# X2C MCC library

**Authors:**

Christoph Baumgartner

christoph.baumgartner@microchip.com

Mark Wendler

mark.wendler@microchip.com

V 1.1.2

06.02.2018

**RECOMMENDED READING**

**MPLAB X IDE User`s guide, XC compiler user`s guide:**

- http://www.microchip.com/mplab/mplab-x-ide

**Microchip Code Configurator user's guide:**

- http://www.microchip.com/mplab/mplab-code-configurator

**X2C Getting Started:**

- X2C_trunk/Documentation/GettingStarted.pdf
- http://www.mechatronic-simulation.org/

**SCILAB beginners guide:**

- http://www.scilab.org/resources/documentation/tutorials

**XCOS beginners guide:**

- http://www.scilab.org/resources/documentation/tutorials

**Fractional representation:**

- https://en.wikipedia.org/wiki/Fixed-point_arithmetic
- https://en.wikipedia.org/wiki/Q_(number_format)

# CONTENTS

# INTRODUCTION

The X2C MCC library purpose is to help to create an X2C based project structure. The module creates the necessary framework files and model files.

The X2C framework provides an easy model driven software development method, which is a good tool to create complex control algorithms, build complex mathematical systems and for fast prototyping. The first steps to start using this solution are not so trivial. The X2C MCC library tries to simplify these steps. The X2C MCC library contains the following features at the moment:

- Copy all the necessary files to the project folder;
  - X2C core source and header files;
  - All block include header files;
  - All precompiled block libraries;
- Add the X2C core source and header files to the project;
- Add the X2C.c model logic code to the project;
- Add include paths and compiler macros
- Connect the X2C scope with the selected communication peripherals;
- Connect the X2C model execution to the selected interrupt handler;
- Create a basic model;
- Create an auto compile and programmer batch script;

The user still has to do some steps manually until the development process can be focused only to the model. The details of these steps are described in the following chapters.

- Add X2C_Communicate(); to the main loop;
- Connect model Inports and Outports to the peripheral interface functions;

Abbreviations:

**MCC:** Microchip Code Configurator

**X2C library or module:** X2C add-on in the MCC

**X2C framework:** Code generation add-on for SCILAB, that allows the generate C-code of the model including a real-time online debugging interface for monitoring and modifying model signal.

# INSTALLATION

*Make sure the following programs are installed before you move on to the next chapter!*

## MPLAB X 4.10 or newer and MCC 3.45.1 or newer

The basic MPLAB X and MCC user guides contains the get started descriptions.

http://www.microchip.com/mplab/mplab-x-ide

## Scilab and XCOS 5.5.2 (6.0 is not supported yet)

The Scilab and XCOS software is used to create the model. The installation process is explained in the following link.

http://www.scilab.org/resources/documentation/tutorials

## X2C framework and MCHP X2C add-on

http://www.mechatronic-simulation.org/

http://www.embeddedcodesource.com/codesnippet/scilab-x2c-addon

Quick start videos: https://www.sim2tronic.com/

## X2C MCC module

The installation process:

1. Find the MCC directory on the computer

Just open the MPLAB X and select the Tools menu and the option point as the picture shows. Then select the Plugins at the tabs. Open the MCC Library with the “Open Library Folder”.

2. Extract all content from the ZIP file to the opened directory.
3. Restart MPLAB X

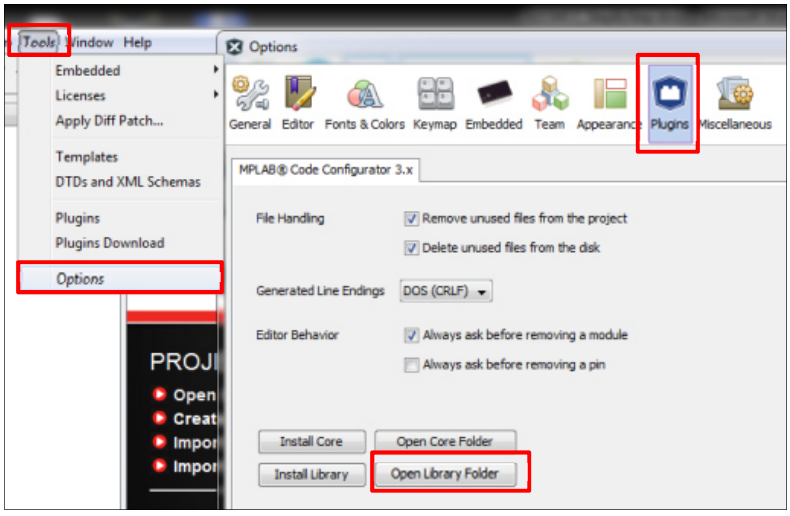



*Figure 1*

Make sure the installation was successful as the following paragraph describes it.

### Check the installation and the module version

Open or create any kind of project in the MPLAB X, then open the MCC. In the left bottom Versions window should be an X2C (v1.x.x) line below the “Software” block. Open the X2C line with the little arrow and a green tick should show up.

If there is not X2C line or there is no green tick, then try to reinstall the module or check the

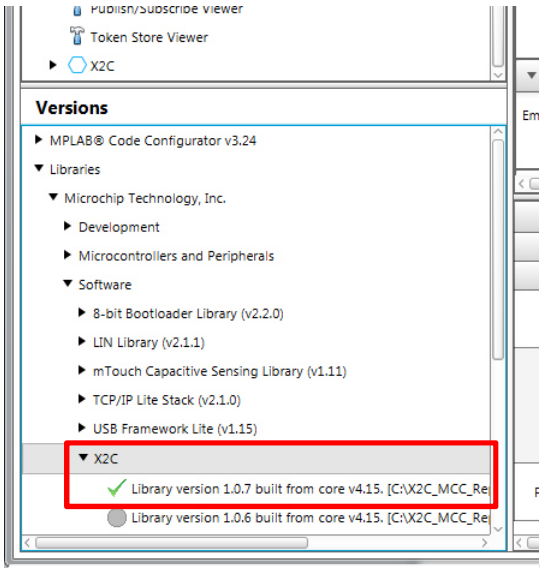



*Figure 2*

Troubleshooting chapter.

# X2C MCC MODULE BASICS

## Main steps to create a plain X2C MCC project

Here is an overview list about the main steps to build up an X2C project structure with the help of the Microchip Code Configurator:

I. Create an empty project in the MPLAB X
II. Open the MCC and add Device Resources to the Project Resources
III. Add X2C module to the Project resources too.
IV. Configure all the peripherals
V. Configure the X2C module
VI. Generate code by the MCC
VII. Setup the project parameters
VIII. Connect model inputs and outputs to the peripheral interfaces (or to the self-written code)
IX. Modify the model and regenerate the logic program
X. Automatic compiling and device programming feature

## I-IV. STEPS – CREATE MCC PROJECT AND CONFIGURE PERIPHERALS

The basic MPLAB X and MCC user guides contains the get started descriptions.

http://www.microchip.com/mplab/mplab-x-ide

# V. CONFIGURE THE X2C MCC LIBRARY

The following figure shows the X2C library. There are 4 separated sub steps to set up the X2C parameters. These steps are detailed in the following chapters.

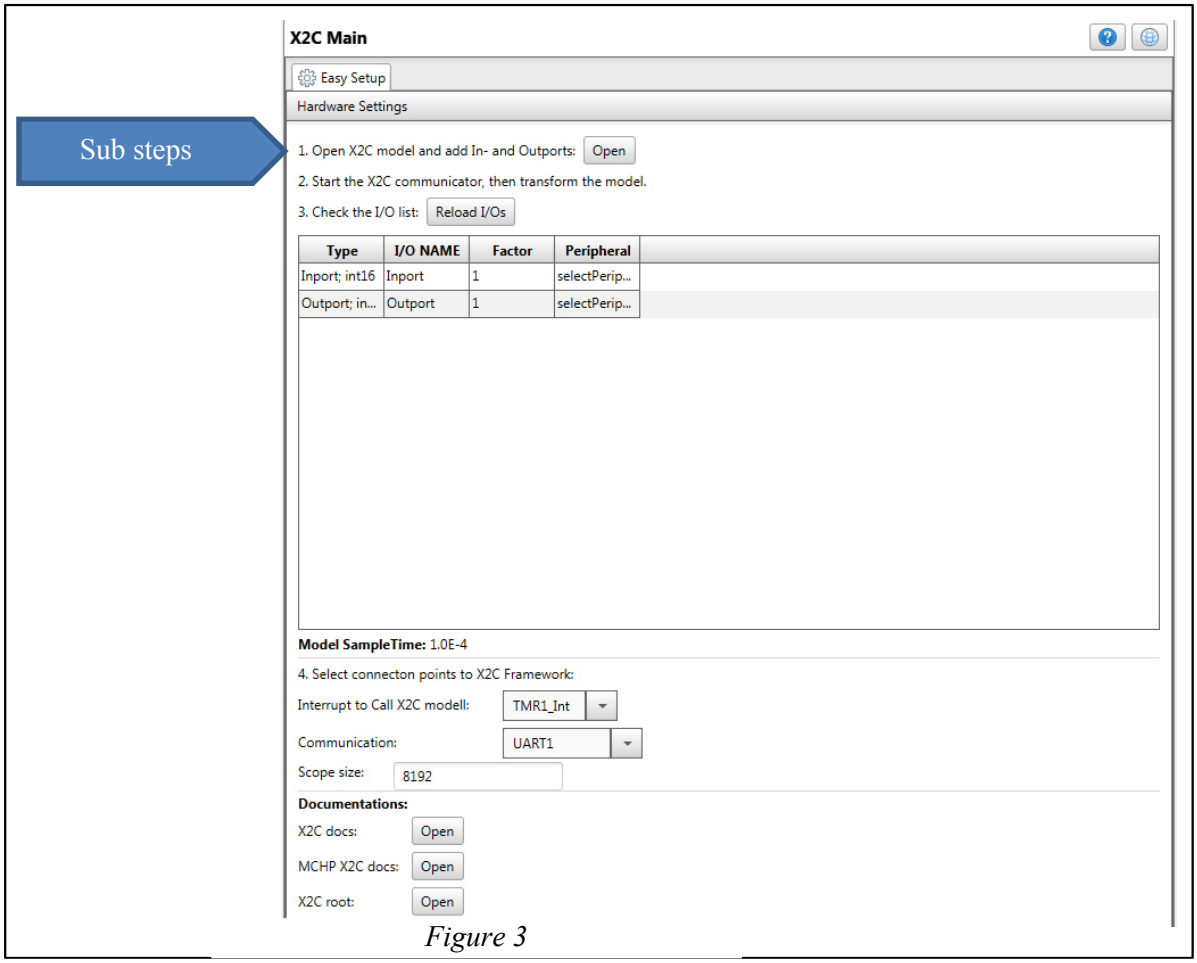



*Figure 3*

### 1. Open the model file

The open button first copies the model file to the project/mcc_generated/X2CCode folder. Then starts the Scilab and opens the project.xcos file. The XCOS model get's the same name as the MPLAB X project.

Add Inputs and Outputs to the model what is necessary for your project. You can add inputs and outputs later at the model development phase too.

Use the palette browser (View->Palette browser). The IN and OUT blocks can be found in the X2C/General palette. Or just copy the existing I/O blocks. Connections are necessary at the inputs of the blocks otherwise the code generation will fail. (The outputs of the blocks do not require connections.)

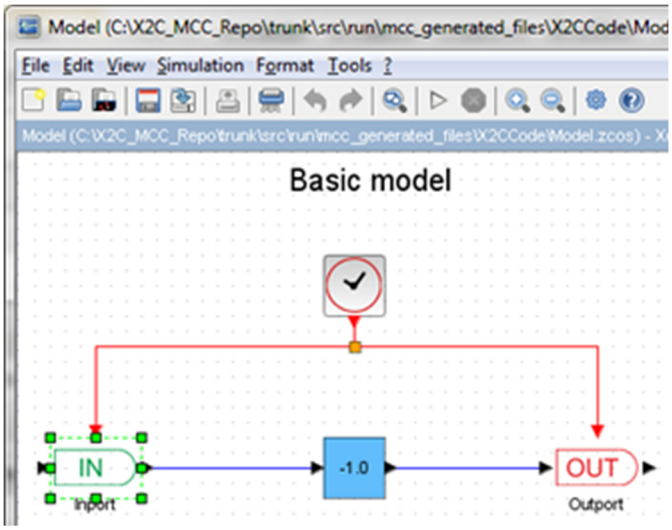



*Figure 4 – Basic model view in Xcos*

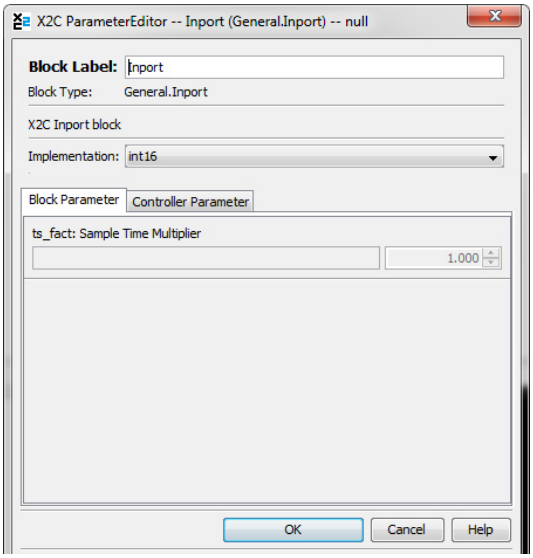



*Figure 5 – Block parameter editor*

### 2. Transform the model and generate code from the transformed model (xml)

The model file contains the following yellow boxes. If these blocks are missing, please add them from palette browser X2C/General palette.

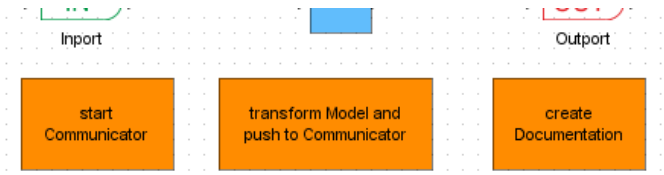



*Figure 6 – Special function blocks in the model view*

Double click on the start communicator and wait until the following X2C communicator window appears:

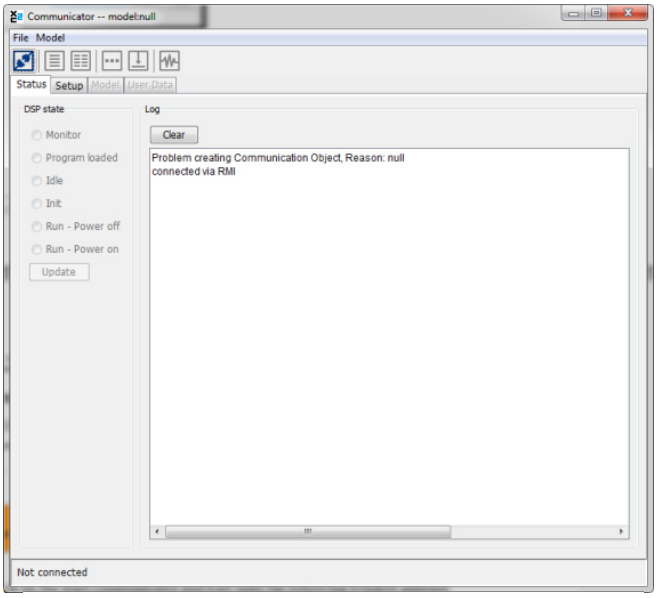



*Figure 7 – X2C Communicator output window*

Next step, go back to the model (without close the X2C Communicator) and double click to the transform Model and push to the communicator. The following outputs should appear in the communicator.

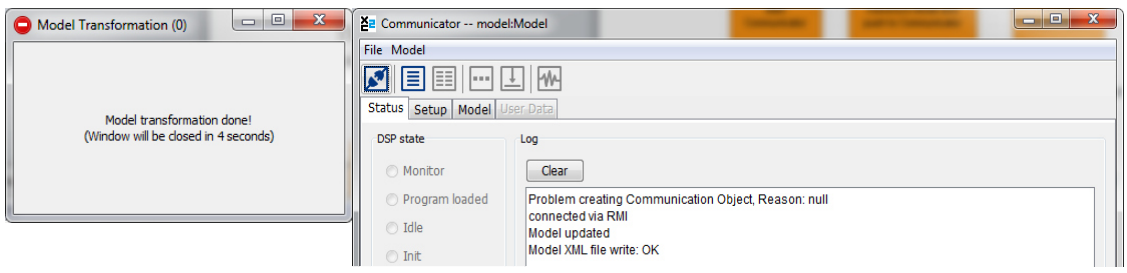



*Figure 8 – Results of the transform*

**Make sure the model name is stated in the first line of the communicator.**

### 3. Connect IO with the peripherals

The Scilab/Xcos software and the X2C communicator window can be closed.

This feature is not working yet. The goal is to connect the peripheral interface functions with the model In- and Outport blocks at the table. The user has to connect these interfaces manually as described later.

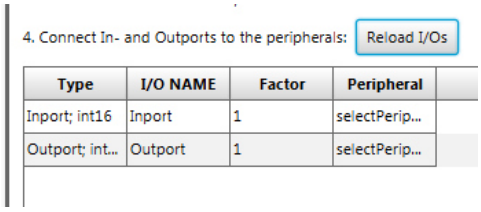



*Figure 9 – X2C MCC, IO table*

Nevertheless, the X2C library can load the model's I/O list and generates some helping comments. Push the Reload I/O button and the generated model xml description will be parsed and the I/Os of the modell will be listed in the table.

### 4. Select X2C calculation and communication connection points

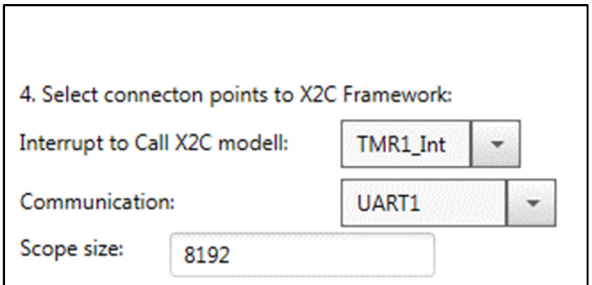



*Figure 10 – X2C main entry points*

- First select where to call the model calculation function. Make sure the selected peripheral interrupt is configured! If you select manual, then do not forget to call X2C_MainUpdate() to run the model calculation! *NOTE: The Model sample time has to be equal with the X2C_MainUpdate() function calling period, namely the selected interrupt period.*
- Then select the X2C_Scope communication interface. UART1 an UART2 supporte at the moment. Make sure again the selecte peripheral is configured fine! The X2C_Communicate(); function must be called periodically in the main loop! Do not forget to call it manually!
- With *Scope size* you can select the size of the buffer array that stores the information for the graphs.

# VI. GENERATE SOURCE CODE BY MCC

If the peripheral and X2C module configurations are ready just click the generate button at the right top corner. The whole files and project structure generated automatically. However, there are left a few steps that need to be manually done.

# VII. MANUAL TODO LIST AFTER CODE GENERATION

The project structure is ready in the MPLAB X. The following steps have to be manually done in the MPLAB X IDE. MCC does not yet support some necessary functions to automate these steps.

### 1. X2C_Communicate() function

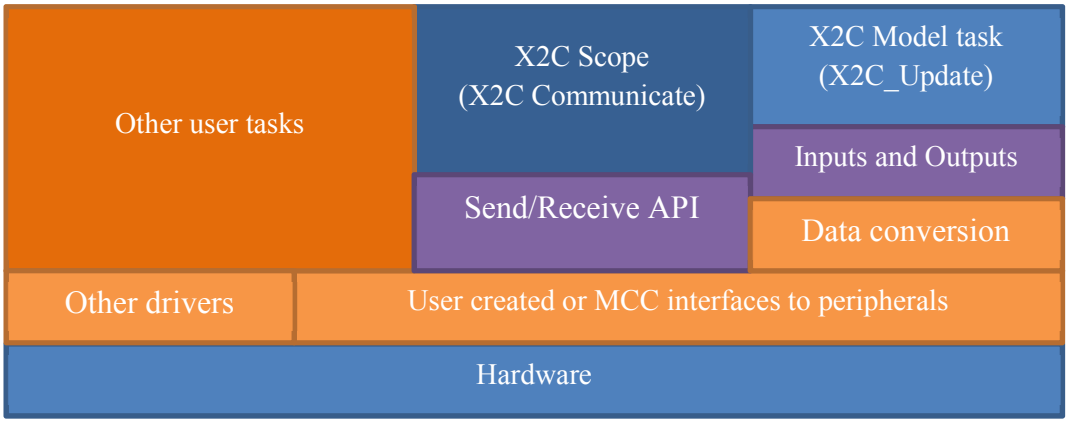
The *X2C_Communicate()* function into the main loop. The function sends and receives the data through the LNet protocol. This task must run parallel to the X2C_Update function.

The X2C_Update function, the connection of the peripherals and the model IOs will be explained in the next steps.

# VIII. CONNECT MODEL I/OS TO THE PERIPHERALS

## X2C Framework structure

To be able to use the generated code, the controller needs a typical configuration layer. These low level peripheral functions provided by Microchip (MCC, Harmony or user made), but the user have to take care to connect them to the X2C model.




## The generated project structure

Here is a short explanation of the generated files:

- mcc_generated_files\X2CCode**X2C.c**: This is generated from the xcos model, do not modify it manually.
- mcc_generated_files\X2CCode**X2CMain.c**: Contains the model update function call to calculate the module. Contains the Input and Output update functions that need to be manually modified.
- mcc_generated_files\X2CCode**X2CUtils.c**: Contains the X2C core and X2C Scope initialization. These functions are called in the mcc's *SYSTEM_Initialize(void)* method.
- mcc_generated_files\X2CCode**X2CComm.c**: This file is responsible to connect the LNet protocol and the hardware communication layer (e.g. Uart interfaces).

## Modify the X2CMain.c

### 1. Connect the inputs of the module

X2CMain.c contains *UpdateInports(void)*. Find the function definition, the comment section should contain a list of the available Input variables connected to the model. These variables must be manually updated. Here is an example, PORTA1 bit is read and converted to INT16 representation.

```
if (PORTA & 1) {
  Inports.Inport = INT16_MAX;
}else {
   Inports.Inport = 0;
 }
```

### 2. Connect outputs of the module

The model output variables are calculated every time, after the *X2C_Update()* function is called. The variables must read out manually, and the scaled value passed to the hardware peripheral interfaces.

***Note:*** *The output variables are pointers.*

```
if (*Outports.pOutport) {  // if model Outport different than zero
   LATB |= 1; // set LATB0
 } else {
   LATB &= ~1; // clear LATB0
 }
```

## Run the code on the hardware

Everything is ready to build the project and try it on the hardware. Just make and program the device as usually done in other projects.

# IX. Automatic compile and flash

The basic idea is that the focus during the development process is the model improvement. So once the peripherals and the hardware are configured well, then the development is focused only for the program logic. In this case, the program logic (mathematical control system) is the XCOS model. Hence a method would be great to compile the generated code and flash it automatically to the hardware, because the only change is in the generated X2C.c file so manual modification is not needed in the source code.

A Batch script is generated automatically during the MCC generation process. The Batch file is located in the project directory, e.g. *project.X/CompileAndFlash.bat.* This script does the compilation and flashes job automatically. So the user can focus on the model development and the rest will be done automatically. A little manual setup is needed.

First open the X2C Communicator as described in the previous sections. Then open the Communicator settings. There is a “Post-Generate:” option. That means the selected script file will be executed automatically after the code generated from the model. Select the automatically created *project.X/CompileAndFlash.bat* file.

The MPLAB X IDE can be closed from now, because the script will make and program the device.

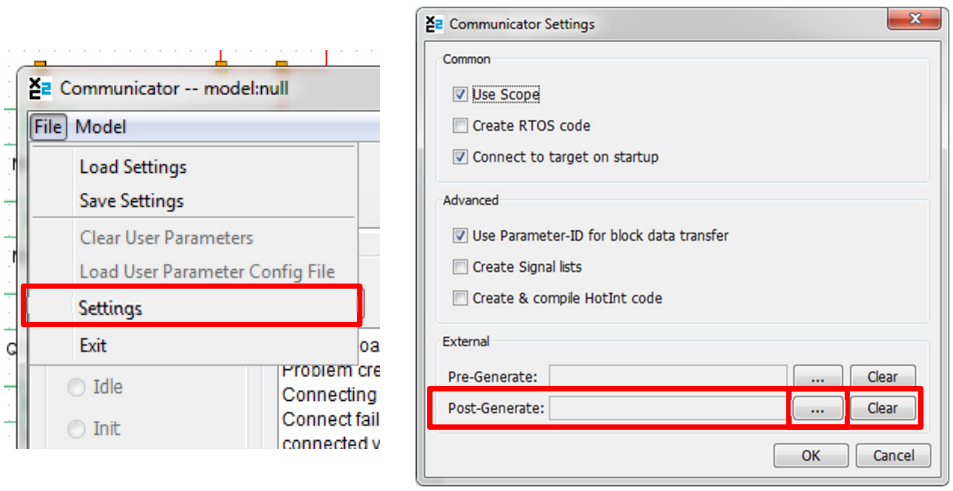
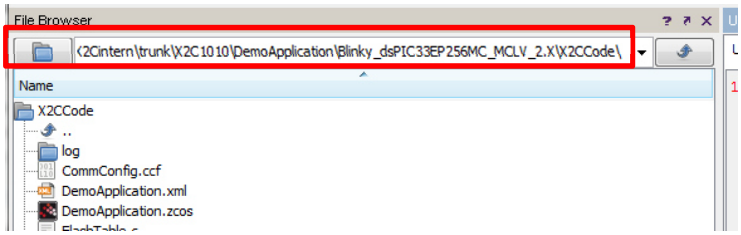



*Figure 13 – CompileAndFlash script setup*

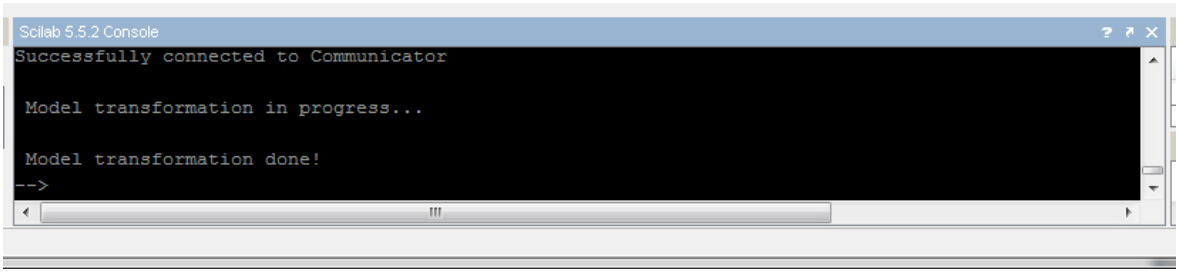
# TROUBLESHOOTING

### 1. Model transform failed:

a. Important! If the Scilab working path is not the same, then the code generation will not work!




b. Check Scilab output window for error messages.




c. Check the selected block implementations. The connected blocks must use the same input and output types.