LINZ CENTER OF MECHATRONICS GMBH

Getting Started with X2C

March 13, 2015

Contents

Part I
Installation

1 Software versions

Following software versions were tested for full *X2C* functionality:

Software	Version
Required:	
Scilab	5.5.1
Java Runtime Environment	6
Optional:	
MiKTeX	2.9
Texas Instruments Code Composer Studio	5.5.0.00077
Texas Instruments Code Generation Tools	6.1.6
Keil μVision	4.x
MPLAB X IDE	2.x

Different versions of these programs may work but without warranty.

2 Setup with Scilab/Xcos support

2.1 Installation

1. Open *Scilab/Xcos* and with the *File Browser* navigate to <X2C_ROOT>\System\Scilab\Scripts. Right click on **setup.sce** and click *Execute in Scilab*.
2. Restart *Scilab/Xcos*
3. The setup command creates a *X2C* configuration file which will automically load *X2C* libraries and palettes at startup of *Scilab/Xcos* .

2.2 Deinstallation

1. Open *Scilab/Xcos* and execute the command `initX2C(%f)` in the *Scilab/Xcos* console.
2. Restart *Scilab/Xcos*
3. Once above command was exeeceuted, the *X2C* configuration file is deleted and *Scilab/Xcos* will not load any *X2C* libraries or palettes anymore.

Part II
How-To

3 X2C code generation with Scilab/Xcos

The following section describes *X2C* code generation of a *Scilab/Xcos* Model. The description is based on the *Blinky* demo application used with the *TI Piccolo F28069 ControlSTICK* programmed with *Code Composer Studio* . However the following section works with other example projects such as *Microstick II* and *MPLAB X*or others as well. The only difference are the path variables of the project.

1. Open *Scilab/Xcos* and in the file browser navigate to your project directory (e.g. C:\projects\Blinky_(TI_Piccolo_F28069_controlSTICK)\X2CCode).

2. Double click on **DemoApplication.zcos**. The example project contains a few blocks used to demonstrate the basic function of *X2C* (see figure 1). The *Inport* and *Outport* blocks define the interface between the generated *X2C* code and the peripheral functions (e.g. ADC or GPIO Pins) on the target. For details about each blocks function read *X2Copen.Doc.pdf* in the documentation folder of the *X2C* directory.

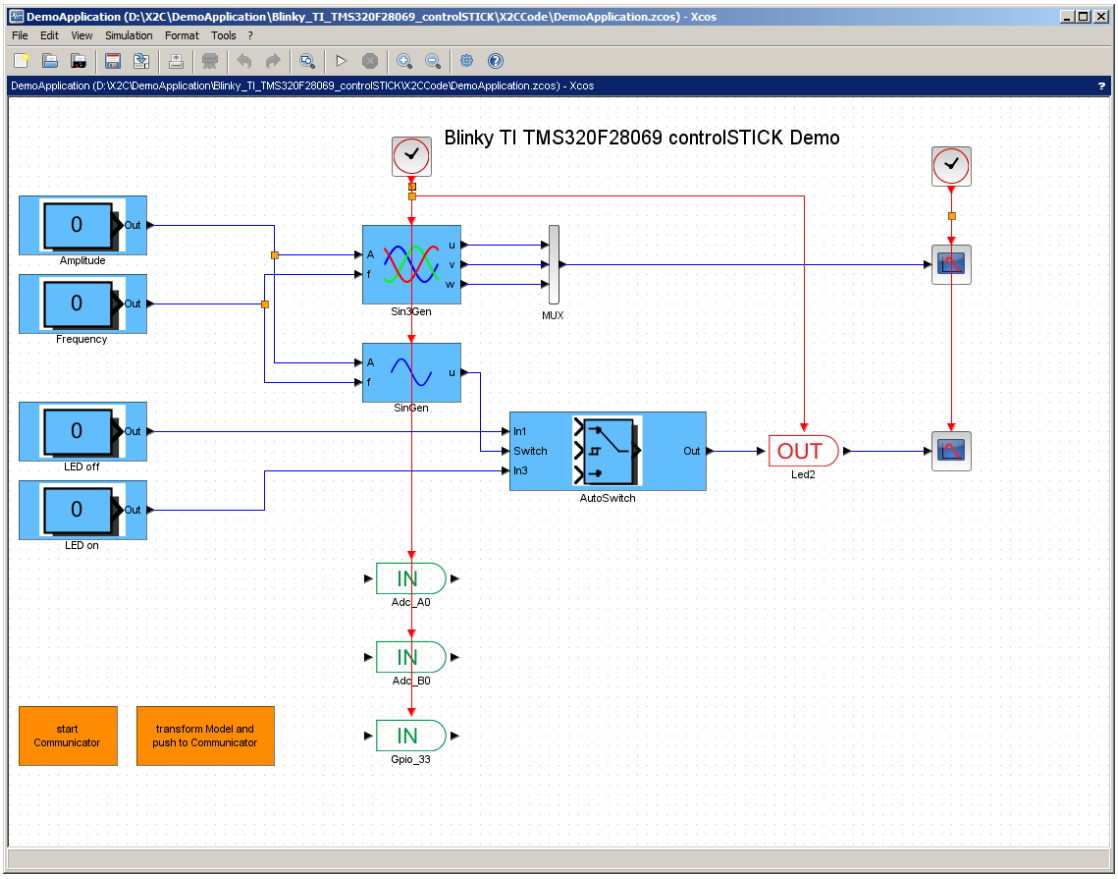

Figure 1: *Blinky* demo application in *Scilab/Xcos*

3. Double click on **start Communicator**. Some details of the current actions of the *Communicator* are shown in the *Log* area of the *Communicator* window.

Listing 1: Commandline *Scilab/Xcos* start *Communicator*

```
Starting Communicator >start ..\\X2C_2014-07-14_r
469_jenkins-X2C-341__open\\X2C\System\Scilab\run\x2c_start_communic
ator_con.cmd<
done
Successfully connected to Communicator
```

4. Double click on **Transform model and push to Communicator** and check the command line of *Scilab/Xcos* .

Listing 2: Commandline *Scilab/Xcos* Transform model and push to Communicator

```
Model transformation start
Model transformation done
```

5. Click **Create Code**. Now the files *X2C.h* and *X2C.c* are generated in the <PROJECT_ROOT>\X2CCode directory.

6. After code generation the Log Screen should look something like seen in listing 3. The C code of the *Scilab/Xcos* Model is now created.

Listing 3: Communicator Log Screen

```
Settings loaded
Connecting ...
Connect OK
*** DEVICE INFO ***
Target: TMS320F28069
Monitor date/time: 2014-07-16 15:44
Monitor version: 5
Application date/time: 2014-07-16 15:44
Application version: 1
connected via RMI
Model updated
Model XML file write: OK
Create code successful.
```

7. The *C* code for the *X2C* application is now created. Depending on the used target start the programming tool (e.g. *Code Composer Studio* or *MPLAB X*) and import the *Blinky* demo application project like described in section 4, or 5 respectively. Follow the instructions on how to configure and flash the project on the target.

4 Loading and building the demo application Blinky in Code Composer Studio

The demo application *Blinky* is build for the combination of the *TI Piccolo F28069 ControlSTICK* with the *TMS320F28069* processor.

1. Connect the *TI Piccolo F28069 ControlSTICK* with the computer.
2. With your OS file browser navigate to the *DemoApplication* folder in your *X2C* directory <X2C_ROOT>\DemoApplication\Blinky_(TI_Piccolo_F28069_controlSTICK).
3. Copy the project folder to a choosen location (e.g. C:\projects\Blinky_(TI_Piccolo_F28069_controlSTICK)).
4. Open *Code Composer Studio* (choose workspace directory as you like). Now click **Project** → **Import Existing CCS Eclipse Project**. Browse to the location of the *Blinky* project . Click **Finish** to import the project.
5. In the *Code Composer Studio* file structure of the *Blinky* demo project there are two virtual folders *Blocks* and *Core* which should be linked directly to the *X2C* directory. To ensure this go to **Project** → **Properties** drop down **Resource** and click **Linked Resources**. Double click on folder **X2C_ROOT** and set the correct link (e.g. <X2C_ROOT>). After hitting **OK** two times there should not be any warnings signs (like shown in figure 2) beside the Icons in the Blocks and Core folders.

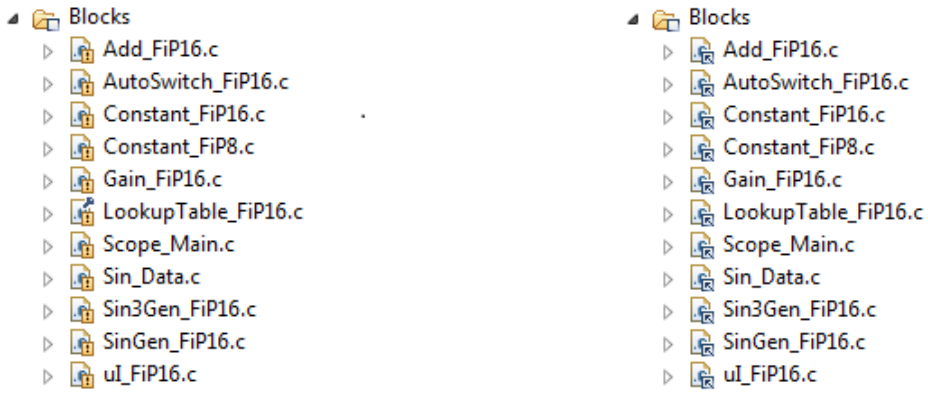

Figure 2: *Code Composer Studio* invalid (left) and valid (right) *X2C* root directory

6. The generated code from *X2C* is located in the folder ../X2CCode. To check if code generation went fine go to the X2CCode folder and open *X2C.c*. Make sure time and date of code generation is plausible.
7. Build the project in *Code Composer Studio* by clicking **Project** → **Build all** or by clicking on the **Hammer** symbol as seen in figure 3 at the top of the screen. Check for errors while building in the console at the bottom of the screen.

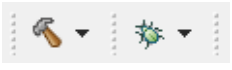

Figure 3: *Code Composer Studio* build and debug buttons

8. If your target is connected to the computer click **Run** → **Debug** or click on the *Bug* symbol as seen in figure 3 at the top. The Program is now transferred to the target an can be started with the **green arrow** button at the top.

9. After starting the program the on-board LED of the *Piccolo controlSTICK* should be blinking!

5 Loading and building the demo application Blinky in MPLAB X

The demo application *Blinky* is build for the combination of the *Microstick II* with the *dsPIC33FJ128MC802* processor and the *Microstick Plus* developer board (Details www.microstick.com).
Info: While flashing new code only the *Microstick II* needs to be connected with the computer.

1. Connect the *Microstick II* with the computer.
2. Open *MPLAB X* and click **File** → **Open Project**. Browse to the location of the *Blinky* demo application in the *X2C* directory <X2C_ROOT>\DemoApplication\...\Blinky_(Microchip_dsPIC33Fxxxx_MicrostickPlus)). Click **Open Project**.
3. The project called *Blinky_dsPIC33Fxxxx_MicrostickPlus* appears in the *Projects* area. If you want to move the project to another directory right click on the **Projectname** → **Move**. Choose the new project location and click **Move**. When moving a project in *MPLAB X* with the *Move* function, *MPLAB X* automatically updates the path variables for the source files. However the path variables for the header include files need to be updated.
4. After moving there might show up an error during building because some path variables in the compilers settings refer to wrong directories.
5. To ensure the compiler uses the correct path variables right click on the **Projectname** → **Properties** → **XC16 Global Options** → **xc16-gcc**. In the drop down menu **Option categories** choose **Preprocessing and messages**. Click on the dots beside *C include dirs*. There are relative paths to the needed include files listed as seen in figure 4. Correct the links by double clicking on the path variables.
   Info: Only the links to the *Library* and *Controller* path need to be updated (as seen in figure 5).

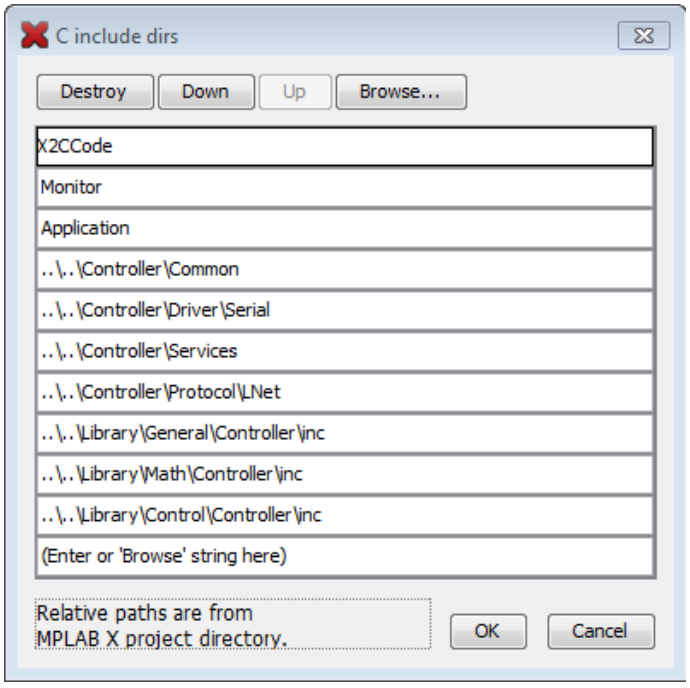

Figure 4: Default path variables for the include files

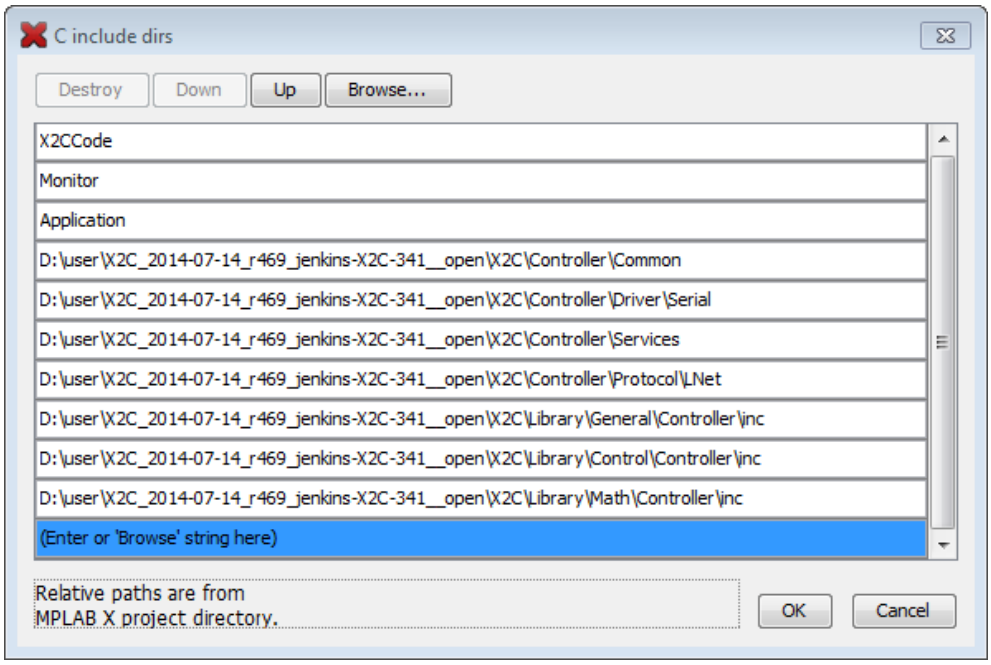

Figure 5: Custom path variables for the include files

6. Go to **Run** → **Clean and Build Main Project** or click the *hammer with brush* button as seen in figure 6. After building there should be a message `BUILD SUCCESSFUL` in the message area at the bottom of the screen.

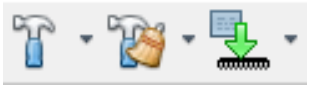

Figure 6: *MPLAB X Clean and Build Main Project* button

7. If the build process was successful go to **Run** → **Run Main Project** or click the *Green Arrow* button as seen in figure 6. If there is a message similar to *MICROSTICK not Found* try to select the *Starter Kits (PKOB)* tool as shown in figure 7.

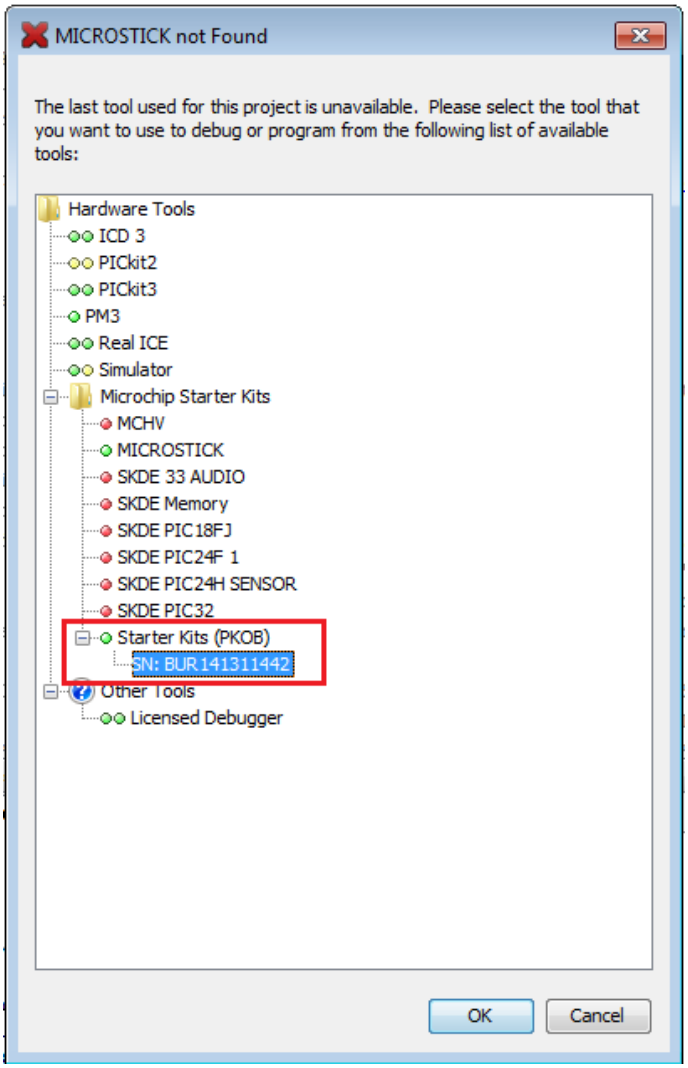

Figure 7: Tool selection before programming the device

8. After starting the program the LED (RB12) on the *Microstick Plus Board* should be blinking!