|  |  |
| --- | --- |
| Document Release Status  RELEASED | |
| Release Date  xx-xx-2016 | Version  0.0.1 |



UiMagician

Design Documentation

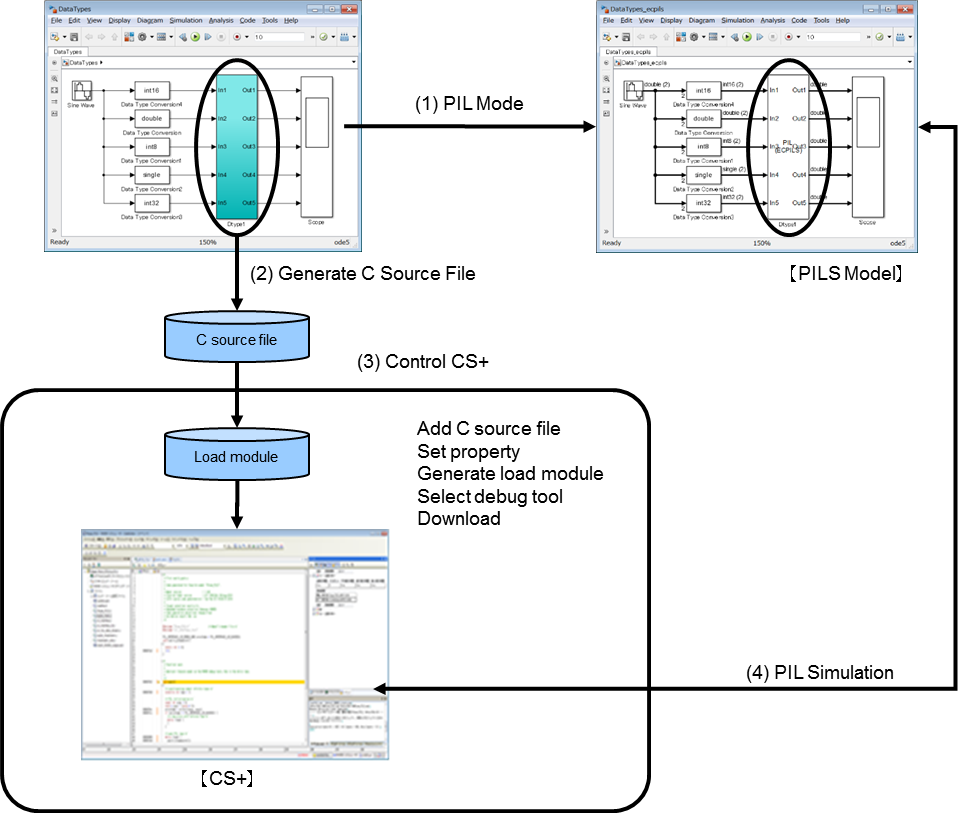
**0. CONTENTS**

1. **[General](#_Project_Introduction)**[.](#_Project_Introduction).................................................................................................................................................. 3
   1. [Overview](#_I/O_Layout_of) of UiMagician............................................................................................................. 4
   2. Features of UiMagician.............................................................................................................. 4
   3. Advantage of UiMagician.………………………….….……………………………………………... 4
   4. Target Mobile Devices supported….……..……………………………………………………….... 4
2. **UiMagician Operating environment**..………………………..……………………………………………... 4
   1. [Operating Environment..…….](#_I/O_Layout_of)……………………………………………………………………….. 6
   2. Install Process UiMagician……………………………………………..…………………………….. 6
   3. Uninstall Process UiMagician ….……………………………………..…………………………….. 6
   4. Intel XDK installation Guide.………………………………………..……………………………….. 6
3. **UiMagician Software Architecture**……………..………………………………………………………….....7
   1. Programming language used………………………..…………….………………………………...7
   2. Basic design & operating concept………………………………….………………………………...7
   3. Software High Level Design…………………..……………………..……………………...………...7
   4. Javascript Engine High Level Design………..………………………..…………………...………...7
   5. Intel XDK Project for mobile app Generation..……………………….…………………...………...7
   6. UiM Tool Basic Layout…………..…………………………………………………………..………...7
   7. UiM XML file layout…………..………………..……………………..……………………...………...7
4. [**UiM**](#_Project_specification)**agician for VSCP**….……………………………………………………………………………...............10
   1. What is VSCP..…………..…………………………………………………………………..………...7
   2. UiM for VSCP Tool Layout…………………………………………………………………..………...7
   3. How to Enter Device details…………………….………………………………………...............10
   4. How to generate HTML /Mobile app………..…………………………….…...........................10
   5. How to generate Mobile app using Intel XDK …………………………………….…..............10
5. **UiMagician - FAQ**………………………………………………………………..…….………………...........19
   1. How does UiMagician works ?………………………..….….......................................................10
   2. Can I Edit my XML configuration file of my House ? …………………………………………..10
   3. What types off apps are generated by UiMagician ?……………………..……………..........10
   4. How to port Html UiM files generated ?……………..……………….….…............................10
   5. Next Release Plan ?……………………………..……………………….….….............................10
6. **CHANGE HISTORY**……...……….…………………………………..………………...……………………... 24

# General

This chapter provides an overview of the functions of “Embedded Target for Renesas CS+ (Processor In the Loop Simulation System)”(ECPILS).

## Overview of ECPILS

ECPILS facilitates the verification of algorithms in embedded models by generating a test environment automatically in Processor In the Loop Simulation System (hereafter referred to as PILS).

Remark ECPILS executes operations (1) to (4) in the above figure automatically.

## Features of ECPILS

### Algorithm verification

PIL simulation, which is sequentially executed in combination with MATLAB/Simulink and CS+,

enables the verification of algorithms for the load module generated by embedded models.

### Verification of execution time

The execution time is measured by executing the load module generated by embedded models

on CS+. The measurement result can be checked on the storage file for execution time

measurement result which has been automatically saved.

## Advantage of ECPILS & Target support Package

Allow processor in loop simulation for model in form of subsystem. Also used for measuring execution time period of your model algorithm on target. ECPILS was never designed with goal to verify complete system simulation on target.

If you want to simulate complete system with multiple tasks. You can create multiple subsystems inside single subsystem and trigger them at right time. And create ECPILS model from it & use it for simulation.

To operate hardware while PILS simulation. You can combine Matlab logic with s-function block (for hardware drivers) to simulate hardware during PILS simulation.

PILS simulation can simulate complete system for you in this way.

## Target Devices Supported

The following target devices are supported.

- RH850

- RX

- V850

- RL78

- 78K0R

# ECPILS & hardware control environment

Described below are the system requirements for ECPILS simulation with target support using s-function.

## Operating Environment

Hardware environment

- Operating system: Microsoft Windows® 7 SP1 (32 bits, 64 bits)

- Processor: 1 GHz or higher (supporting hyper-threading or multi-core CPU)

- Main memory: At least 1 GB (2 GB or more recommended)

• Software environment

* MATLAB or Simulink products (from The MathWorks, Inc.)

MATLAB V7.13 (R2011b) to V8.5 (R2015a)

Simulink V7.8 (R2011b) V8.5 (R2015a)

Stateflow V7.8 (R2011b) V8.5 (R2015a)

MATLAB Coder V2.1 (R2011b) to V2. 8 (R2015a)

Simulink Coder V8.1 (R2011b) to V8. 8 (R2015a)

Embedded Coder V6.1 (R2011b) to V6. 8 (R2015a)

* MEX-file compilers (from Microsoft Corporation)

Microsoft Windows SDK 7.1

* IDE (from Renesas Electronics)

CS+ for CC **V3.02.00**

- Build environment (for generating a load module)

CC-RL(**v1.02.00**) Included with CS+ V3.02.00 (from Renesas Electronics)

CC-RH Included with CS+ V3.02.00 (from Renesas Electronics)

CC-RX Included with CS+ V3.02.00 (from Renesas Electronics)

Remarks 1. For the MATLAB and Simulink products, an environment is constructed by using option products corresponding to the versions of MATLAB and Simulink being used.

**2. When installing MATLAB, it is recommended that the installation folder is changed to other than the folder for UAC (user account control). Depending on the version of MATLAB in use, if the installation folder is the folder for UAC such as “<system drive>:\Program Files” or “<system drive>:\Program Files (x86)”, a problem such that MEX cannot be built or the MATLAB path cannot be saved may occur.**

3. The IDE is limited to CS+.

• Debug tools

* Emulators (from Renesas Electronics)

E1

* Simulator (from Renesas Electronics)

Remark The simulator is included with CS+.

## Packages

The following package is necessary to install ECPILS.

- ECPILS\_<version information>.zip

version information = V3.20.00

Expand the package. The programs, documents, and samples are located in the following folder structure.

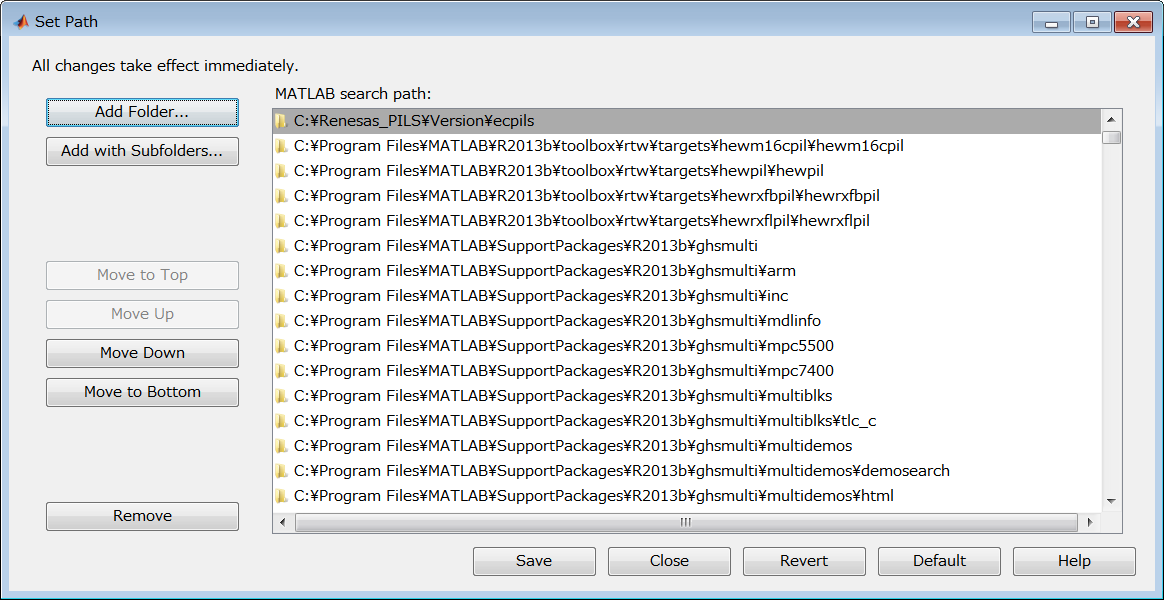
|  |  |  |  |
| --- | --- | --- | --- |
| <version Information > | ecpils | … | A set of ECPILS programs |
|  | matlab\_bin32 | … | A set of programs to be installed in the MATLAB folder (in case of using MATLAB 32-bit versions). |
|  | matlab\_bin64 | … | A set of programs to be installed in the MATLAB folder (in case of using MATLAB 64-bit versions). |
|  | Doc | … | Documents |
|  | smp | … | Samples |

## Install Process

Proceed as follows to install ECPILS.

1. Install the ECPILS package (Step 1)  
   Copy <version information> folder to any folder.
2. Install the ECPILS package (Step 2)
   * In case of using MATLAB 32-bit versions, copy the following files in matlab\_bin32\ folder to <MATLAB installation folder>\bin\win32\ folder:
     + - IdeCom.dll
       - IdeComRap.dll
       - Microsoft.Dynamic.dll
       - PythonAdapter.dll
       - PythonConsoleLibrary.dll
       - PythonConsoleRemoting.dll
       - IronPython.dll
       - rtiostream\_cubesuiteplus.dll
   * In case of using MATLAB 64-bit versions, copy the following files in matlab\_bin64\ folder to <MATLAB installation folder>\bin\win64\ folder:
     + - IdeCom.dll
       - IdeComRap.dll
       - Microsoft.Dynamic.dll
       - PythonAdapter.dll
       - PythonConsoleLibrary.dll
       - PythonConsoleRemoting.dll
       - IronPython.dll
       - rtiostream\_cubesuiteplus.dll
3. Register in the MATLAB search path   
   Start MATLAB, and add the <ECPILS installation folder>\<version information>\ecpils folder copied in the procedure (1) using the Set Path dialog box.

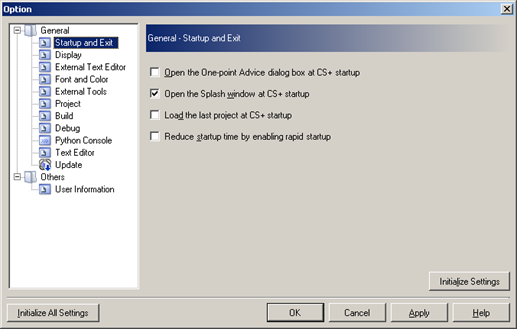
Figure 2.1 Set Path Dialog Box



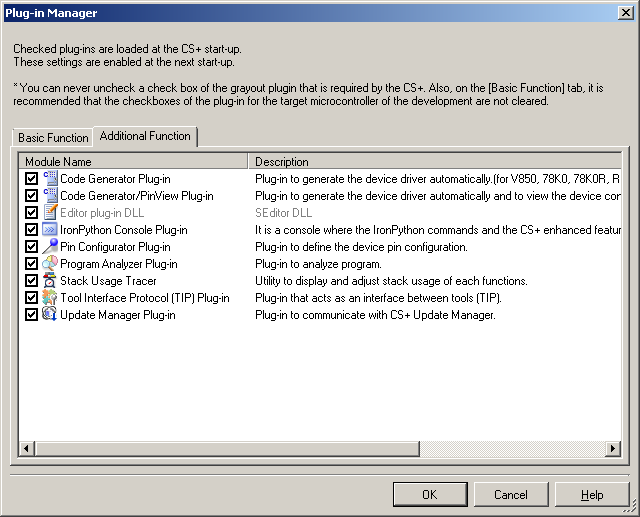
1. Register the MEX-file compiler   
   Execute the following command from the MATLAB command window to specify Microsoft Windows SDK 7.1 as a compiler for MEX files.   
   In this command, ">>" denotes the command prompt, "△" denotes a space entered with the space key, and "[Enter]" denotes entry of the Enter key.

>> mex △ -setup [Enter]

1. Restart MATLAB. After that, ECPILS can be used.
2. Start CS+, and then uncheck the checkbox for [Reduce startup time by enabling rapid startup] in the option dialog box which can be opened by selecting [Options] from [Tool].

Figure 2.2 CS+ Option Dialog Box

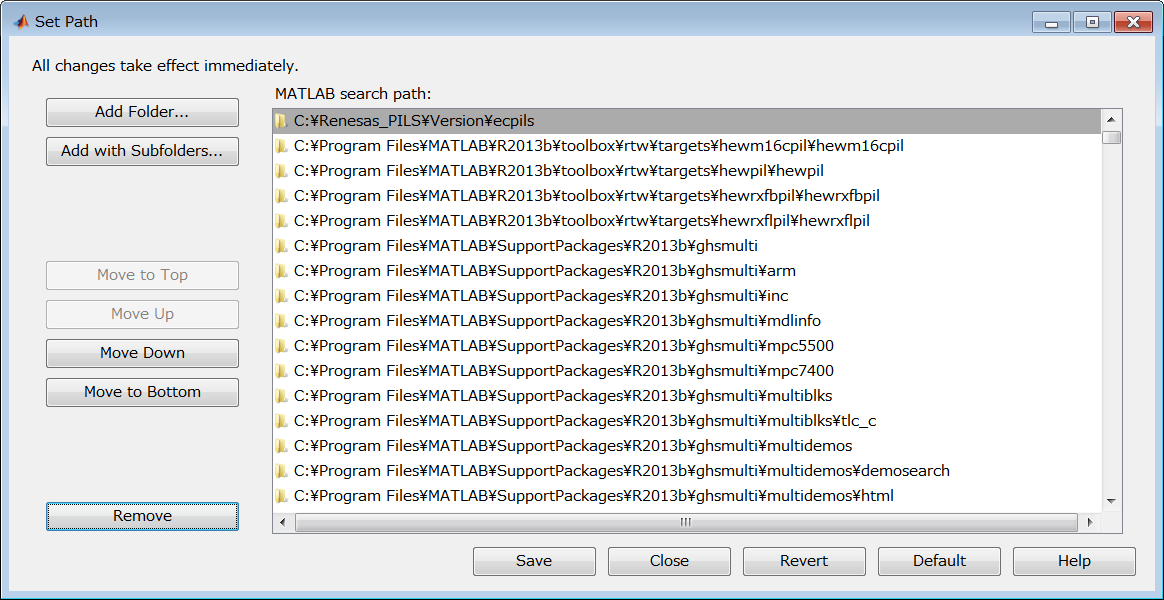
1. Check the checkbox for [IronPython Console Plug-in] in the Plug-in Manager dialog box which can be opened by selecting [Plug-in Setting] from [Tool].

Figure 2.3 CS+ Plug-in Manager Dialog Box

## Uninstall Process

Proceed as follows to uninstall ECPILS.

1. Start MATLAB and remove the <ECPILS installation folder>\<version information>\ecpils folder using the Set Path dialog box.

Figure 2.4 Set Path Dialog Box

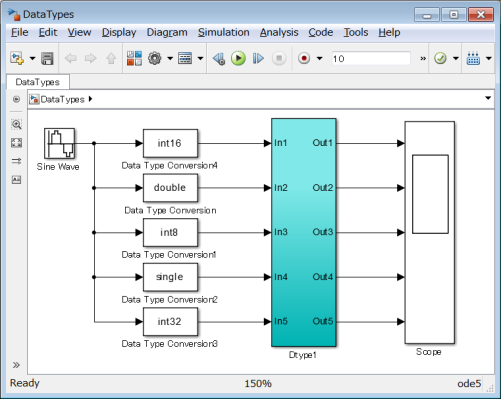
1. Delete the <ECPILS installation folder>\<version information> folder and delete the files in <MATLAB install folder>\bin\win32\ (in case of using MATLAB 32-bit versions) or <MATLAB install folder>\bin\win64\ (in case of using MATLAB 64-bit versions) which has been copied in installation.

## Matlab Model settings for Hardware Simulation

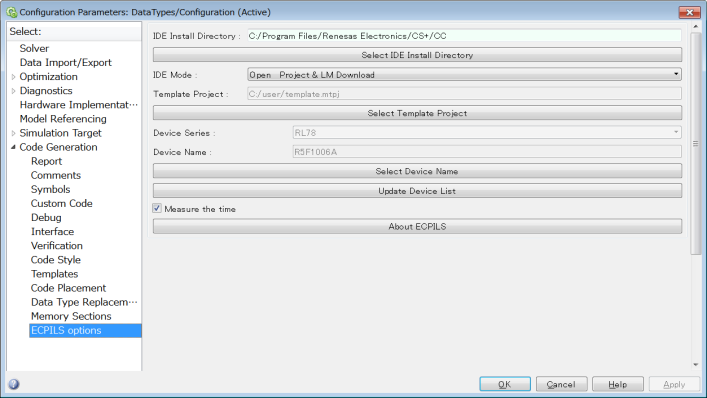
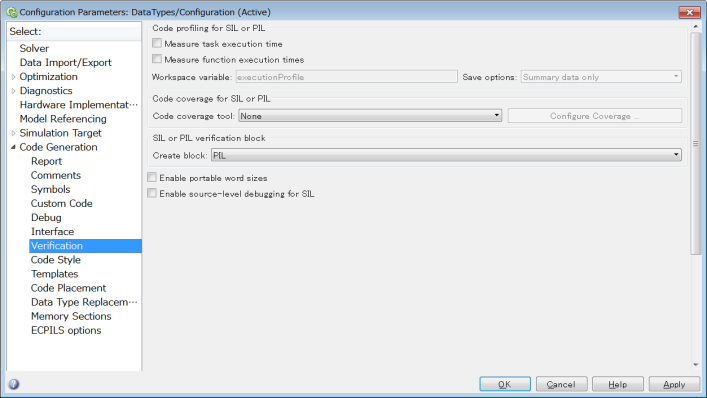
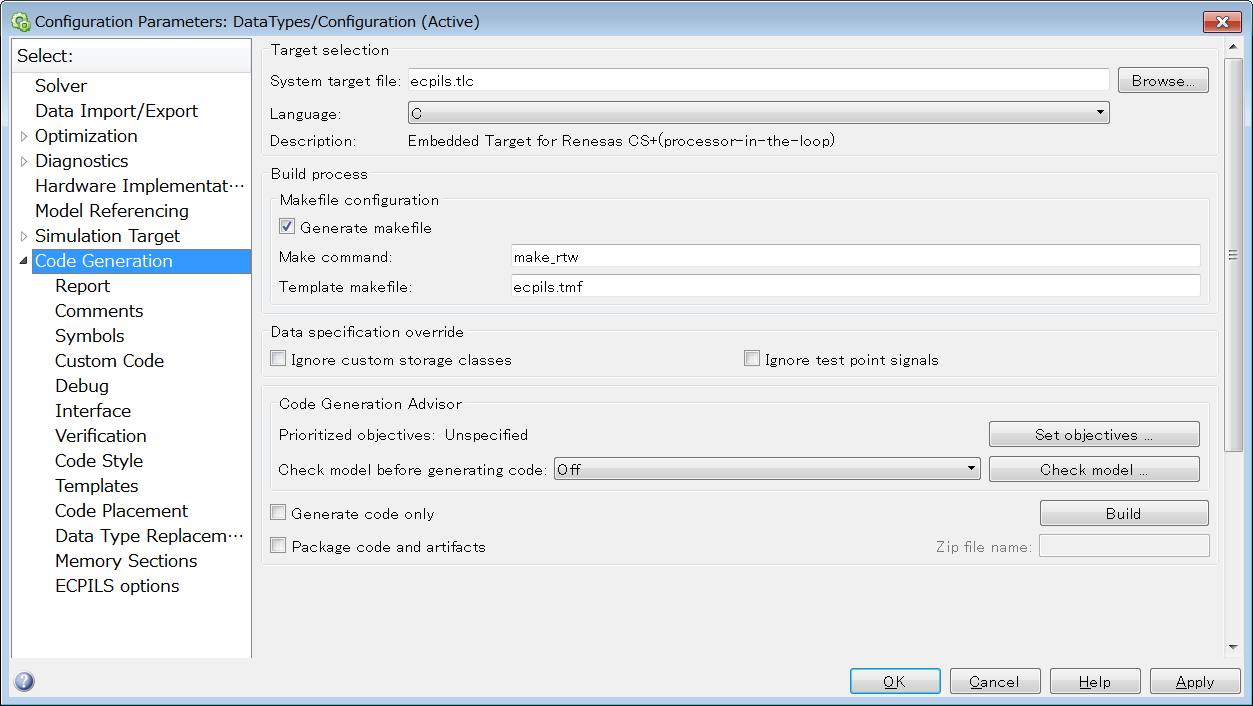
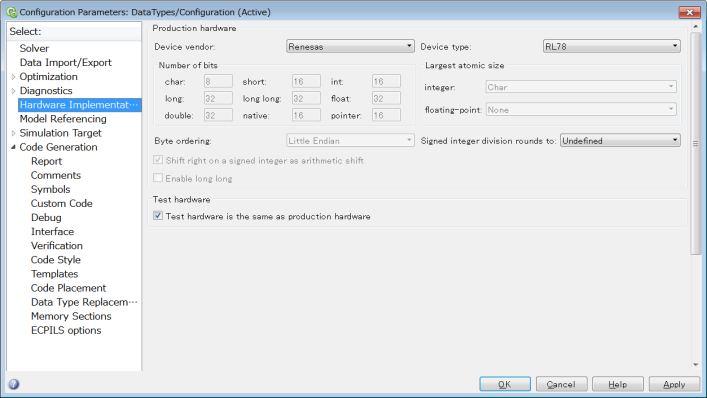
Ecpils script “ecpils\_build” use to generate code out of the Model subsystem create an CS+ project for PILS simulation.

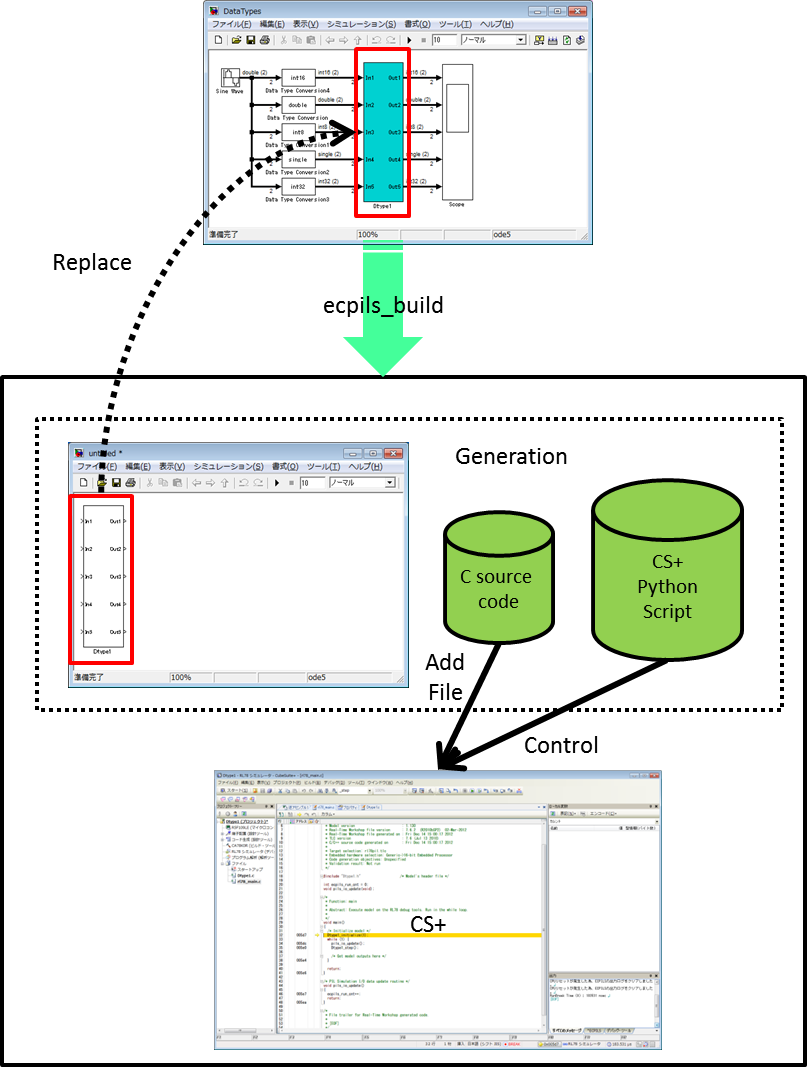
Following settings have to be done in base model properties & hardware configuration, before running the Ecpils script “ecpils\_build”.

* 1. Model subsystem for LED Toggle.
  2. Create a Simulink model. Convert the code generation target blocks to the subsystems and group them into a single subsystem block.



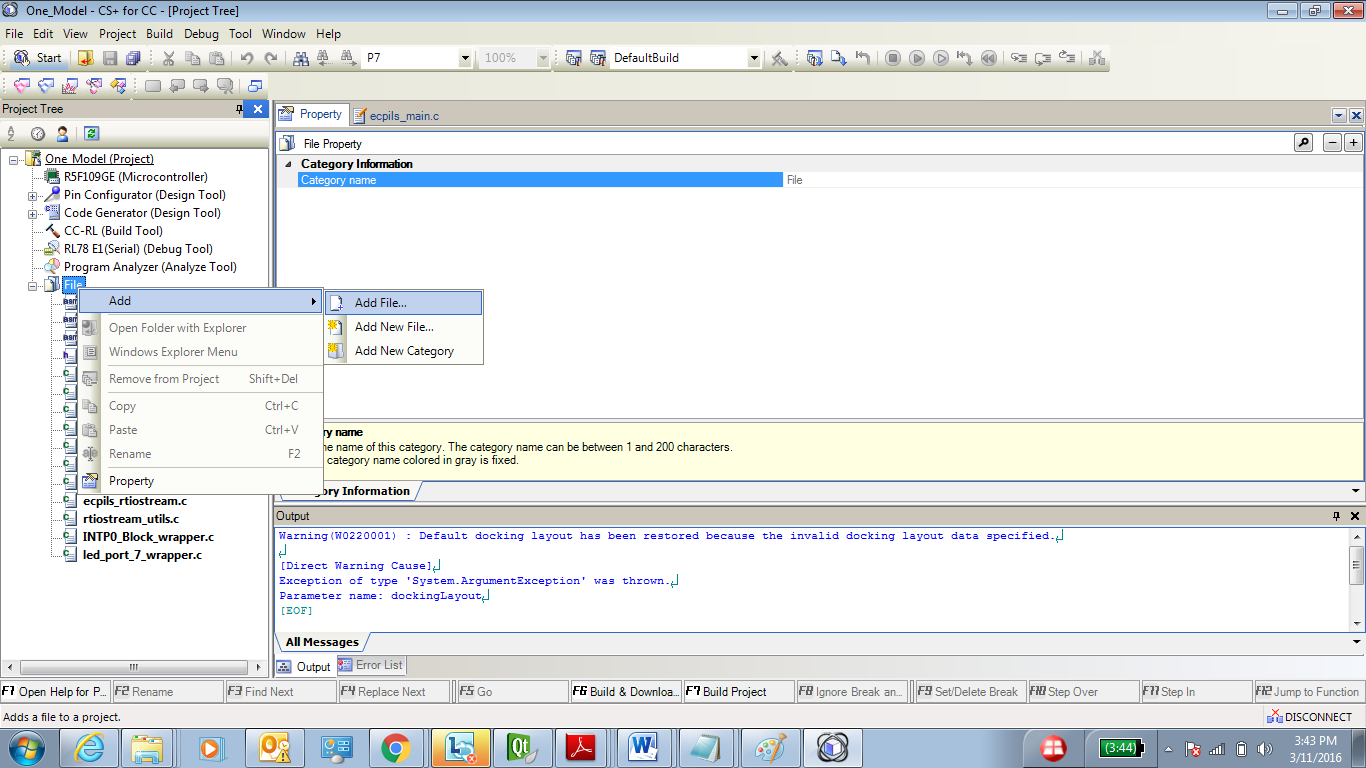
* 1. Use the configuration dialog box to set the parameters necessary for generation of codes and a test environment.

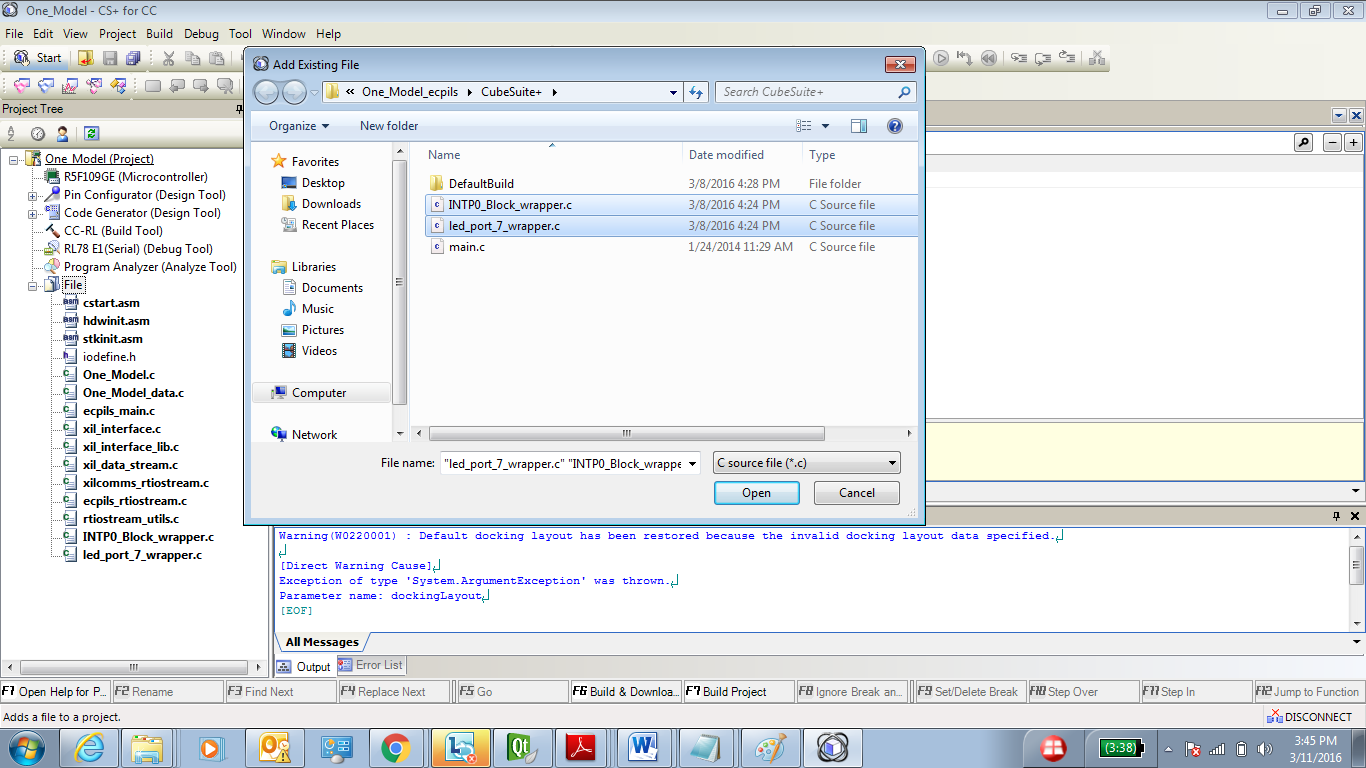


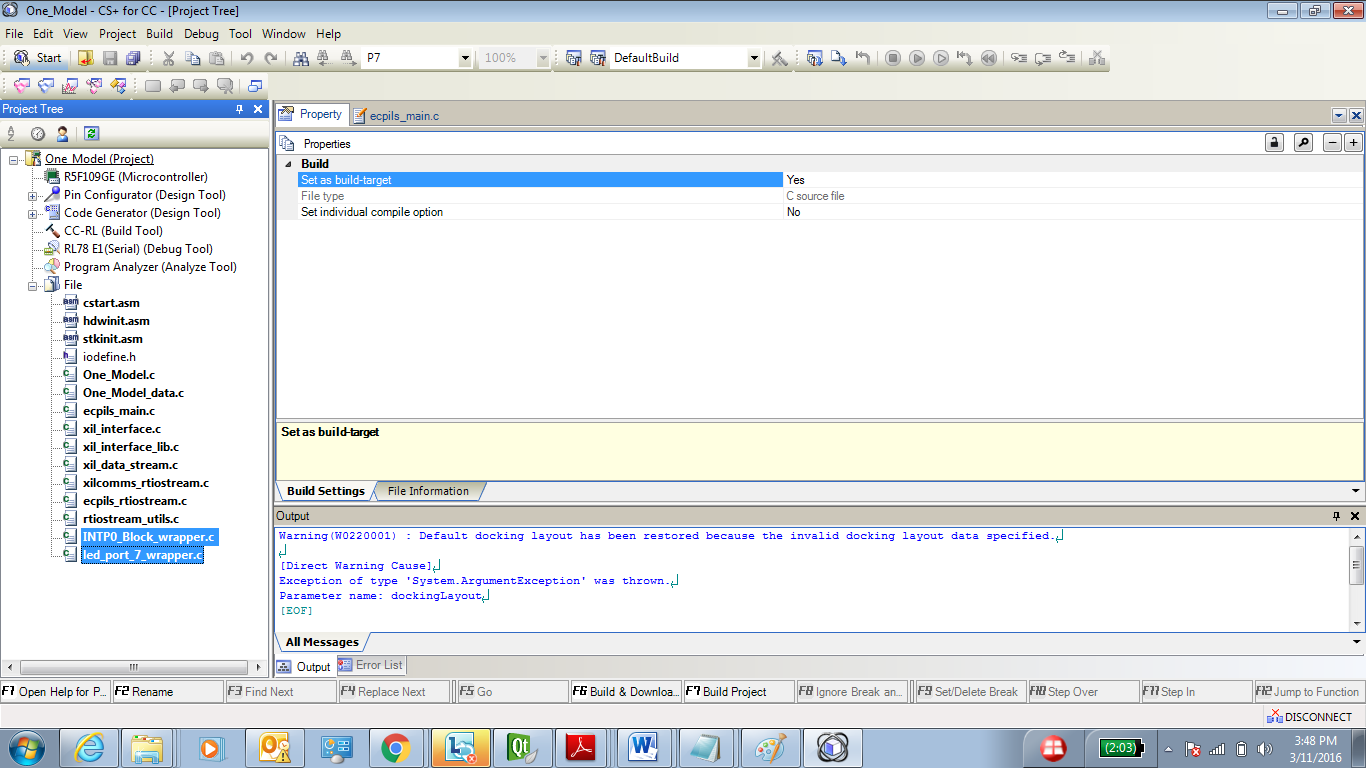
* 1. Execute the ecpils\_build command in the MATLAB command window to generate codes and to generate a PIL test environment. Generate a C source code and start CS+. The Simulink model replaces the generated block for PIL sequential execution with the subsystem.

## Cubesuite settings for Hardware Simulation

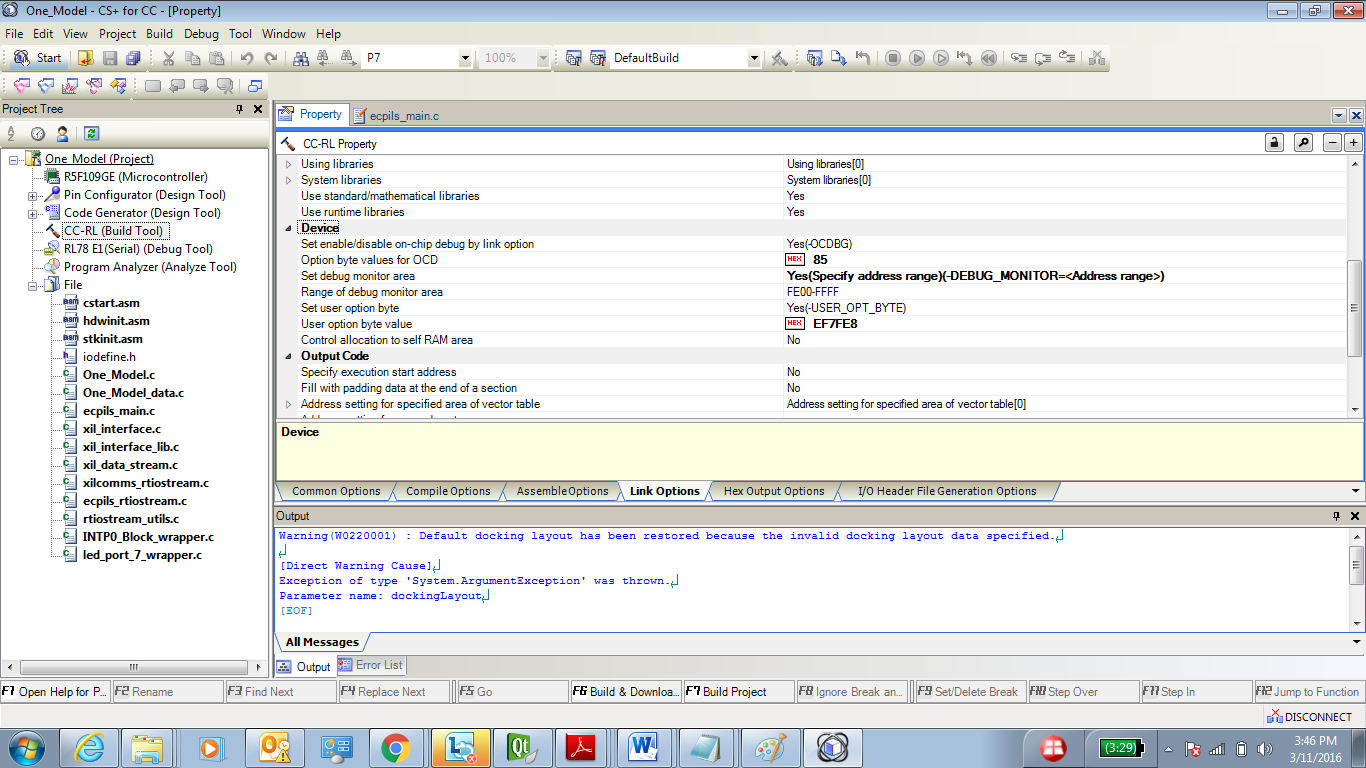
1. Add S-function generated files to CS+ auto generated project.



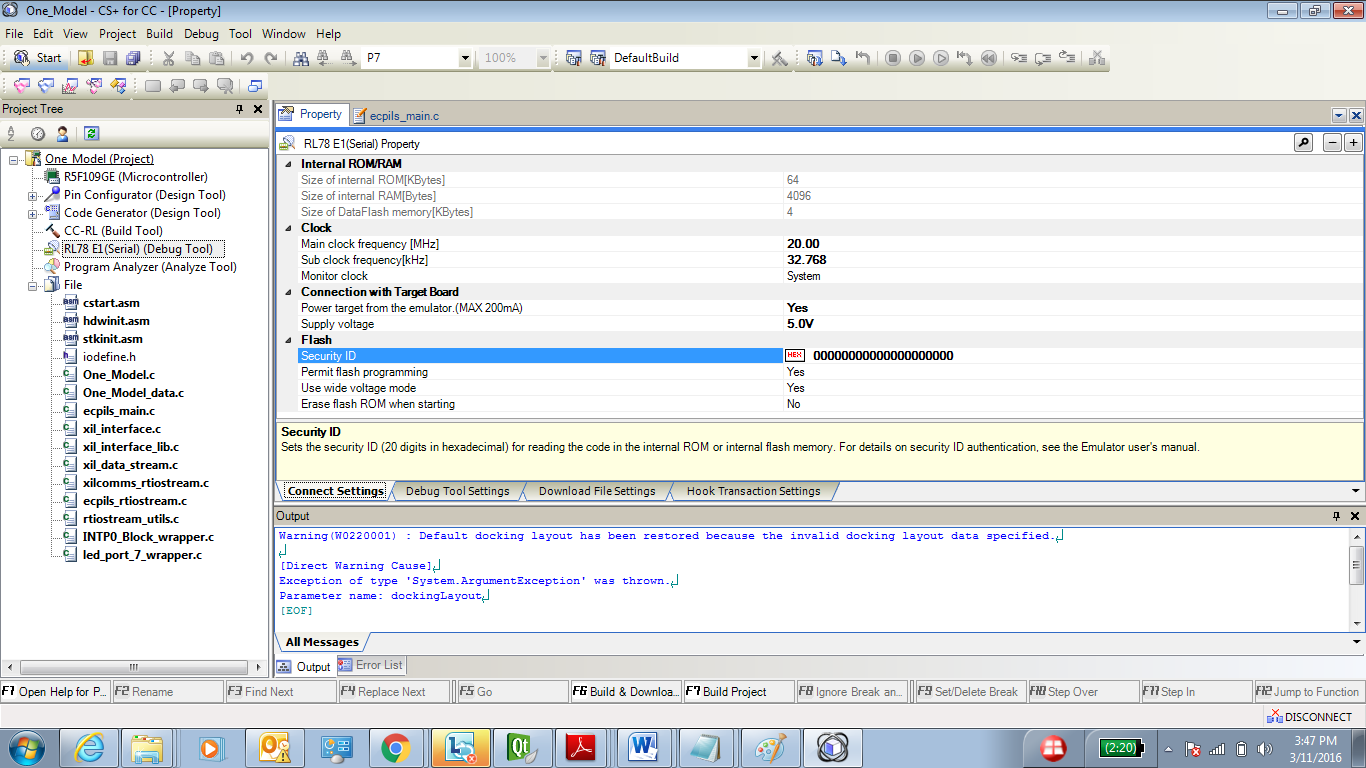


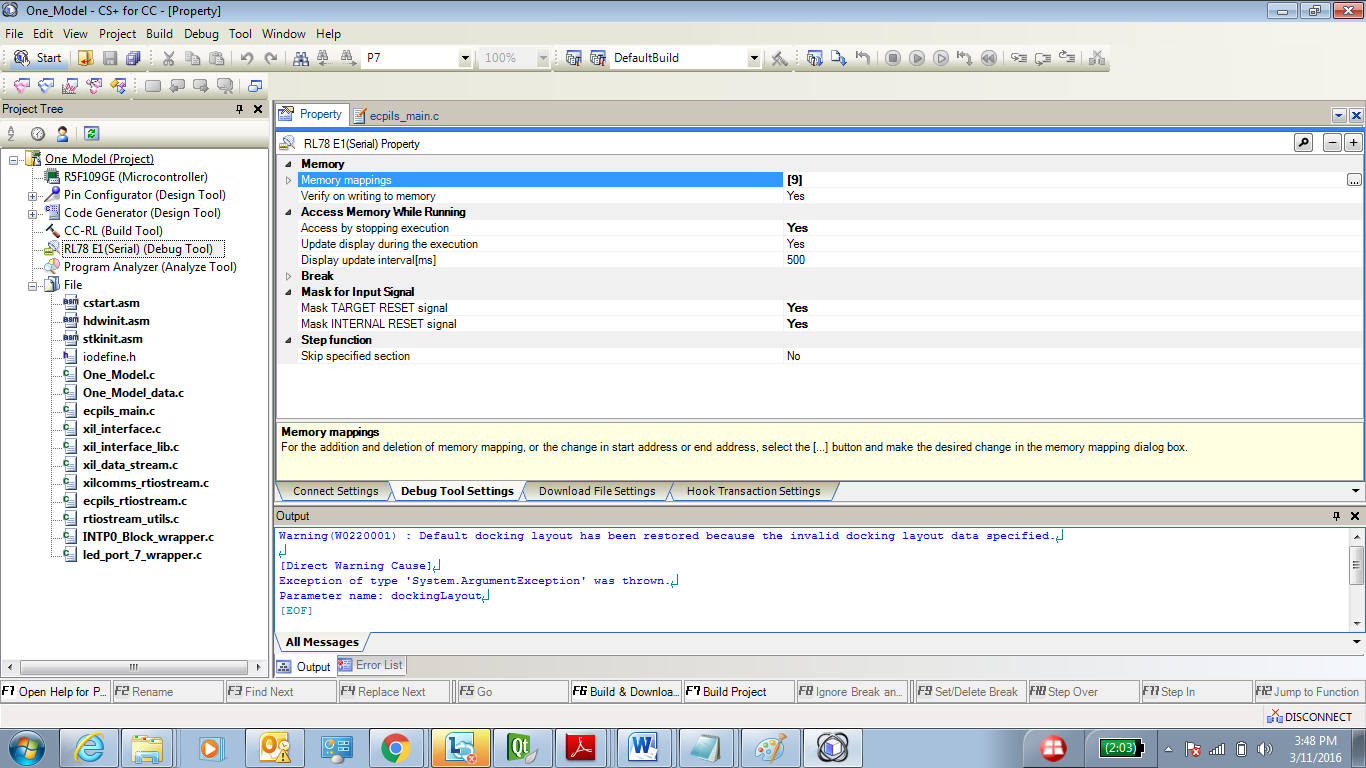


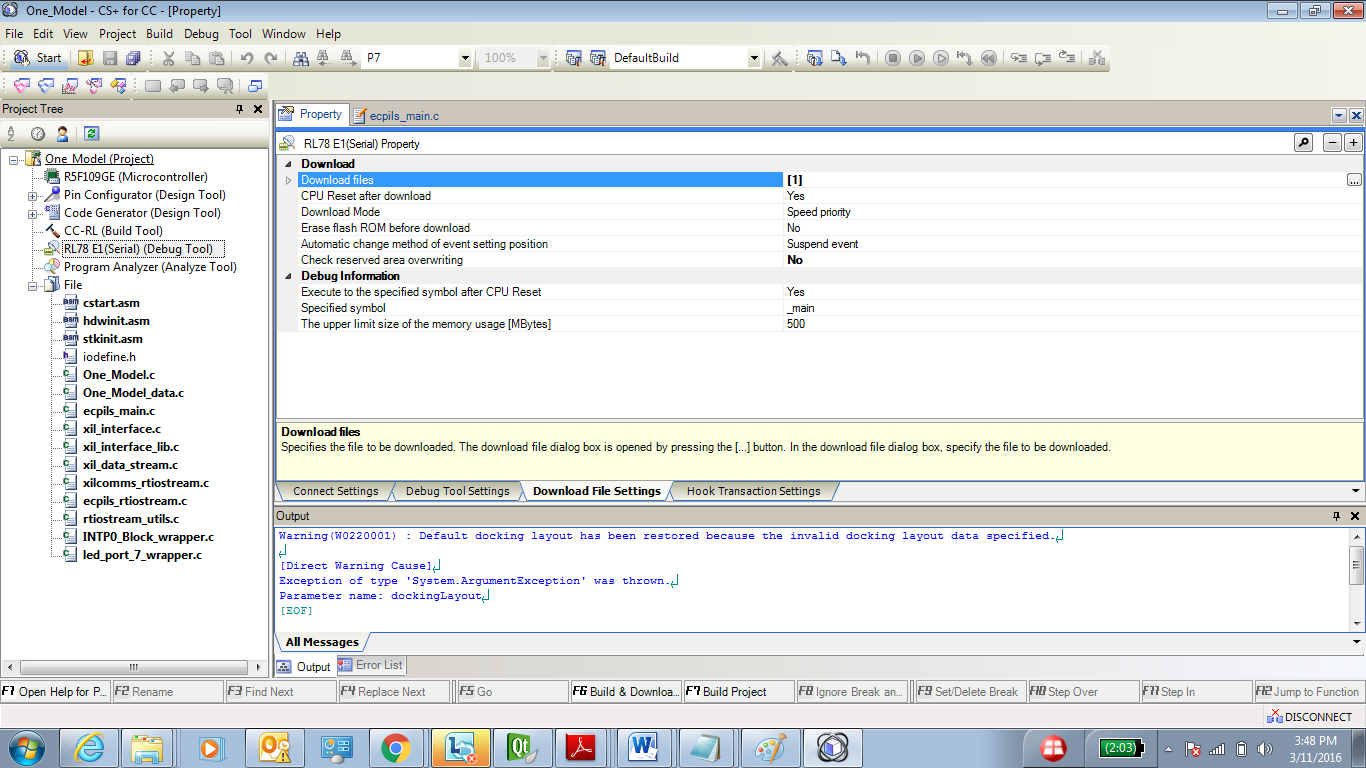
1. Set Link Option :--



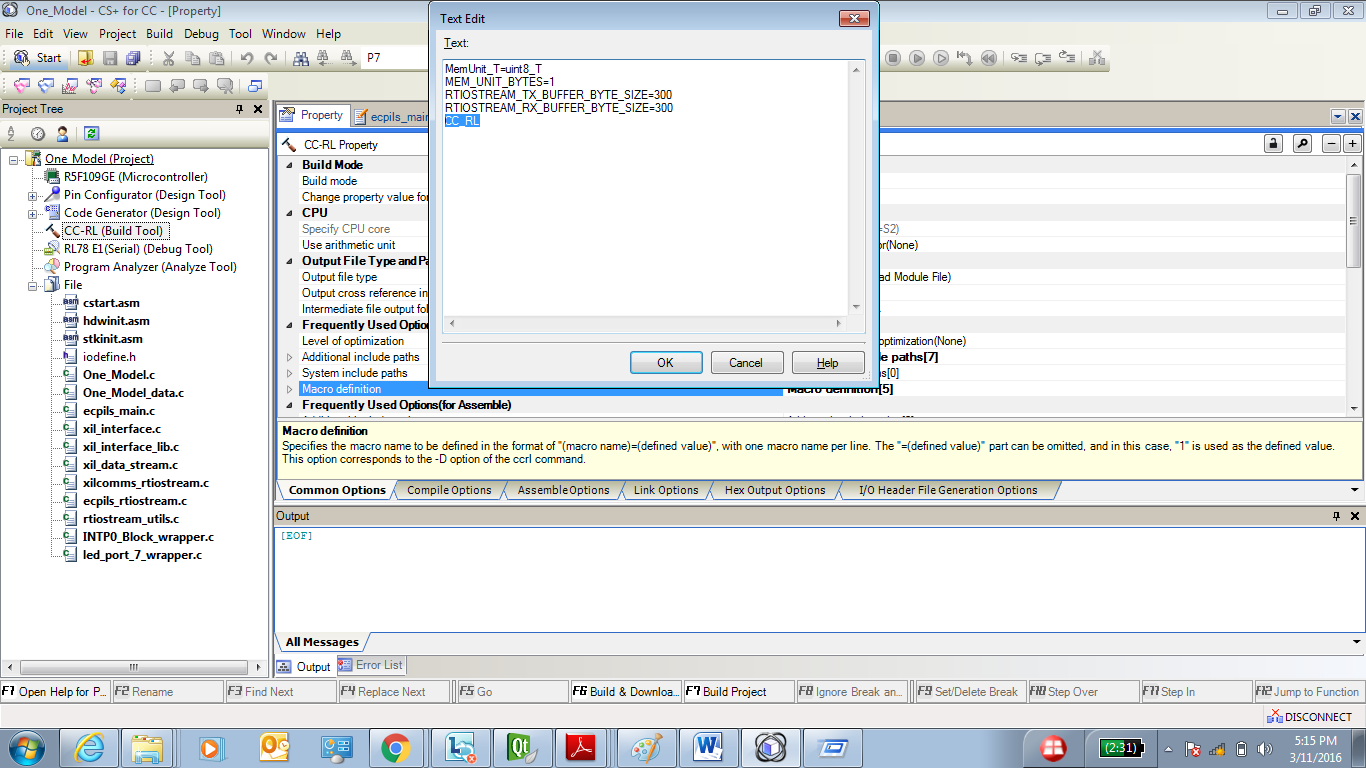
1. Set EI settings :--



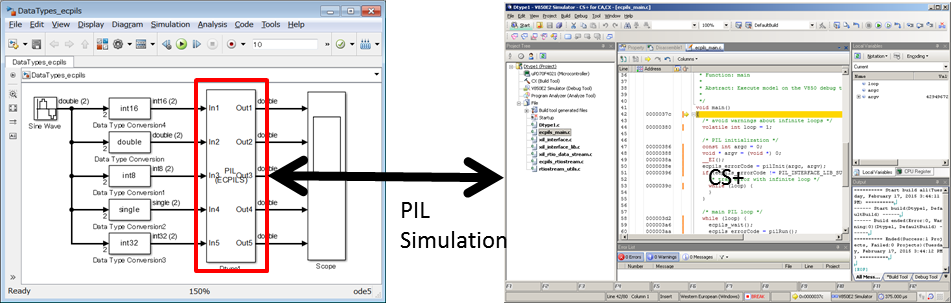




1. Set the CC\_RL macro in the Macro Definition tab for CC\_RL build tool :--



1. Start simulation using the Simulink model to execute PIL simulation for Hardware control.



# Matlab Target Support package

This section we will cover how we can add the hardware control support directly to the MATLAB model.

## What is S-Function

S-functions (system-functions) provide a powerful mechanism for extending the capabilities of the Simulink® environment.

An S-function is a computer language description of a Simulink block written in MATLAB®, C, C++, or Fortran.

C, C++, and Fortran S-functions are compiled as MEX files using the mex utility (see Build MEX File). As with other MEX files, S-functions are dynamically linked subroutines that the MATLAB execution engine can automatically load and execute.

## How to integrate s-function in Matlab Model

Matlab use to provide s-function Simulink block which you can link to the \*.MEX file related to that block.

Also using S-function builder you can integrate your C language driver code to the Simulink s-function builder block. S-function builder also allow you to directly build the s-function builder block to MEX file and auto generate code for your s-function, which you can integrate with CS+ project manually, before starting the hardware simulation.

## Using s-function as Target support Package

Inside the Simulink block of the s-function builder you can fit your device driver code written in C language. So in this way you can create a Target support package for your MCU device.

Once you build the S-function builder block it creates a MEX file & generates wrapper code around your device driver using Simulink coder.

You integrate this wrapper C code to your auto generated CS+ project from ecpils\_build script. This way you can simulate your hardware MCU device & retrieve status of hardware MCU directly inside the MATLAB Simulink model.

# Sample Models

This section will cover the details of the sample models which have been tested for Target support package using s-function Simulink block.

## Toggle LED & External Interrupt

## Horn control example

## Combining multiple subsystem tasks

# ECPILS & target support package – FAQ

## PILS simulation v/s Rapid ECU Prototyping

PILS simulation allows processor in loop simulation for model in form of subsystem. Also used for measuring execution time period of your model algorithm on target. In PILS simulation the model simulation time is decided by the Simulink model running on PC.

PILS simulation is bit slow to retrieve result from target & showing on the Simulink scope or display block.

Rapid ECU prototyping is done on special Hardware which is high speed running processor. So it’s easy to retrieve state of system at faster rate and displaying them onto the Matlab Simulink model.

## Can ECPILS hardware simulation be used for time critical application?

PILS simulation is bit slow to retrieve result from target & showing on the Simulink scope or display block. So it will not be compatible with Rapid ECU prototyping for realtime retrieving state of the system onto the Simulink model & showing on the Simulink scope or display block.

So say for example Engine control system, operation on target will not match the simulation state on the scope block of the Simulink model. Because state is not updating on Simulink scope window at the same rate at which its updating on the target.

## Actual target support package vs S-function target support package.

Professional made target support package consist of set of S-functions in the form of custom library block. They can be added and removed from the MATLAB Simulink library browser.

To create a library we need MATLAB package license for this.

If we do not have license to create library inside MATLAB, we can directly add S-function block to the MATLAB Simulink Model & can drive our hardware.

So conclusion is that both will be able to drive the hardware but S-function target support package cannot be integrated as MATLAB Simulink Library

## Why C99 standard complier (CCRL) is required?

S-Function use to create a wrapper code around your C driver code & auto code generated by MATLAB Simulink coder for s-function builder block use to place few header files at the top of the file. As a result of this compiler specific pragma directive which we declare for EI & DI or using SFR or interrupt handler related pragma will always come below the header files. But compilers like CACX which are based on C89 standard requires compiler specific pragma directives to be placed at the top of the file, gives error when integrating this auto generated code to the CS+ project.

We can get around this type of error using C99 standard specific compiler like CCRL which allow placing pragma directive anywhere in the C source file.

## How to combine multiple subsystem tasks?

Most of the embedded system is combination of many subtasks doing their job at specified period of time let’s say 5 msec or 10 msec time etc.

If you want to simulate complete system with multiple tasks, you can create multiple subsystems inside single subsystem and trigger them at right time.

And create ECPILS model from it & use it for simulation.

## Can code generated with s-function block used in Production system?

Yes code generated after including s-function driver block to your MATLAB Simulink model can be brought to production level, as that code is simply combination of the driver code plus the code generated by other MATLAB Simulink or stateflow block inside your model subsystem.

## Target support package is ideally used when?

As we know that in Automotive ECU development is mostly moving towards MATLAB model based design.

Matlab target support package is ideally used for Rapid ECU prototyping when your software is ready in the form of MATLAB models but your Hardware is still under production. And at this point of your ECU development lifecycle you want to validate your software with the complete system. So under this case you need some special hardware for ECU prototyping which vendor (e.g ETAS) use to provide drivers for the various peripherals in the form of Simulink library block for Target support. You can integrate these hardware support Simulink block with your Matlab model & put your software under HIL testing.

Example you can also use Rapid ECU prototyping tools & respective target support package with your Matlab models for faster calibration of your Engine control unit.

In this way your development time is reduced & before your hardware is ready your software have been calibrated.

## Ideal s/w development for ECU using MATLAB.

Most of the embedded system is combination of many subtasks doing their job at specified period of time let’s say 5 msec or 10 msec time etc. You design your tasks into individual models like for 4 wheeler BCM ECU design wiper control model, Horn control model, exterior lighting control model, interior lighting control model , climate control model or defroster control model etc this will also make your model based software design MODULAR also.

Matlab use to generate the step() and init() function for every MATLAB model. So best is you generated your code for all the MODEL modules & integrate the step & init function to your baremetal coded scheduler.

Also most of the companies with stable software architecture use to keep their driver software as hardcoded C code & do not integrate them to the Matlab model in the form of s-function. They simply fits the auto code generated out of Matlab model algorithm in there software architecture.

# CHANGE HISTORY

* Specs Author : Dinesh Guleria (Field Application Engineer)

|  |  |  |
| --- | --- | --- |
| Version | Category | Summary |
| V0.0.1 | Environment | ・ECPILS & Target support Package for Renesas MCU |