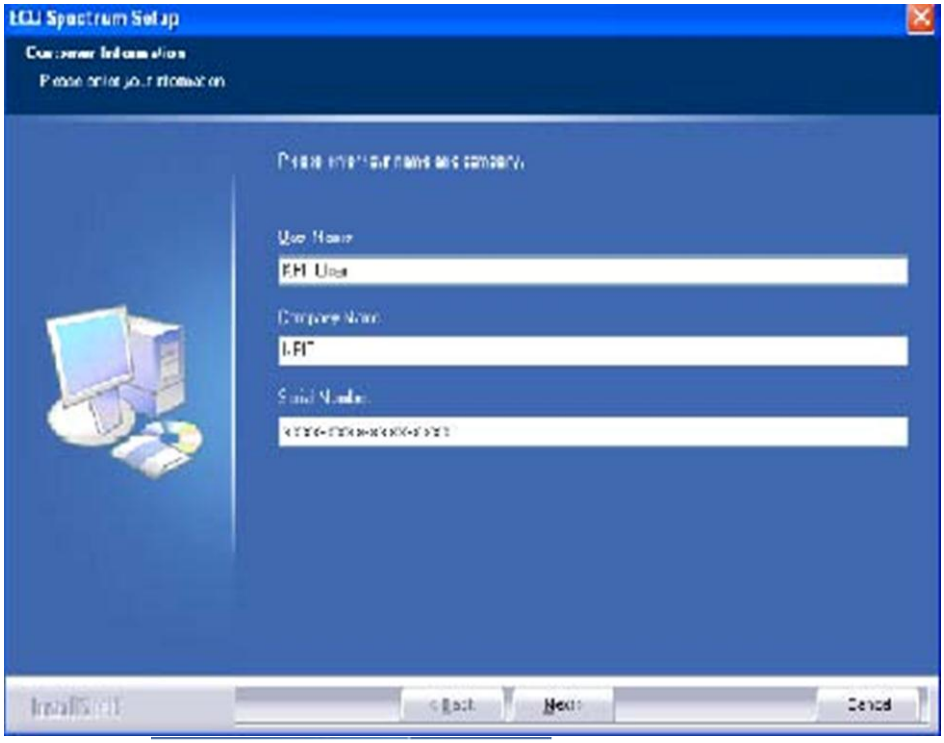


Figure 2-4 ECU Spectrum installation step 2

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:

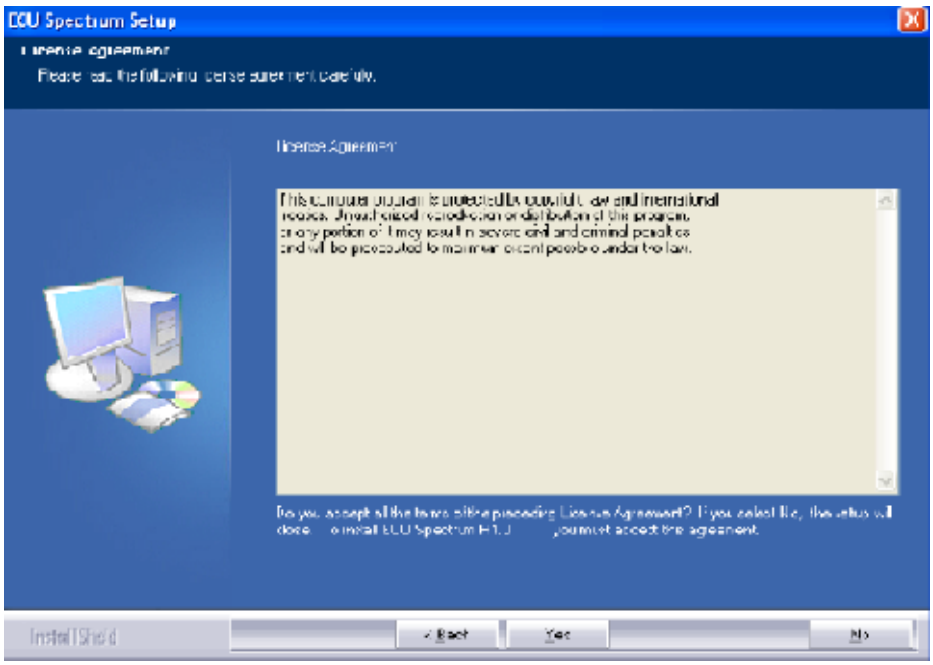


Figure 2-5 ECU Spectrum installation step 3

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:

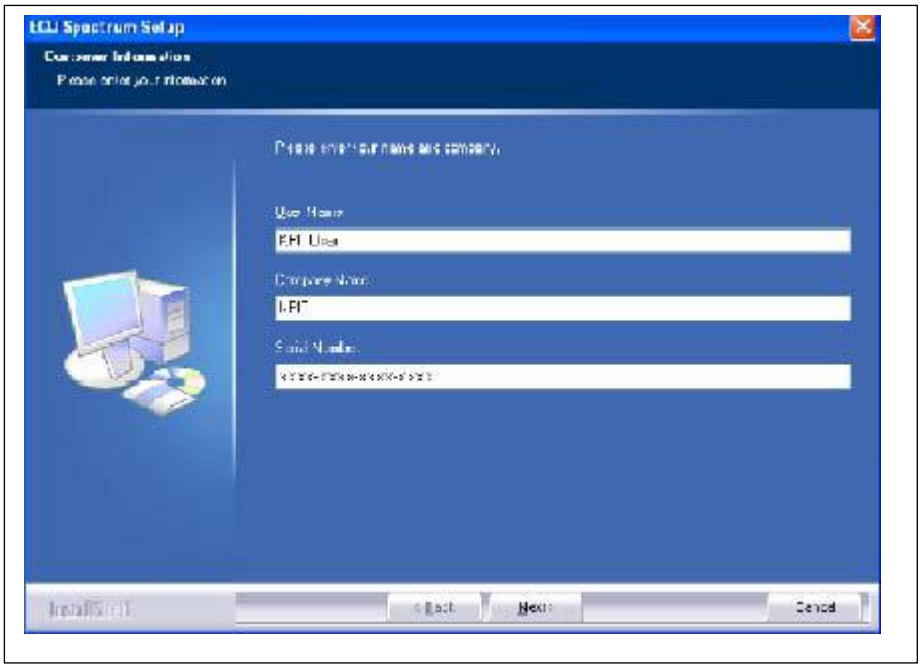


Figure 2-6 ECU Spectrum installation step 4

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)

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:

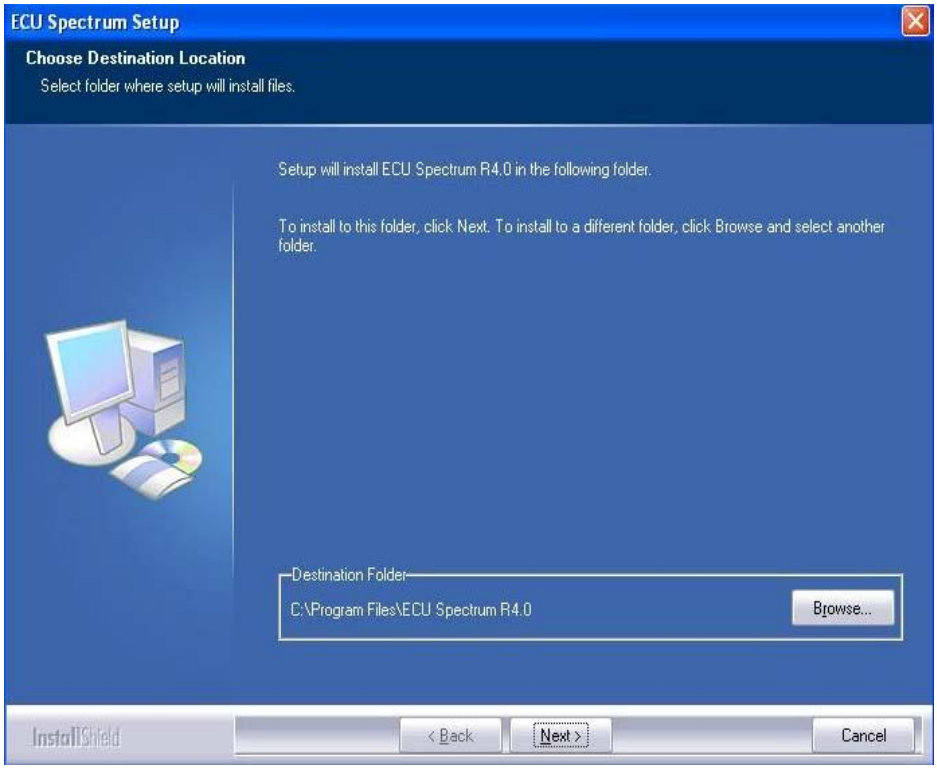


Figure 2-8 ECU Spectrum installation step 6

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:

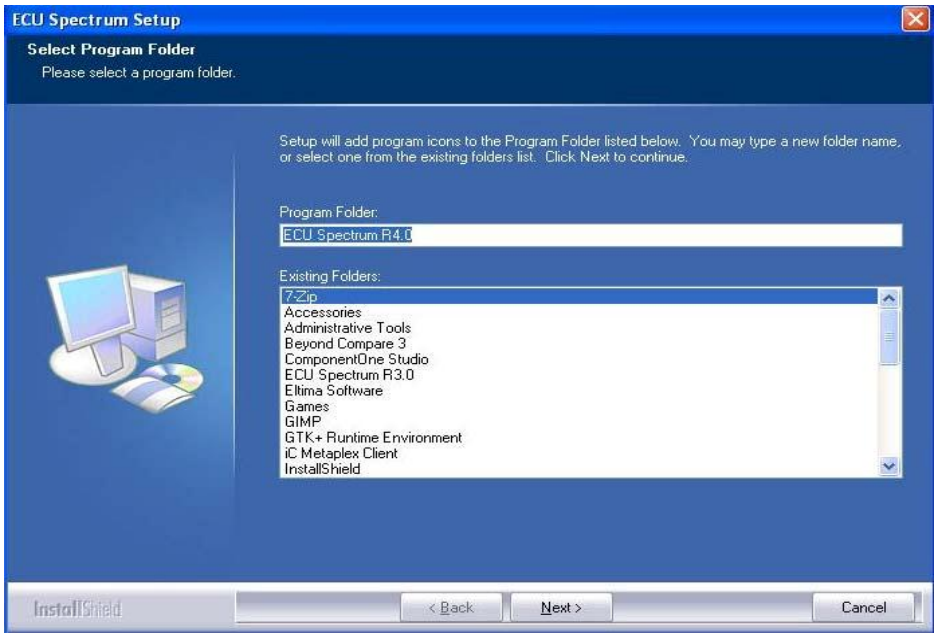


Figure 2-9 ECU Spectrum installation step 7

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:

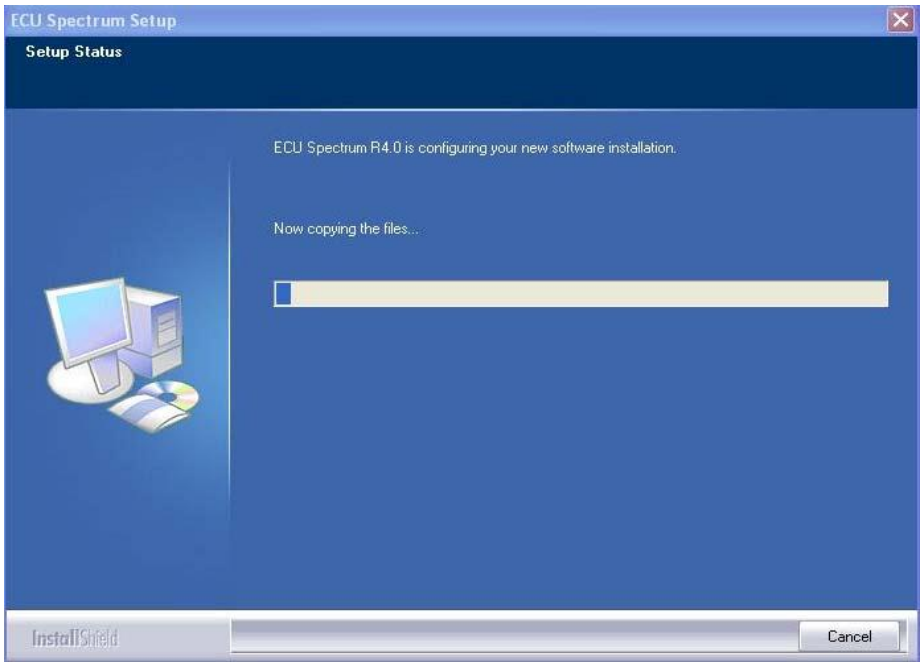


Figure 2-10 ECU Spectrum installation step 8

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:

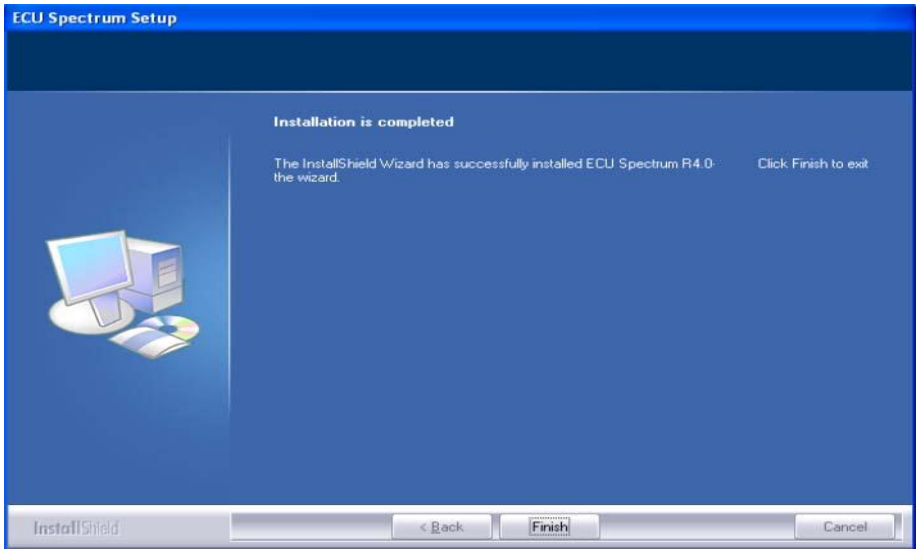


Figure 2-11 ECU Spectrum installation step 9

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 Generation Tool executable for the individual components that takes ECU Configuration Description File as input and generates the required ‘C’ source and header files.

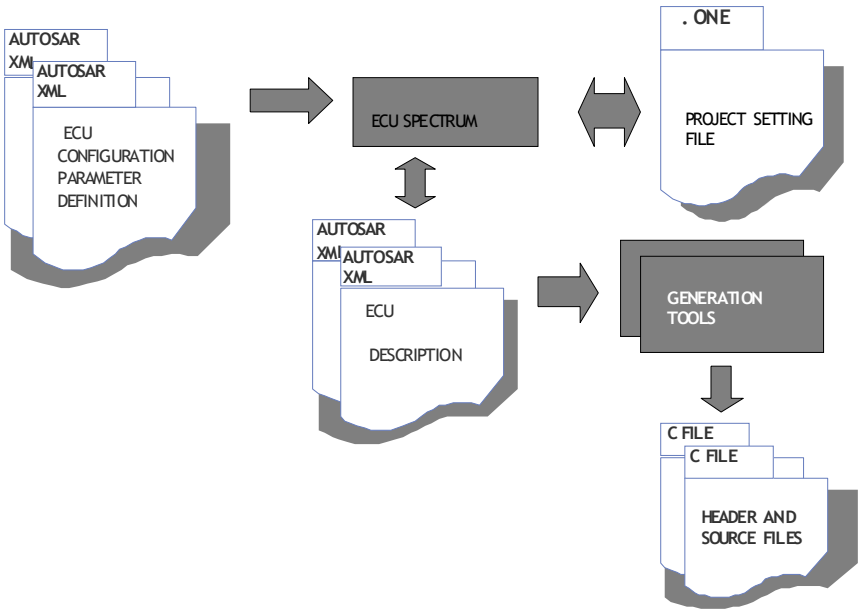


Figure 2-12 ECU Spectrum Overview

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.

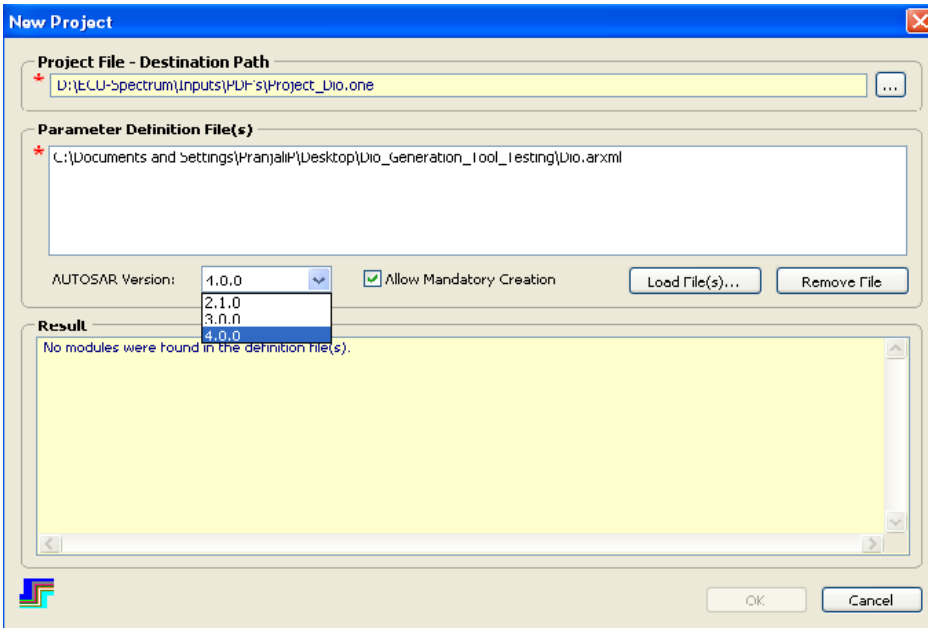


Figure 3-1 Creating New Project

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.

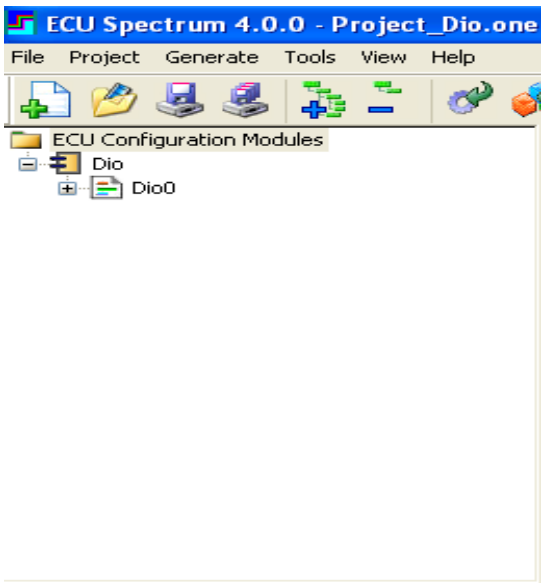


Figure 3-2 Adding New Module Instance

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

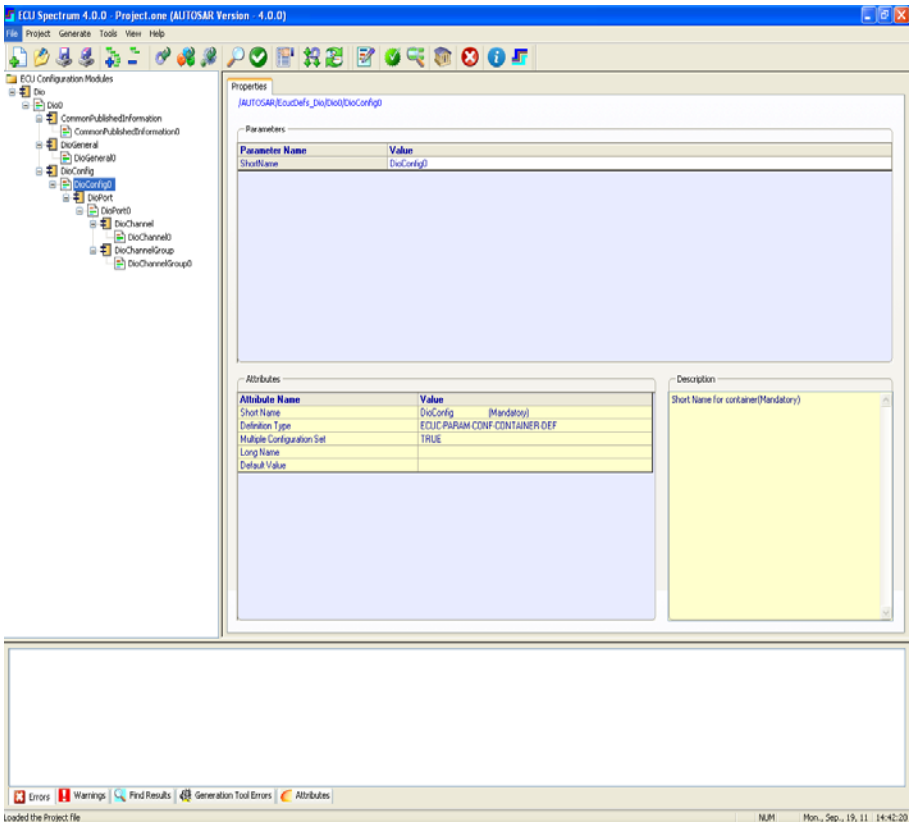


Figure 3-3 GUI for Configuration

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:

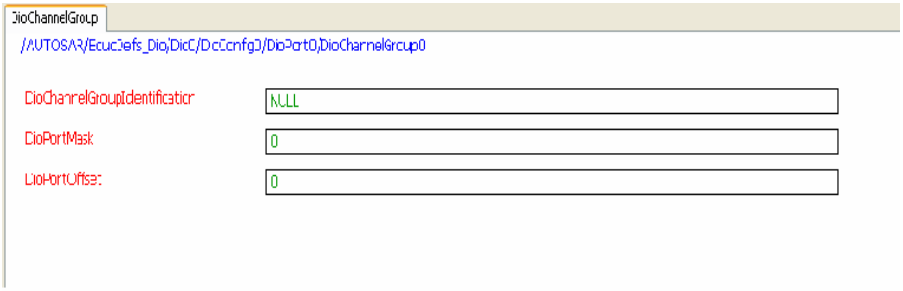


Figure 3-4 Parameter Configuration

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 gets displayed at the top of the ‘Right Configuration View’ as shown in the following figure. On selecting the container, Parameters and sub-containers gets displayed in the grid control as shown in the following figure.

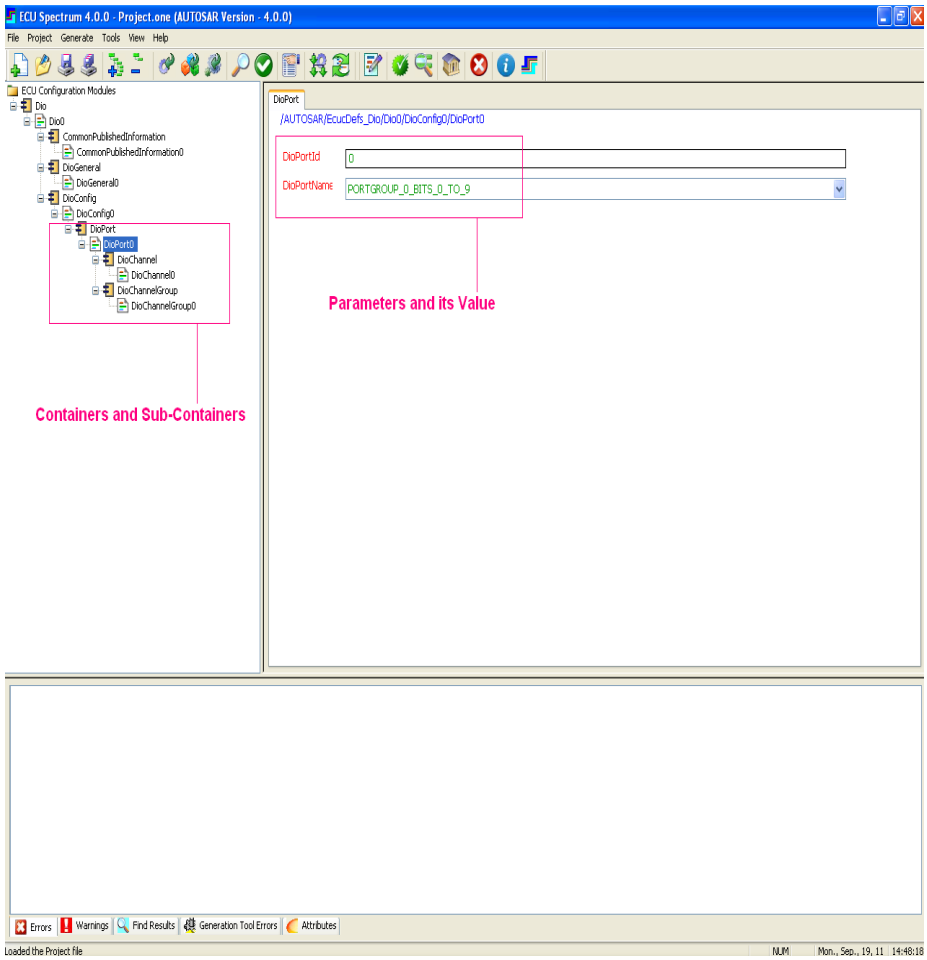


Figure 3-5 Distinguish Between Containers

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.

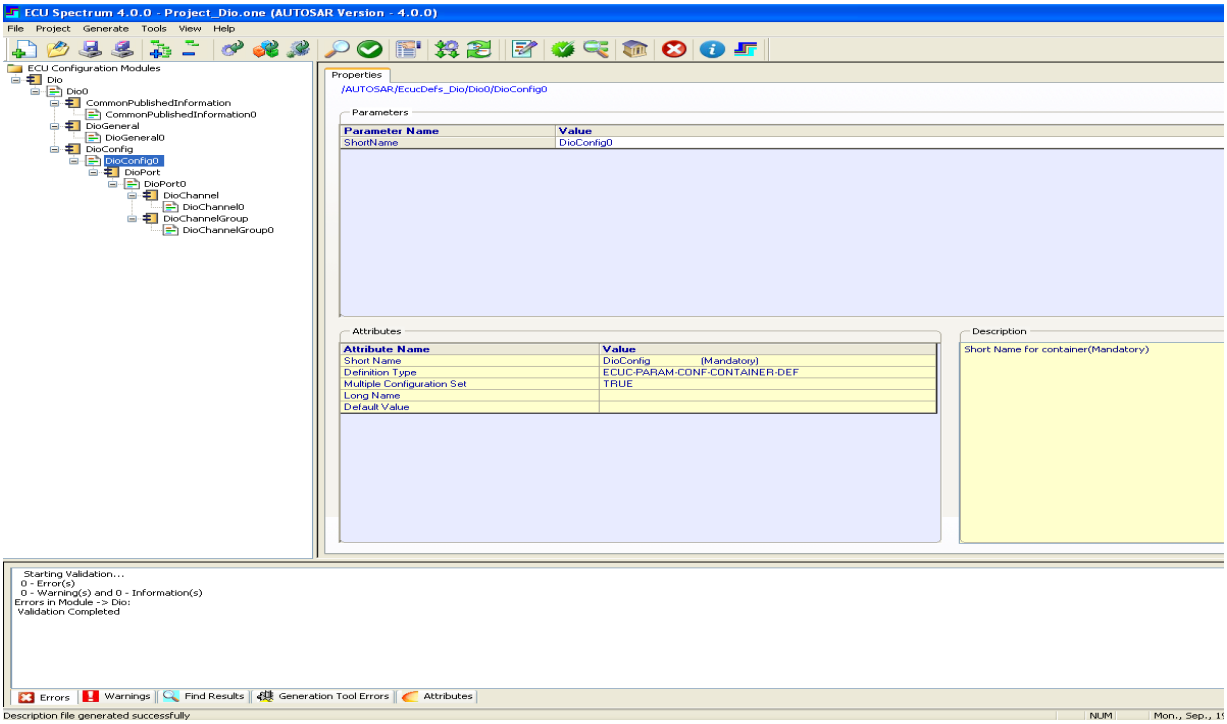


Figure 3-6 Validation

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

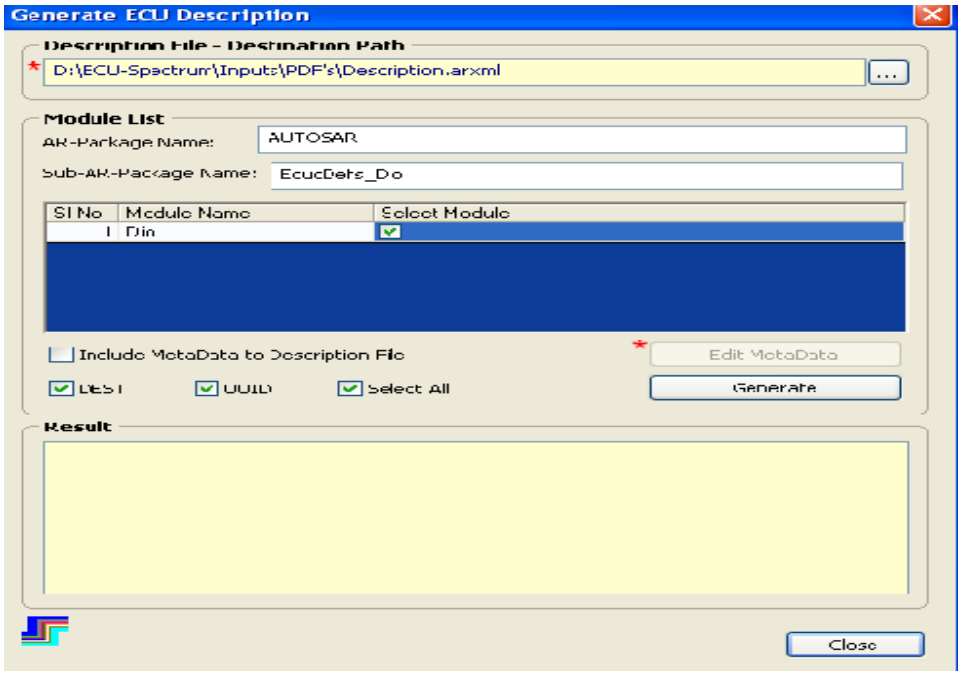


Figure 3-7 Description File Generation

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

Generation Tool is a command line tool that provides scalability and configurability for the component. It accepts ECU Configuration Description File(s), BSWMDT File, Translation XML File and Configuration XML File as input and generates the C Header and C Source files. However Configuration XML File is optional.

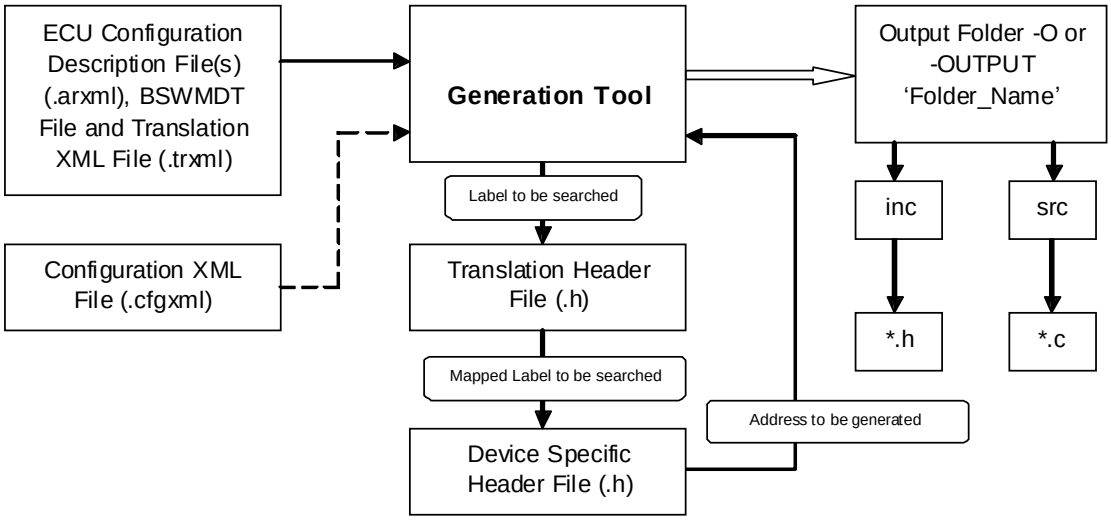


Figure 4-1 Generation Tool Overview

4.1. Translation XML File

Generation Tool accepts ECU Configuration Description File(s) (.arxml), BSWMDT File (.arxml) and Translation XML File (.trxml) as an input. Translation XML File is in XML format which contains translation and device specific header file path. For the syntax of the contents of Translation XML File, please refer the Chapter 8 *Appendix*.

If mapped device specific address label is changed/updated then only respective mapping in Translation Header File needs to be updated. In this case there will not be any impact on Generation Tool for the generation of address in tool generated output file(s).

4.1.1. Translation Header File

This file is look-up table (mapping in the form of definitions) for the device specific address labels. Based on the configuration in ECU Configuration Description File, the mapped device specific address labels will be searched in Device Specific Header File by Generation Tool to generate associated address in tool generated output file(s). For the Translation Header File path, the value of '<Msn>DeviceName' parameter from the container '<Msn>General' container should be present as child tag of TRANSLATION-FILE-PATH in Translation XML File. Both 'Absolute' and 'Relative' paths are supported by generation tool however default path is 'Relative' path.

E.g.

<TRANSLATION-FILE-PATH>

<Value_Of_MsnDeviceName>..\DF_Timer.h ..\DF_Timer_ISR.h</ Value_Of_MsnDeviceName>

</TRANSLATION-FILE-PATH>

4.1.2. Device Specific Header File

This file contains device specific labels and associated address. Based on the configuration in ECU Configuration Description File, the mapped device specific address labels will be used to generate associated address in tool generated output file(s). For the Device Specific Header File path, the value of '<Msn>DeviceName' parameter from the container '<Msn>General' container should be present as child tag of DEVICE-FILE-PATH in Translation XML File. Both 'Absolute' and 'Relative' paths are supported by generation tool however default path is 'Relative' path.

If multiple Device Specific Header Files need to be provided for the same device (value of '<Msn>DeviceName' parameter) in Translation XML File, then each Device Specific Header File path should be separated with 'space'.

E.g.

<DEVICE-FILE-PATH>

<Value_Of_MsnDeviceName>..\DF_Timer.h ..\DF_Timer_ISR.h</ Value_Of_MsnDeviceName>

</DEVICE-FILE-PATH>

Remark Generation Tool will searches the mapped labels in Device Specific Header File by using Translation Header File for the respective address generation in tool generated output file(s).

4.2. Configuration XML File

Configuration XML File is in XML format which contains command line options and input/output path. For the syntax of the contents of Configuration XML File, please refer the Chapter 8 *Appendix*.

E.g.

<LOG>ON/OFF</LOG>

<HELP>ON/OFF</HELP>

4.3. 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.

{<Component_Name>_X1x.exe} <Options> <Input Filename(s)>

Where,

<Component_Name>_X1x.exe: Name of the Generation Tool Executable

Options: [-H/-Help -C/-Config -O/-Output -Osrc -Oinc -L/-Log -D/-Dryrun]

Input Filename(s): {ECU Configuration Description File(s), BSWMDT File and Translation XML File [optional]}

Notations:

{data} represents compulsory data

<data> represents the actual data that will be specified on command line during tool usage.

[data] represents optional data.

Table 4-1 Options and Description

Option	Description
-H/-Help	To display help regarding usage of the tool. Gets the highest priority when used with other options.
-C/-Config	To execute tool with the options provided in the Configuration XML File. Command line options get the higher priority than the options provided in Configuration XML File.

Table 4-1 Options and Description

Option	Description
-O/-Output	By default, the tool generates output files in the '<Component_Name>_Output' folder in the path where executable is present. The user can use the -O option followed by the folder name, to generate the output files in the specified folder. Either absolute path or relative path can be provided to specify the folder name. The C Source and C Header files are generated in the sub folders 'src' and 'inc' within the output folder.
-Osrc	The user can use the -Osrc option followed by the folder name, to generate the C Source files in the specified folder.
-Oinc	The user can use the -Oinc option followed by the folder name, to generate the C Header files in the specified folder.
–L/-Log	To log the output to the <Component_Name>.log file.
-D/-Dryrun	To execute tool in validation mode. The tool will not generate output files even though the input file provided is error free.

Remark

- If Translation XML File is not provided on the command line then '<Component_Name>_X1x.trxml' which is present in the same location of '<Component_Name>_X1x.exe' is considered as 'default' Translation XML File by the Generation Tool.
- If Configuration XML File is not provided on the command line then '<Component_Name>_X1x.cfgxml' which is present in the same location of '<Component_Name>_X1x.exe' is considered as 'default' Configuration XML File by the Generation Tool.
- The Generation Tool should not be executed more than five times in parallel

4.4. Sample Usage

Sample usage of the generation tool is given below. "<Msn>_X1x.exe" is taken as example. Similar usage is applicable for other MCAL Generation Tools.

<Msn>_X1x

<Msn> Driver Generation Tool usage is displayed on the terminal. Generation Tool accepts Configuration XML File as default and performs the execution, based on the settings provided in Configuration XML File.

<Msn>_X1x -H

Displays <MSN> Driver Generation Tool help information on the terminal, where <MSN> Driver Generation Tool executable is present.

<Msn>_X1x -L -O output Sample.arxml BSWMDT.arxml

Generation Tool logs the output to the <Msn>.log file. <Msn>_PBcfg.c file is generated in 'src' folder. <Msn>_Cfg.h file is generated in 'include' folder.

<Msn>_X1x -D Sample.arxml BswMd.arxml

Generation Tool validates an input file and displays error/warning/information messages if any on the command line. Output files are not generated since –D option is provided in the command line.

<Msn>_X1x -O output Sample.arxml BswMd.arxml

Output files are generated in folder “output”. <Msn>_PBcfg.c is generated in ‘src’ folder. <Msn>_Cfg.h file is generated in ‘inc’ folder.

<Msn>_X1x C:\Input\Sample.arxml C:\Input\BswMd.arxml -O output

Generation Tool accepts input file (Sample.arxml) from absolute directory path “C:\Input”. Output files are generated in folder “output”. <Msn>_PBcfg.c is generated in ‘src’ folder. <Msn>_Cfg.h file is generated in ‘inc’ folder.

<Msn>_X1x Sample.arxml BswMd.arxml -O C:\Output

Output files are generated in folder “C:\Output”. <Msn>_PBcfg.c is generated in ‘src’ folder. <Msn>_Cfg.h file is generated in ‘inc’ folder.

<Msn>_X1x Sample.arxml BswMd.arxml Sample.trxml

Generation Tool accepts ECU Configuration Description File (Sample.arxml), BSWMDT File (BswMd.arxml) and Translation XML File (Sample.trxml) from the current working directory. Output files are generated in the default folder “<Msn>_Output”, since –O option is not provided in the command line. <Msn>_PBcfg.c is generated in ‘src’ folder. <Msn>_Cfg.h file is generated in ‘inc’ folder.

<Msn>_X1x -C Sample.cfgxml

Generation Tool accepts ECU Configuration Description File (Sample.arxml), BSWMDT File (BswMd.arxml) and Configuration XML File (Sample.cfgxml) from the current working directory. Tool accepts options provided in the Configuration XML File. If Configuration XML File name is not provided as input with -C option, Generation Tool errors out.

Remark

- If Translation XML File is not provided on the command line, <Msn>_X1x.exe considers <Msn>_X1x.trxml as ‘default’ Translation XML File from the same directory where the tool is located.
- If Configuration XML File is not provided on the command line, <Msn>_X1x.exe considers <Msn>_X1x.cfgxml as ‘default’ Configuration XML File from the same directory where the tool is located.
- If any filename/directory name related argument on the command line contain the ‘space’, then the same argument on the command line should be provided in double quotes “” as followed by standard command line feature. E.g. if file name is ‘Sample Description.arxml’, then on the command line the same name should be provided in double quotes “” as “Sample Description.arxml”.
- The ‘include’ and ‘src’ directories are generated inside the output directory provided on the command line or in the default output directory

<Msn>_Output.\

- BSWMDT file should not be updated manually since it is “Static Configuration” file.

4.5. Tool Installation Requirements

The minimum hardware and software requirements for proper installation of Module Specific Generation Tool are 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 Module Specific 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

Copy the Module Specific Generation Tool executable file to the local hard disk.

Run the executable with -H option to get help on usage of the tool.

<Msn>_X1x.exe -H

This command generates usage of Module Specific Driver Generation Tool on the command line.

4.7. Tool Un-Installation

There is no specific method for un-installing the Module specific Generation Tool. To un-install, delete the Module specific Generation Tool executable from the existing directory.

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 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 will occur, 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 will occur, 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.

ERR000013: <File name> should contain the TRANSLATION-FILE-PATH' and 'DEVICE-FILE-PATH' tags.

This error will occur, if the translation <File name> doesn't have 'TRANSLATION-FILE-PATH' and 'DEVICE-FILE-PATH' tags.

ERR000014: 'TRANSLATION-FILE-PATH' tag in <File name> is empty.

This error will occur, if the translation <File name> doesn't have 'TRANSLATION-FILE-PATH' tags.

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

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

ERR000019: Cannot open <File name>.

This error will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, 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 will occur, if the command line option is not given.

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

This information will occur, during opening the file.

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

This information will display the number of errors and warnings.

INF000006: Execution completed successfully.

This information will occur, if the execution completed successfully.

INF000007: Execution completed successfully with warnings.

This information will occur, if the execution completed successfully with warnings.

INF000008: Execution terminated due to command line errors.

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

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

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

4.9. R3.2.2 Messages

This section contains the list of error/warning/information messages which is specific to AUTOSAR Renesas R3.2.2 X1x MCAL Driver module that will be generated by the Generation Tool.

4.9.1. Error Messages

ERR000030: The 'parameter tag name' tag should be configured in BSWMDT File.

This error will occur, if any of the configuration parameter(s) mentioned below is (are) not configured in BSWMDT File.

The list of mandatory parameters with respect to container is listed below:

Table 4-2 Mandatory Parameters

Container	Parameters
BswImplementation	SW-MAJOR-VERSION
	SW-MINOR-VERSION
	SW-PATCH-VERSION
	AR-MAJOR-VERSION
	AR-MINOR-VERSION
	AR-PATCH-VERSION
	VendorApiInfix
BswModuleDescription	ModuleId

Note: VendorApiInfix parameter is mandatory for only some modules.

4.9.2. Warning Messages

None.

4.9.3. Information Messages

None.

4.10. R4.0.3 Messages

This section contains the list of error/warning/information messages which is specific to AUTOSAR Renesas R4.0.3 X1x MCAL Driver module that will be generated by the Generation Tool.

4.10.1. Error Messages

ERR000030: The 'parameter tag name' tag should be configured in BSWMDT File.

This error will occur, if any of the configuration parameter(s) mentioned below is (are) not configured in BSWMDT File.

The list of mandatory parameters with respect to container is listed below:

Table 4-3 Mandatory Parameters

Container	Parameters
BswImplementation	SwVersion
	VendorId
	ArReleaseVersion
	VendorApiInfix
BswModuleDescription	ModuleId

Note: VendorApiInfix parameter is mandatory for only some modules.

4.10.2. Warning Messages

None.

4.10.3. Information Messages

None.

4.11. 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 are 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.

Chapter 5 Application Example

5.1. Folder Structure

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

5.2. Makefile Description

Makefile is available in the folder “X1X\< MICRO_VARIANT >\modules\<msn>\sample_application\< MICRO_SUB_VARIANT >\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:

• A p p _<Msn>_<MICRO_SUB_VARIANT>_<DEVICE_NAME>Sample.mak (Contains the device specific instructions).

5.2.1. 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 TRXML path
TRXML_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 TRXML Configuration File which is required for this module
TRXML_CONFIG_FILE =
"$(TRXML_CONFIG_PATH)\Sample_Application_$(MICRO
_VARIANT).trxml""

# Path to ECUM Configuration File which is required for this module
ECUM_CONFIG_PATH = $(STUBS_PATH)\EcuM

ifeq ($(AUTOSAR_VERSION), 3.2.2)
ECUM_CONFIG_FILE = "$(ECUM_CONFIG_PATH)\xml\EcuM.arxml"
else
```

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

# Path to TRXML Configuration File which is required for Test Application
TRXML_CONFIG_FILE =
"$(TRXML_CONFIG_PATH)\Sample_Application_$(MICRO_VARIANT).trxml"

# 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 =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\$(MICRO_VARIANT)\modules\<msn
>\sample_application\$(MICRO_SUB_VARIANT)\make\ghs\$(CURRENT_APP
L)_$(MICRO_SUB_VARIANT)_$(DEVICE_NAME)_Sample.ld
LNKFILE_DB =
$(PROJECT_ROOT)\$(MICRO_FAMILY)\$(MICRO_VARIANT)\modules\<ms
n>\sample_application\$(MICRO_SUB_VARIANT)\make\ghs\$(CURRENT_A
PPL)_$(MICRO_SUB_VARIANT)_$(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                                       #
#####################################################################
```

5.3. 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).

5.4. 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.
- TRXML_CONFIG_PATH: TRXML Configuration file path used for the <MSN> Driver Component.
- TRXML_CONFIG_FILE: TRXML 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:

5.4.1. Targets Supported By The Sample Base Makefile

5.4.1.1. 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.

5.4.1.2. 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).

5.4.1.3. 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)

.

5.4.1.4. 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” in this case).

5.4.1.5. 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”).

5.4.1.6. 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”.

5.4.1.7. <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.

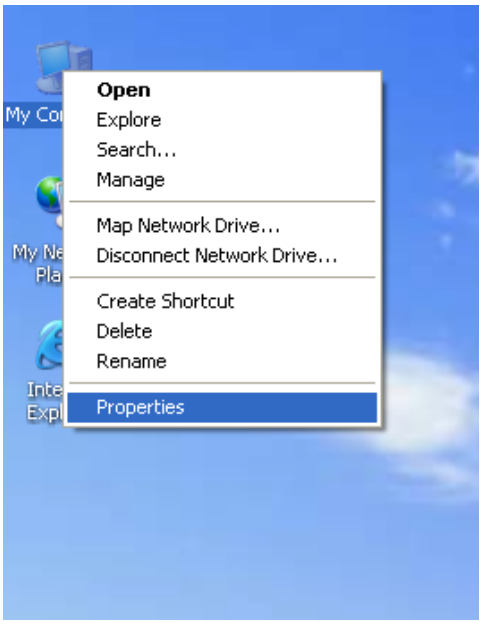
7.1. Build Process With GNUMAKE

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.

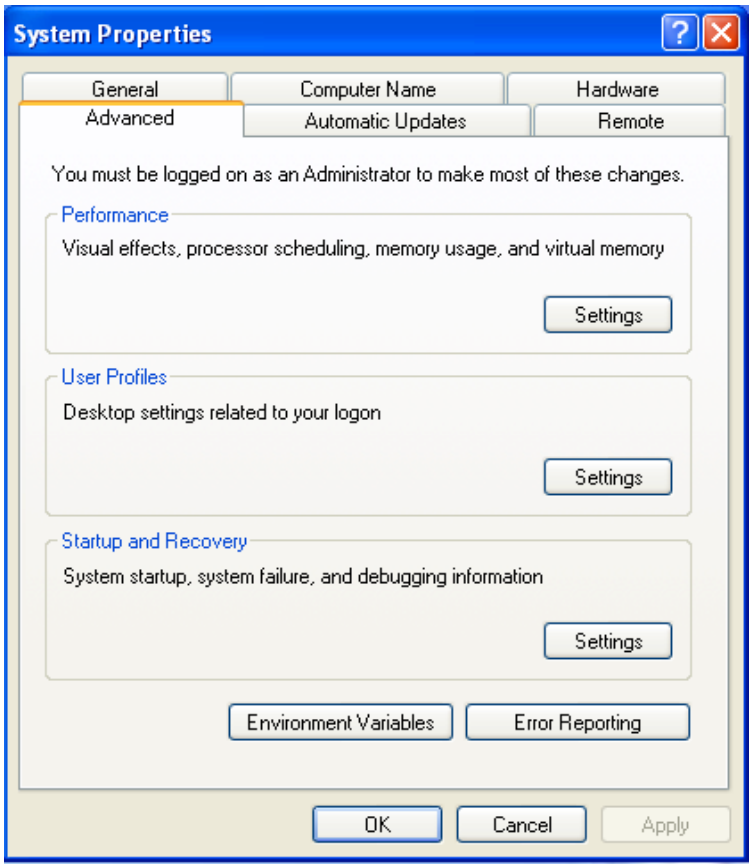
7.2. Build Process Without GNUMAKE

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.

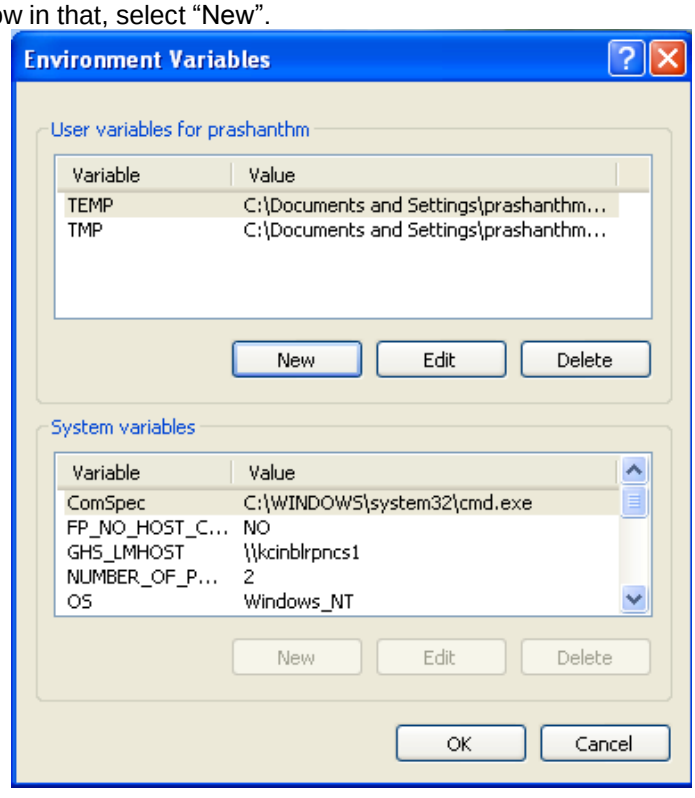
1. Right click on "My Computer" select properties, user will find System Properties.



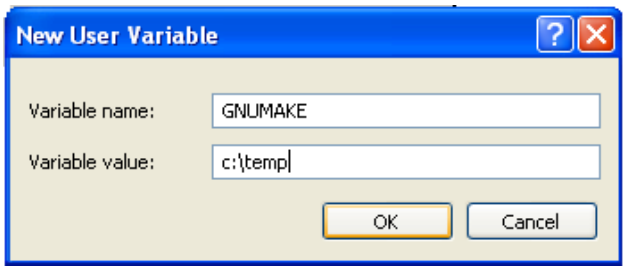
2. In System Properties select “Advanced” option, user will find “Environmental Variables” at the bottom side of window.



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



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.



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

9.1. Translation XML File

Translation XML File content format shall be given as mentioned below:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
The tag PATH-DETAILS should not be renamed since it is top level element.
-->
<PATH-DETAILS>
<!--
TRANSLATION-FILE-PATH should contain the path of the translation
header file.
The tag TRANSLATION-FILE-PATH should not be renamed. Only respective
value should be updated for the translation header file.
-->
<TRANSLATION-FILE-PATH>
   <value_of_<MSN>DeviceName>Path</value_of_<MSN>DeviceName>
</TRANSLATION-FILE-PATH>
<!--
The tags present in DEVICE-FILE-PATH tag should contain the path of
the device specific C Header File.
The tags present in DEVICE-FILE-PATH should be equal to the value
for parameter <MSN>DeviceName present in <MSN>General container.
The tag DEVICE-FILE-PATH should not be renamed.

If multiple device header files need to provide for same device then each file
name should be separated with space.
-->
 <DEVICE-FILE-PATH>
   <value_of_<MSN>DeviceName>Path</value_of_<MSN>DeviceName>
</DEVICE-FILE-PATH>
</PATH-DETAILS>
```

9.2. Configuration XML File

Configuration XML File content format shall be given as mentioned below:

```
<?
xml version="1.0" encoding="UTF-8"?>
<!--
```

```
None of the tag from this XML should be renamed or deleted.
-->
<XML>
  <!-- Supported Command Line options  -->
  <OPTION>
    <!-- Only ON or OFF should be provided. -->
    <HELP>ON/OFF</HELP>

    <!-- Only ON or OFF should be provided. -->
    <LOG>ON/OFF</LOG>

    <!-- Only ON or OFF should be provided. -->
    <DRYRUN>ON/OFF</DRYRUN>

   <!-- Only ON or OFF should be provided. -->
    <OUTPUT>OFF</OUTPUT>

    <!-- Name of output directory -->
    <OUTPUT-PATH>Path</OUTPUT-PATH>
  </OPTION>

  <!-- To provide input files. If multiple input files need to be provided then
       each file should be separated with ",". -->
  <INPUT-FILE>Path</INPUT-FILE>
</XML>
```

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	31-Jan-2013
2.	Following changes are made: 1. -Osrc and -Oinc options are added at section 4.3. Usage. 2. Error message ERR000008 is updated at section 4.8.1. Error Messages. 3. F1x is renamed to X1x in all relevant places.	1.0.1	16-Oct-2013
3.	Following changes are made: 1. Chapter 5 is updated for paths. 2. F1x and F1L names are removed. 3. Makefile location is updated. 4. Name of executable is updated.	1.0.2	24-Jan-2014
4.	Following changes are made: 1. Page Number alignment is corrected. 2. R- Number is added for document.	1.0.3	08-April-2014
5.	Following changes are made: 1. Copyright year information is corrected. 2. R- Number is added for document.	1.0.4	17-July-2014
6.	Following changes are made: 1. Document is updated as per template.	1.0.5	09-Aug-2014

Getting Started Document for X1x MCAL Driver User's Manual Version 1.0.5

Publication Date: Rev.0.01, August 09, 2014

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, 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 Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0

Getting Started Document for X1x MCAL Driver

User's Manual

Renesas Electronics Corporation

R20UT2979EJ0001