Getting Started with X2C

March 3, 2017

Contents

Part I

Installation

1 Software versions

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

Software	**Version**
Required:	
Scilab (www.scilab.org)	5.5.x
Java Runtime Environment	6
Optional (for documentation):	
MiKTeX (www.miktex.org)	2.9
Doxygen (www.doxygen.org)	1.8.10
Graphviz (www.graphviz.org)	2.38
Optional (for programming):	
Texas Instruments Code Composer Studio	5.5.x
Texas Instruments Code Generation Tools	6.1.6
Keil μVision	4.x
Microchip MPLAB X IDE	3.x
Microchip Compiler XC16	1.25

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 exeecuted, the *X2C* configuration file is deleted and *Scilab/Xcos* will not load any *X2C* libraries or palettes anymore.

For the unlikely event that Scilab freezes at startup and remains in a deadlock state, the deinstallation can be done manually by deleting the file **scilab.ini** located in the Scilab home directory (for Windows typically C:\Users\<your user name>\AppData\Roaming\Scilab\scilab-5.x.x).

Part II

How-To

3 *X2C* code generation with *Scilab/Xcos*

The following section describes *X2C* code generation of a *Scilab/Xcos* model based on the *Blinky* demo application.

1. Open *Scilab/Xcos* and in the file browser navigate to your project directory (e.g. <X2C_ROOT>\DemoApplication\Blinky_TI_TMS320F28069_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 block function read *X2Copen.Doc.pdf* in the documentation folder of the *X2C* directory.

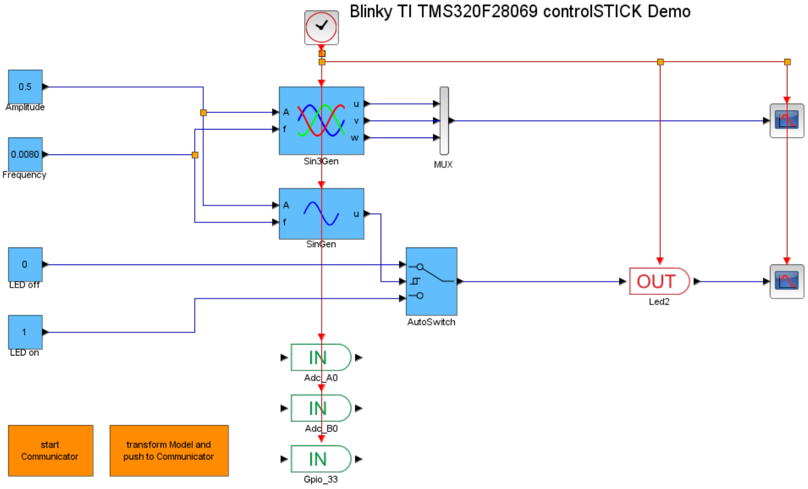


Figure 1: *Blinky* demo application in *Scilab/Xcos*

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 and the *Scilab/Xcos* command line:

```
Starting Communicator
done
Successfully connected to Communicator
```

4. Double click on **Transform model and push to Communicator** and check the pop-up window for the end of the transformation process.

5. Click **Create Code** in the *Communicator*. Now the files *X2C.h* and *X2C.c* are generated in the <PROJECT_ROOT>\X2CCode directory and the Log screen should contain the lines:

```
[...]
Model updated
Model XML file write: OK
Create code successful.
```

6. The *C* code for the *X2C* application has been 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 as 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 intended to be used with *TI F28069 Piccolo controlSTICK*.

1. Connect the *TI F28069 Piccolo controlSTICK* with the computer.

2. 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 (<X2C_ROOT>\DemoApplication\Blinky_TI_TMS320F28069_controlSTICK). Click **Finish** to import the project.

3. 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 to your X2C installation directory (<X2C_ROOT>). After hitting **OK** two times there should not be any warning signs (like shown in figure 2) at the icons for the linked files in the Blocks and Core folders.

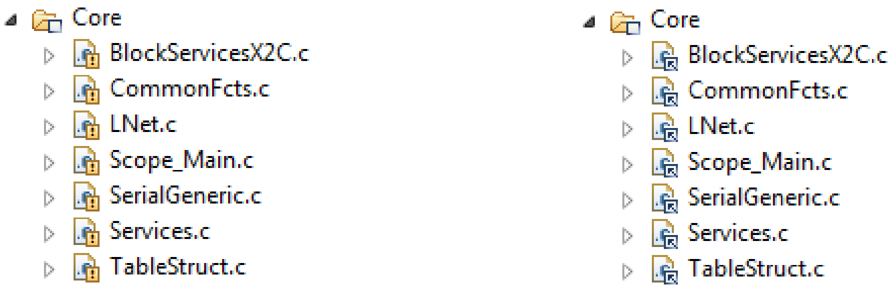


Figure 2: *Code Composer Studio* invalid (left) and valid (right) *X2C* root directory

4. The generated code from *X2C* is located in the folder <X2C_ROOT>\DemoApplication\Blinky_TI_TMS320F28069_controlSTICK\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.

5. 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.

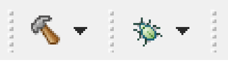


Figure 3: *Code Composer Studio* build and debug buttons

6. 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 and can be started with the **green arrow** button at the top.

7. After starting the program the on-board LED of the *TI F28069 Piccolo controlSTICK* should be blinking!

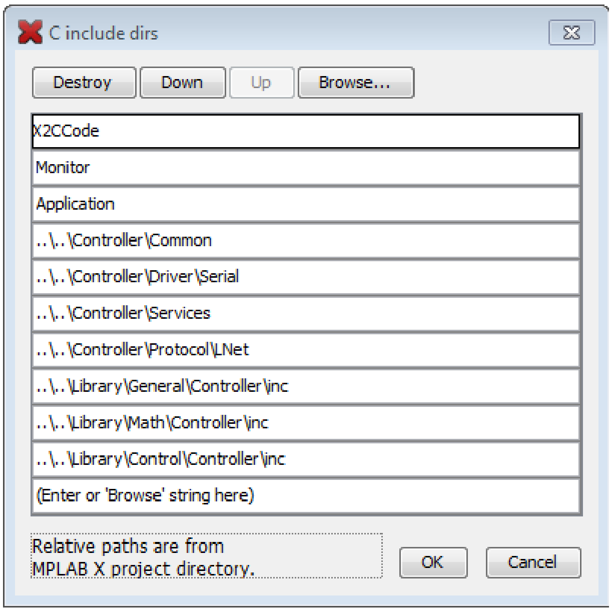
5 Loading and building the demo application Blinky in *MPLAB X*

The demo application *Blinky* is build for the combination of the *Microstick II* with the *dsPIC33FJ128MC802* processor and the *MicrostickPlus* developer board (for details see `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. In the case the demo application is copied/moved to a different location, the include paths have to be adapted. 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.



X2CCode
Monitor
Application
..\..\Controller\Common
..\..\Controller\Driver\Serial
..\..\Controller\Services
..\..\Controller\Protocol\LNet
..\..\Library\General\Controller\inc
..\..\Library\Math\Controller\inc
..\..\Library\Control\Controller\inc
(Enter or 'Browse' string here)

Relative paths are from MPLAB X project directory.

Figure 4: Default path variables for the include files

4. Go to **Run** → **Clean and Build Main Project** or click the *hammer with brush* button as seen in figure 5. After building there should be a message `BUILD SUCCESSFUL` in the message area at the bottom of the screen.

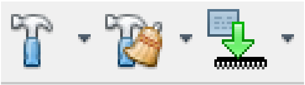


Figure 5: *MPLAB X Clean and Build Main Project* button

5. If the build process was successful go to **Run** → **Run Main Project** or click the *Green Arrow* button as seen in figure 5. If there is a message similar to *MICROSTICK not Found* try to select the *Starter Kits (PKOB)* item which represents your board.

6. After starting the program the LED (RB12) on the *MicrostickPlus Board* should be blinking!

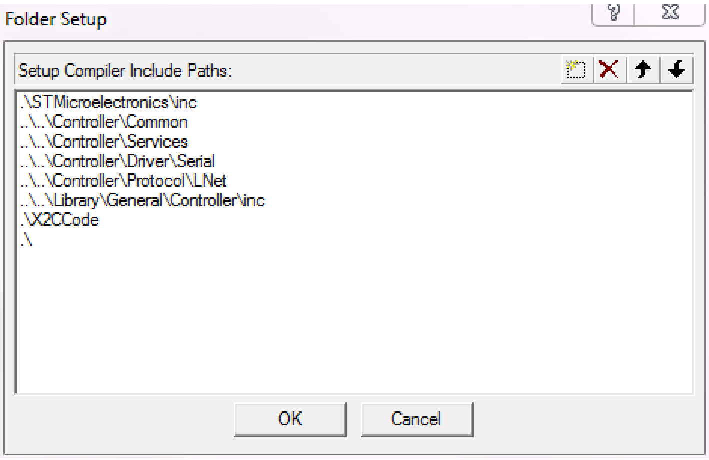
6 Loading and building the demo application Blinky in *Keil μVision*

The demo application *Blinky* is intended to be used with the *ST STM32F051R8 Discovery kit*.

1. Connect the *ST STM32F051R8 Discovery kit* with the computer. You may have to install the ST-Link USB driver (available on www.stm.com) to get the board recognized by your operating system.

2. Open *Keil μVision* and click **Project** → **Open Project**. Browse to the location of the *Blinky* project (<X2C_ROOT>\DemoApplication\Blinky_ST_STM32F051R8_Discovery). Click **Open** to import the project.

3. In the *Keil μVision* file structure of the *Blinky* demo project there are two virtual folders *Blocks* and *Core*, which are linked relatively to the *X2C* directory. If the *Blinky* demo project is copied/moved to a different location, the include paths as seen in figure 6 have to be adapted.



```
.\STMicroelectronics\inc
..\..\Controller\Common
..\..\Controller\Services
..\..\Controller\Driver\Serial
..\..\Controller\Protocol\LNet
..\..\Library\General\Controller\inc
.\X2CCode
.\
```

Figure 6: *Keil μVision* include paths setting

   To open shown window go to **Project** → **Options for target 'Blinky Demo'** change to tab **C/C++** and click **...** next to the include paths text field.

4. The generated code from *X2C* is located in the folder <X2C_ROOT>\DemoApplication\Blinky_ST_STM32F051R8_Discovery\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.

5. Build the project in *Code Composer Studio* by clicking **Project** → **Build target** or by clicking on the *Build* symbol as seen in figure 7 at the top left of the *Keil μVision* screen. Check for errors while building in the console at the bottom of the screen.

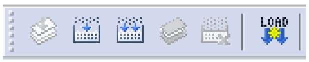


Figure 7: *Keil μVision* build and load buttons

6. If your target is connected to the computer click **Flash** → **Download** or click on the *Download* symbol as seen in figure 7 at the top left of the *Keil μVision* screen. The program is now transferred to the target and is automatically started.

7. After starting the program the green on-board LED of the *ST STM32F051R8 Discovery kit* should be blinking!
8. To use *X2C Communicator* and *Scope* the computer has to be connected via serial interface to *ST STM32F051R8 Discovery kit*. For this reason a TTL-level compatible RS-232 adapter should be connected to pin PA9 - TxD, PA10 - RxD and GND.