(/xwiki/bin/view/Main/) / MASTERs (/xwiki/wiki/masters/view/Main/)
/ MASTERs 2025 Lab Manual Developement (/xwiki/wiki/masters/view/masters-2025-lab-manuals/)
/ 25085 MTR4 - Developing Motor Control Applications using Model Based Design (/xwiki/wiki/masters/view/masters-2025-lab-manuals/25085/)
/ Lab 1 - Model-Based Design – “HelloWorld” (/xwiki/wiki/masters/view/masters-2025-lab-manuals/25085/lab1/)

Lab 1 - Model-Based Design – “HelloWorld”

Last modified by Mark Reynolds (/xwiki/bin/view/XWiki/MarkReynolds) on 2025/07/22 11:49

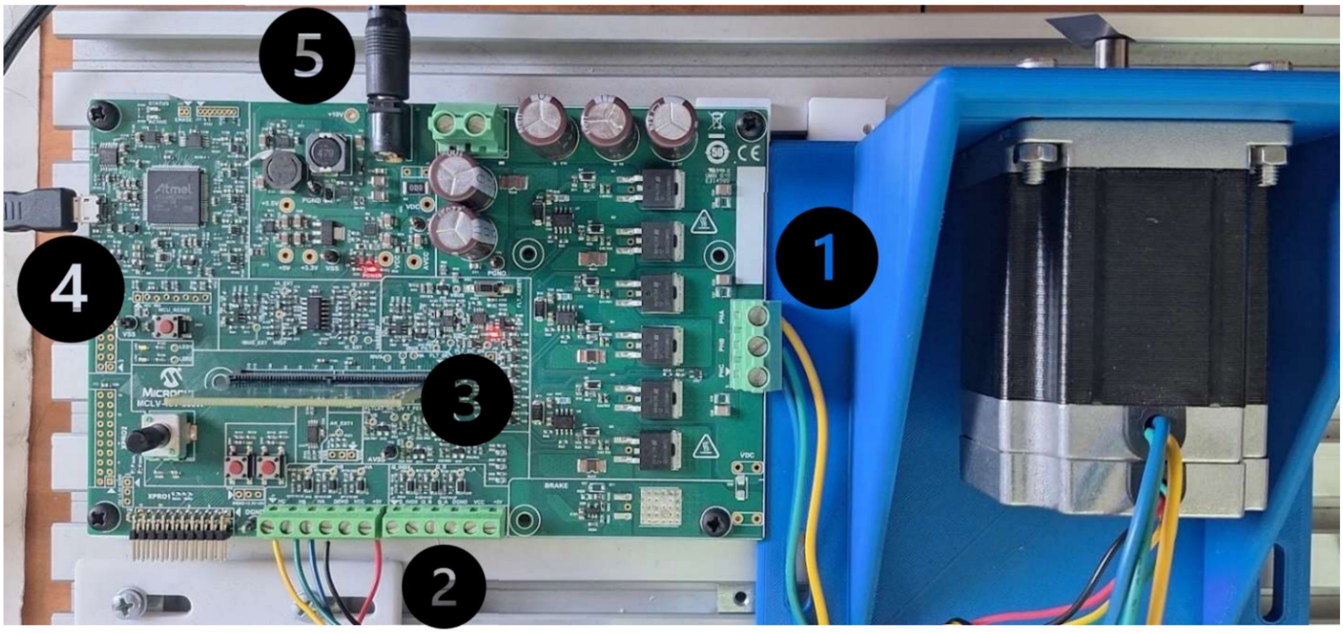
Purpose

Embedded systems hello world applications are typically a blinking LED. The goal of this lab is to simulate a blinking LED application in a model-based development environment. An instruction is given to modify the model and run a simulation. Then the final step is the verification of the designed model on a real hardware (HW) environment.

Overview of the HW Setup

The motor does not need to be connected for the first lab, but the rest of the labs will need the motor to be connected. The connected motor does not interfere with the actual blinky lab activity, therefore it can be kept connected for simplicity.

1. Motor winding connections (check the wiring order)
2. Motor HALL sensors connections (check the wiring order)
3. Dual-Inline-Module (DIM) module with dsPIC33AK128MC106
4. PICkit™ On-Board and Universal Serial Bus (USB)/Universal Asynchronous Receiver-Transmitter (UART) converter
5. 24V power supply



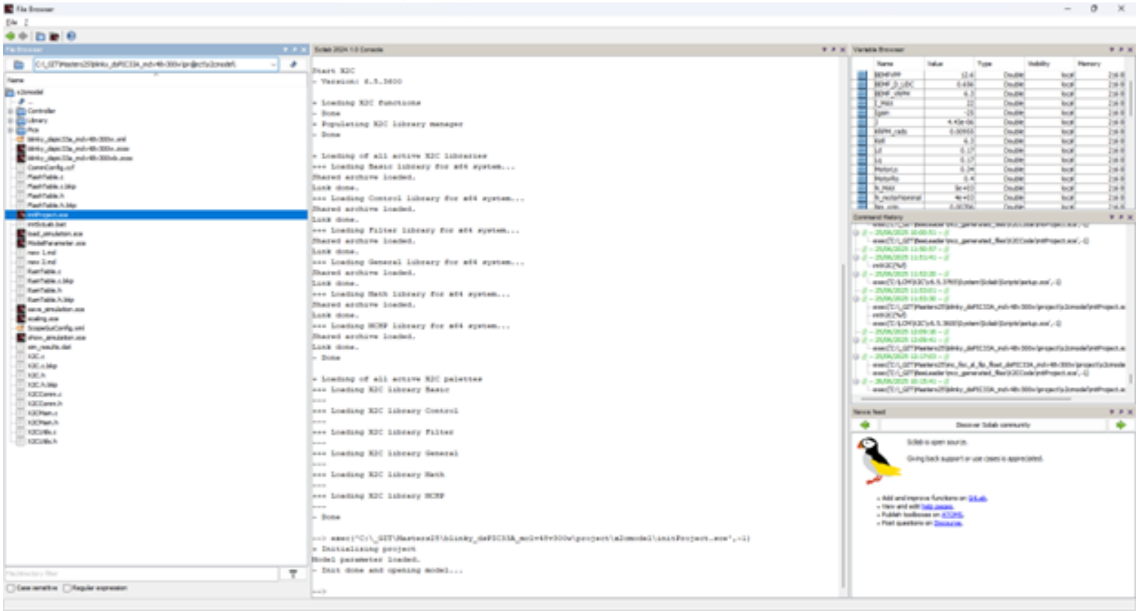
Procedure

Lab 1 Part 1: Simulation

1 Open the Project and the Model

1.1 Open the **blinky_dsPIC33A_mclv48v300w.X** project in MPLAB® X IDE.

1.2 Open Scilab® 2024.1.0 (Desktop)

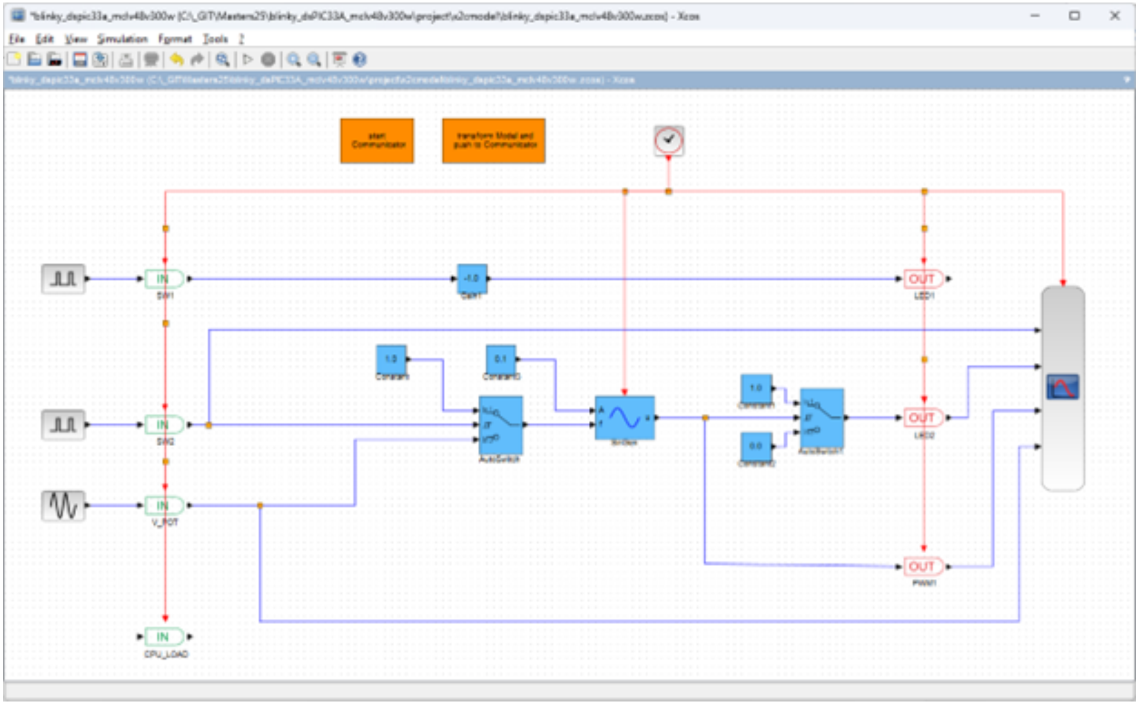


```
Start X2C
- Version: 6.5.3600

* Loading X2C functions
- Done
* Populating X2C library manager
- Done

* Loading of all active X2C libraries
+++ Loading Basic library for x64 system...
Shared archive loaded.
Link done.
+++ Loading Control library for x64 system...
Shared archive loaded.
Link done.
+++ Loading Filter library for x64 system...
Shared archive loaded.
Link done.
+++ Loading General library for x64 system...
Shared archive loaded.
Link done.
+++ Loading Math library for x64 system...
Shared archive loaded.
Link done.
+++ Loading MCHP library for x64 system...
Shared archive loaded.
Link done.
- Done

* Loading of all active X2C palettes
+++ Loading X2C library Basic
---
+++ Loading X2C library Control
---
+++ Loading X2C library Filter
---
+++ Loading X2C library General
---
+++ Loading X2C library Math
---
+++ Loading X2C library MCHP
---
- Done

--> exec('C:\_GIT\Masters25\blinky_dsPIC33A_mclv48v300w\project\x2cmodel\initProject.sce',-1)
* Initializing project
Model parameter loaded.
- Init done and opening model...

-->
```

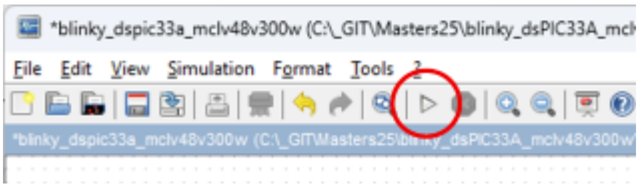
1.3 Navigate with Scilab into the model directory C:\...\blinky_dsPIC33A_mclv48v300w\project\x2cmodel

1.4 Execute the initProject.sce file

1.5 The model opens:

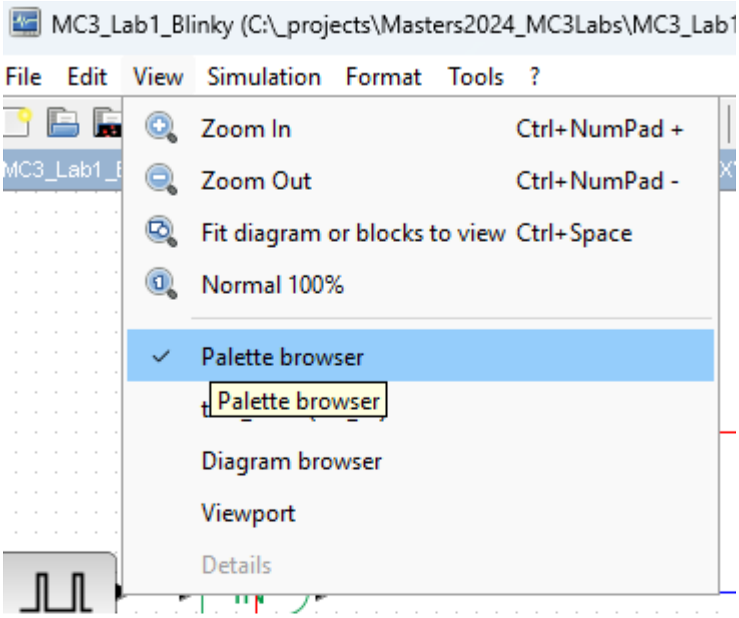

*blinky_dspic33a_mclv48v300w (C:_GIT\Masters25\blinky_dsPIC33A_mclv48v300w\project\x2cmodel\blinky_dspic33a_mclv48v300w.zcos) - Xcos
File Edit View Simulation Format Tools ?
start Communicator
transform Model and push to Communicator
IN
SW1
-1.0
Gain
OUT
LED1
1.0
Constant
0.1
Constant1
AutoSwitch
SinGen
Constant3
Constant2
AutoSwitch1
IN
SW2
OUT
LED2
IN
V_POT
OUT
PWM1
IN
CPU_LOAD

1.6 Simulate and play with parameters:

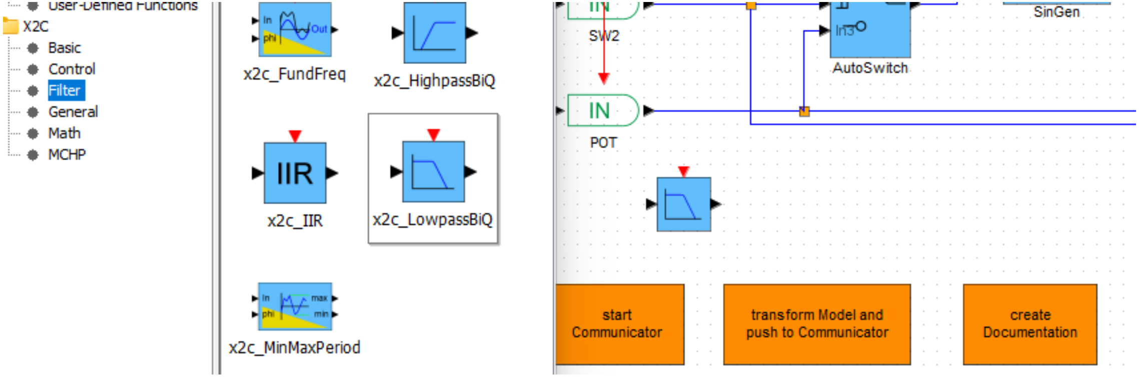

*blinky_dspic33a_mclv48v300w (C:_GIT\Masters25\blinky_dsPIC33A_mcl
File Edit View Simulation Format Tools ?
*blinky_dspic33a_mclv48v300w (C:_GIT\Masters25\blinky_dsPIC33A_mclv48v300w

2 Modifying the Model

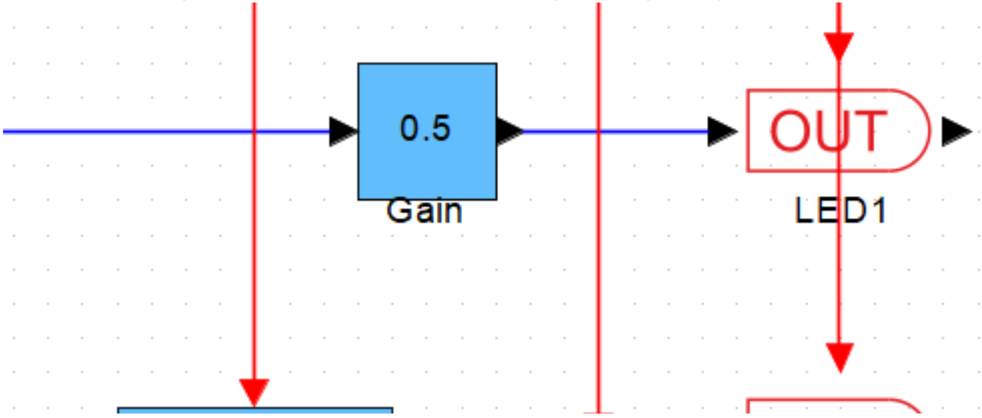
2.1 Navigate to the **Palette browser**.

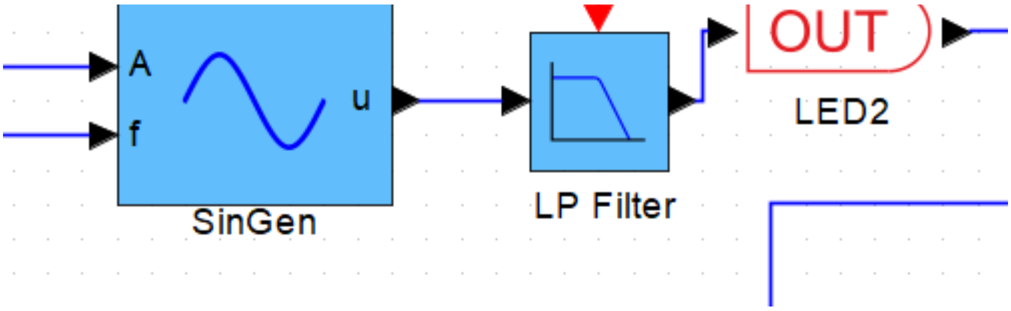

MC3_Lab1_Blinky (C:_projects\Masters2024_MC3Labs\MC3_Lab1
File Edit View Simulation Format Tools ?
Zoom In Ctrl+NumPad +
Zoom Out Ctrl+NumPad -
Fit diagram or blocks to view Ctrl+Space
Normal 100%
Palette browser
Palette browser
Diagram browser
Viewport
Details

2.2 Drag and drop the low pass filter block from the block library to the model.



2.3 Connect the block ports as shown in the accompanying image.



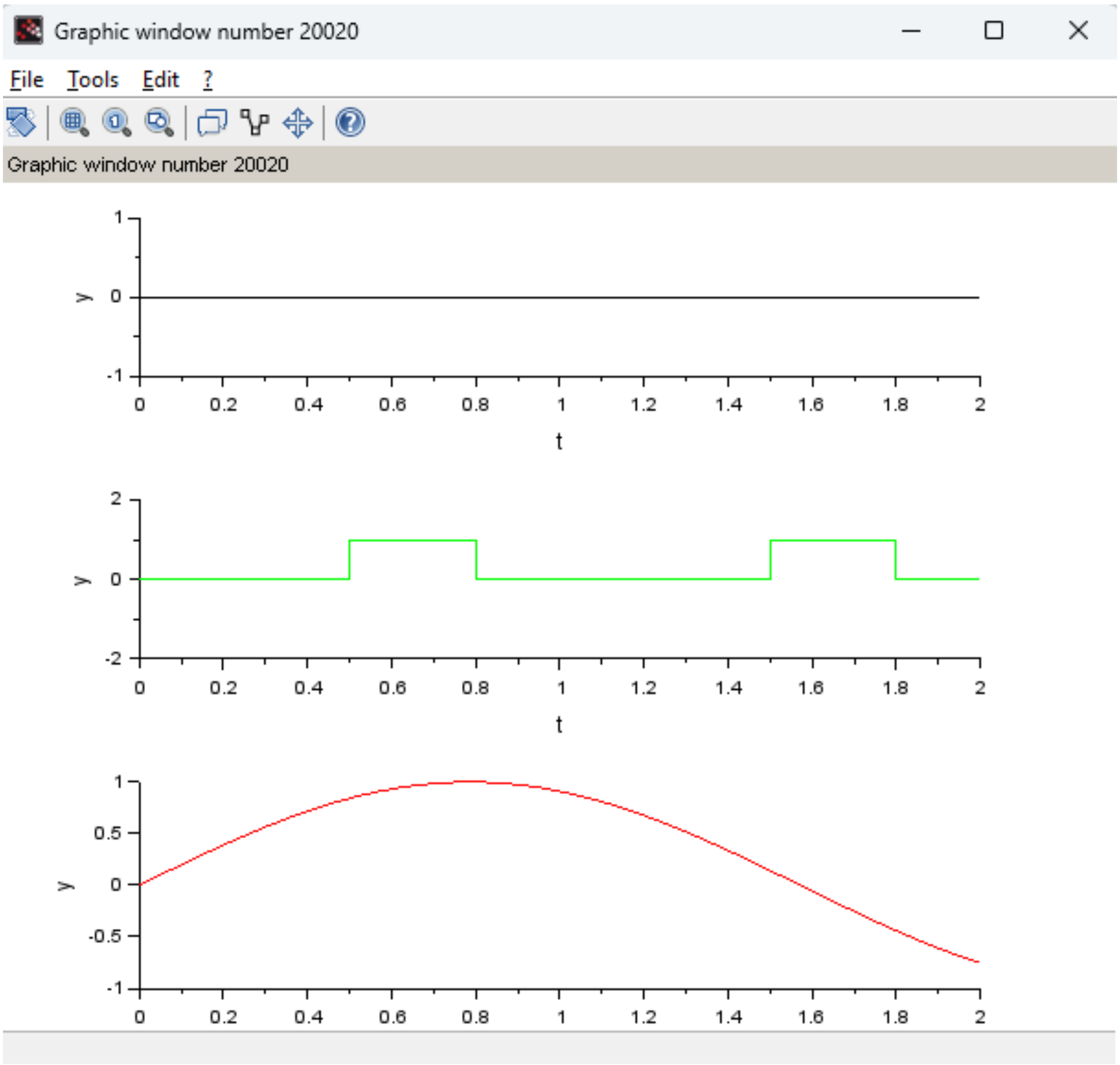


2.4 Optionally, label the block “LP Filter”. Select the block and press **F2**.

> The block label and block names are different! The name must be unique, whereas the label is simply a descriptor. Double-clicking on the block opens the block parameters, allowing you to rename it.

3 Simulation

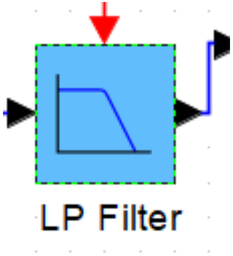
3.1 Click the **play** button ▷ to start the simulation.



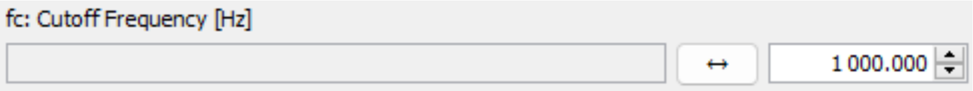
3.2 After the simulation is done, the scope window should open automatically.

4 Modify Block Parameters

4.1 Double-click on **LP Filter** block. It opens the block parameter window.

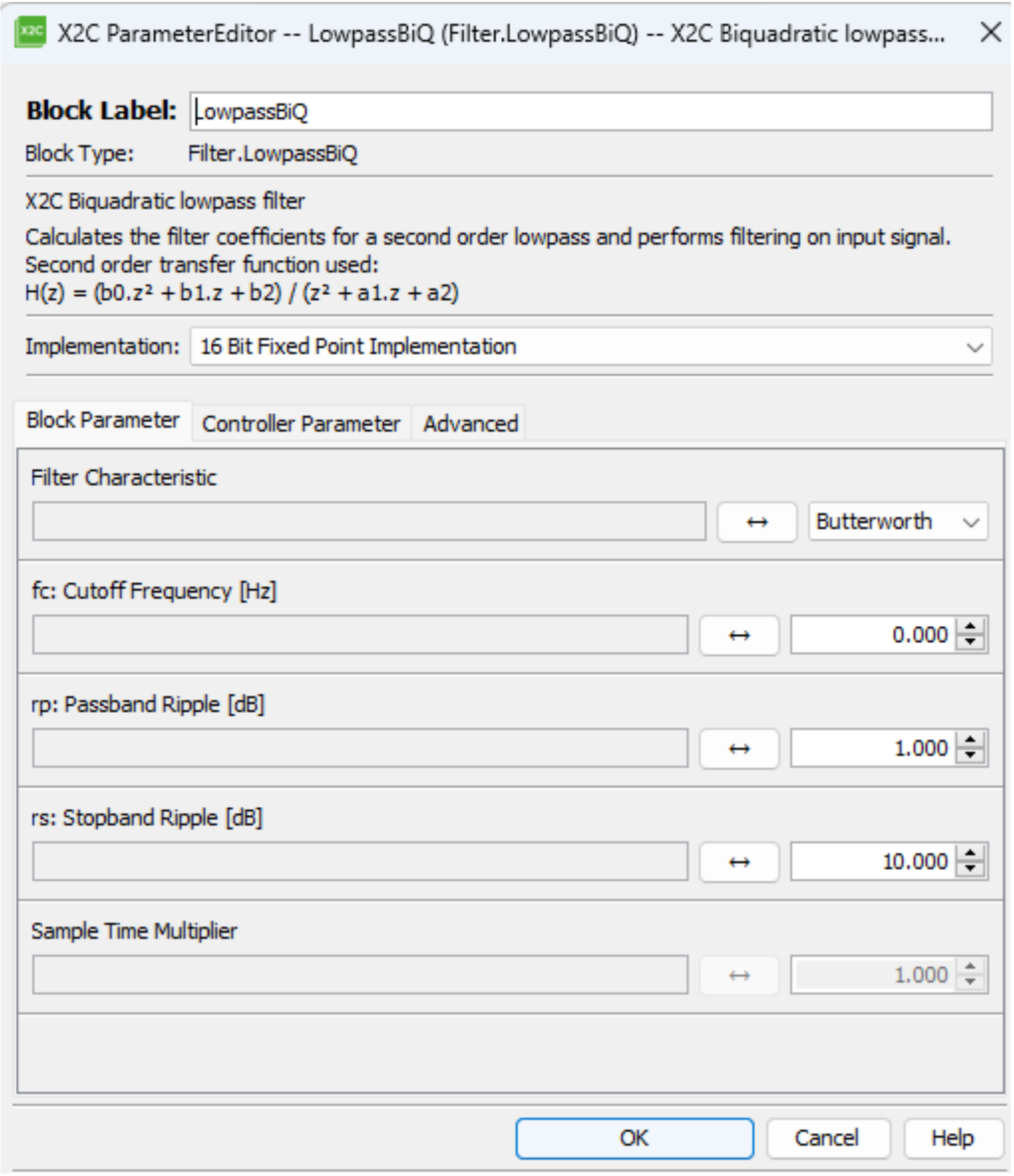


4.2 Modify **Cutoff Frequency** to **1000Hz**.



fc: Cutoff Frequency [Hz] ↔ 1 000.000

4.3 Apply the changes by clicking **OK**.



X2C ParameterEditor -- LowpassBiQ (Filter.LowpassBiQ) -- X2C Biquadratic lowpass...

Block Label: LowpassBiQ

Block Type: Filter.LowpassBiQ

X2C Biquadratic lowpass filter

Calculates the filter coefficients for a second order lowpass and performs filtering on input signal.
Second order transfer function used:
H(z) = (b0.z² + b1.z + b2) / (z² + a1.z + a2)

Implementation: 16 Bit Fixed Point Implementation

Filter Characteristic ↔ Butterworth

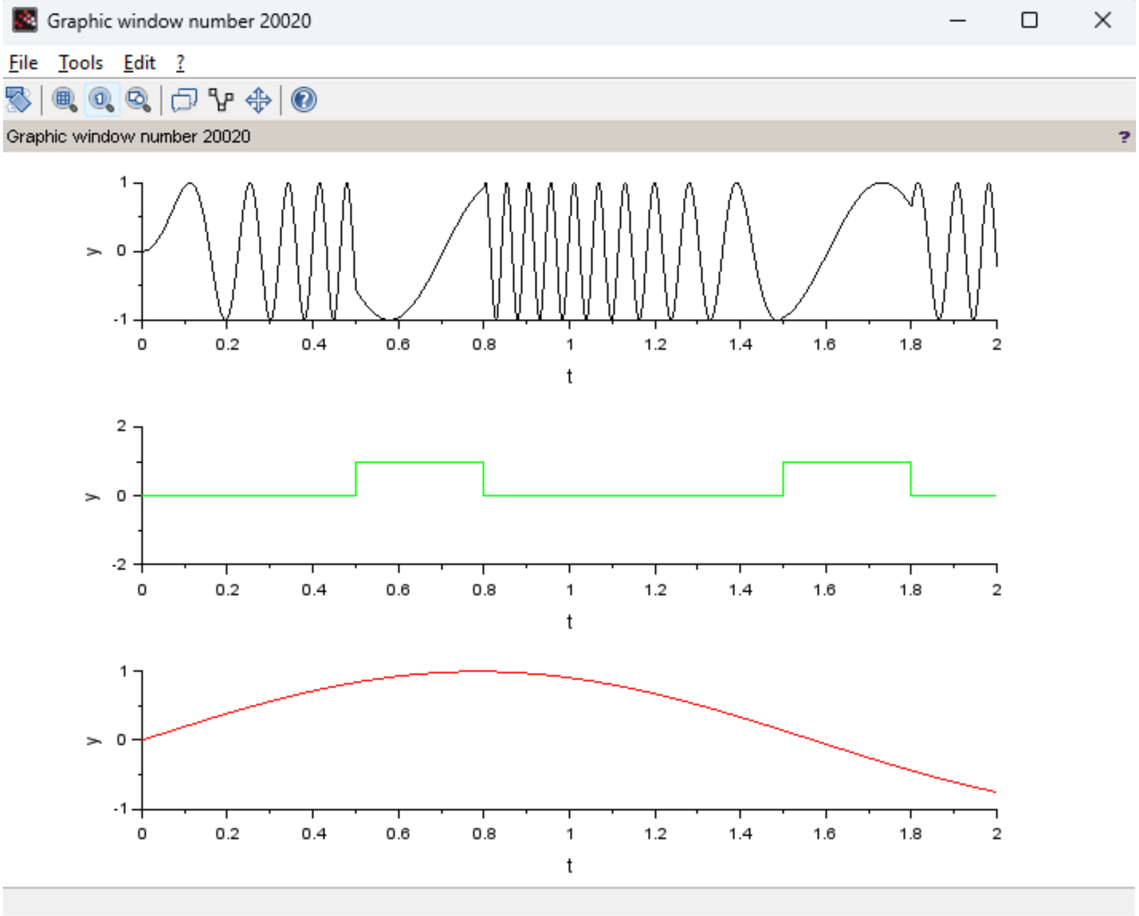
fc: Cutoff Frequency [Hz] ↔ 0.000

rp: Passband Ripple [dB] ↔ 1.000

rs: Stopband Ripple [dB] ↔ 10.000

Sample Time Multiplier ↔ 1.000

4.4 Run the simulation again and observe the new output:



5 Experiment Simulation Features

> The following steps are optional and can be used to experiment with different scilab features for simulation.

5.1 Try to change base frequency of the sine generator, **SineGen** block.

5.2 Add new scope to the simulation, “CSCOPE”.

5.3 Extend the simulation time. Note the scope window overflow.

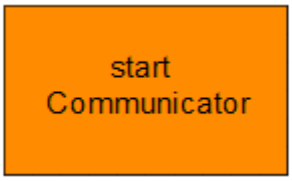
5.4 Extend the scope window to fit the longer simulation time. See Scilab's help page, "*CMSCOPE (https://help.scilab.org/CMSCOPE.html)* ."

5.5 Add new scope input.

Lab 1 Part 2: Run the Model on the HW

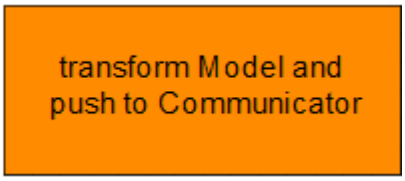
1 Generating Code

1.1 Double-click on the **start Communicator** block.



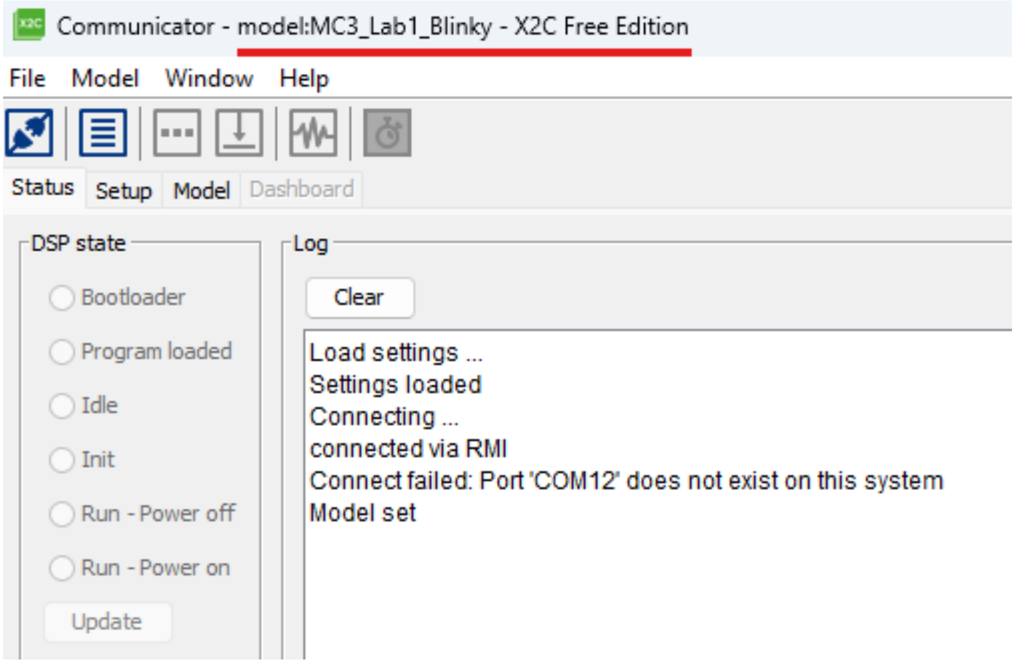
start Communicator

1.2 Wait for the communicator to start then double-click **transform Model and push to Communicator**.



transform Model and push to Communicator

1.3 Verify that the communicator and the Scilab model are connected via Remote Method Invocation (RMI). To do so, check that the Communicator window title shows the model name.



Communicator - model:MC3_Lab1_Blinky - X2C Free Edition

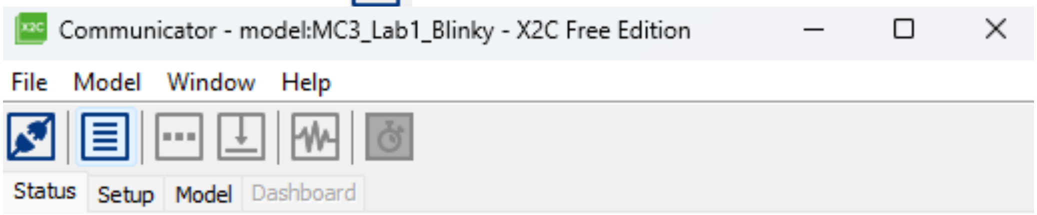
```
Load settings ...
Settings loaded
Connecting ...
connected via RMI
Connect failed: Port 'COM12' does not exist on this system
Model set
```

If the Communicator *does not* show the model name in the title:

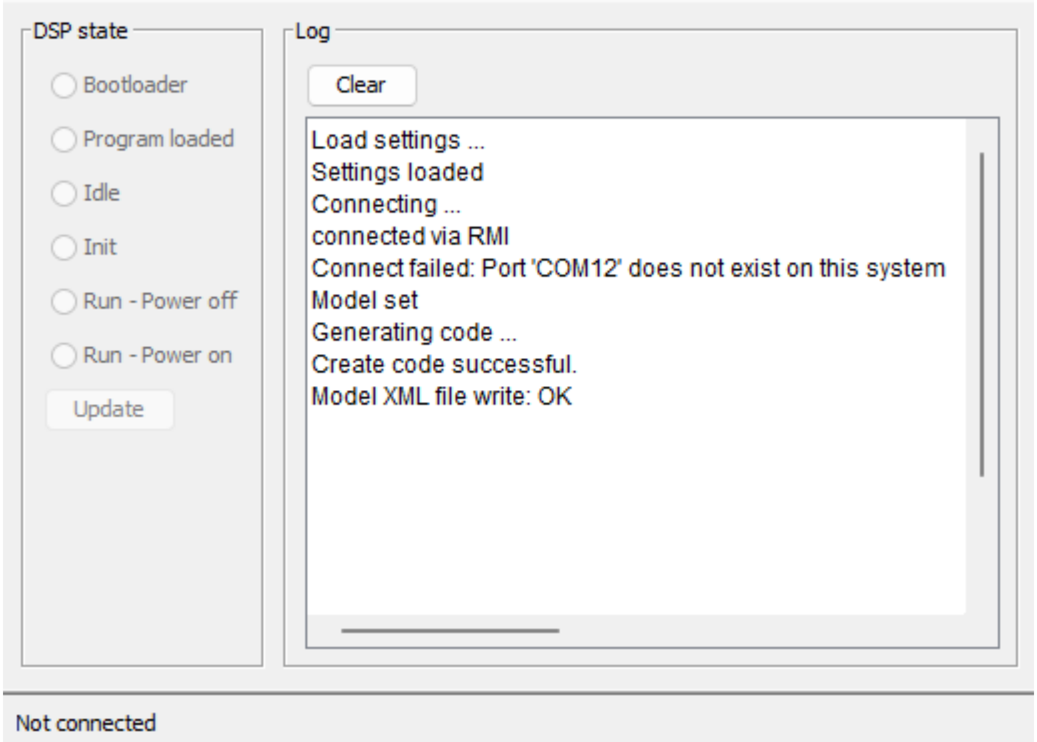
1. Close the **Communicator**.
2. Start the process from Step 1.1.

> ❗ Only one instance of X2C Communicator is allowed to run. Do not start the Communicator twice!

1.4 Generate code by clicking the ☰ button.



Communicator - model:MC3_Lab1_Blinky - X2C Free Edition



```
Load settings ...
Settings loaded
Connecting ...
connected via RMI
Connect failed: Port 'COM12' does not exist on this system
Model set
Generating code ...
Create code successful.
Model XML file write: OK
```

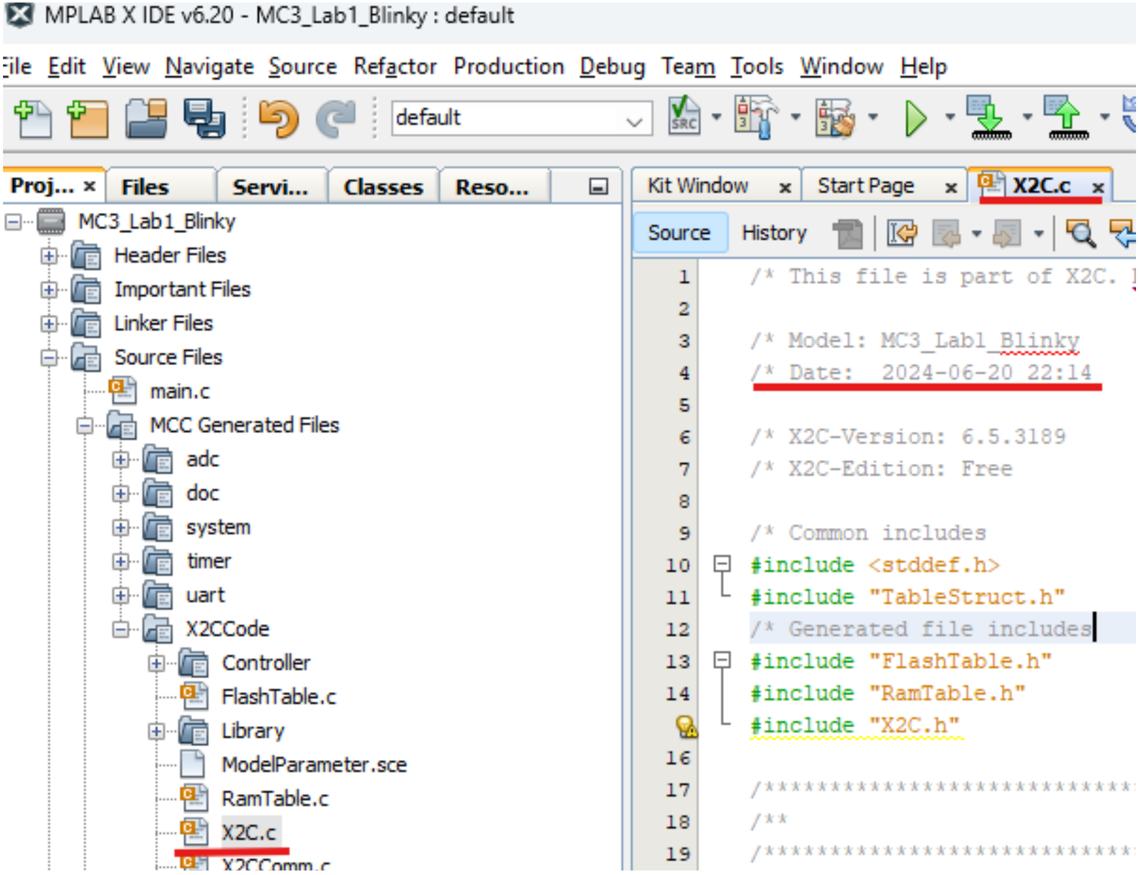
Not connected

1.5 Verify in the Communicator output window that code generation was successful.

2 Compile and Program the MCLV48V Board

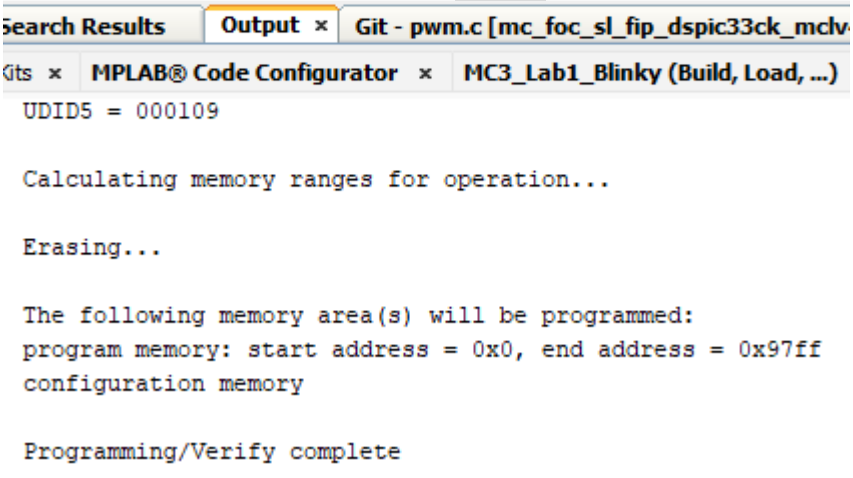
2.1 Go back to **MPLAB X IDE**.

2.2 Locate and open the **X2C.c** file and check the generated time stamp at line 4. It should match the actual time as the recently generated file.



```
1   /* This file is part of X2C.
2
3   /* Model: MC3_Lab1_Blinky
4   /* Date:  2024-06-20 22:14
5
6   /* X2C-Version: 6.5.3189
7   /* X2C-Edition: Free
8
9   /* Common includes
10  #include <stddef.h>
11  #include "TableStruct.h"
12  /* Generated file includes
13  #include "FlashTable.h"
14  #include "RamTable.h"
    #include "X2C.h"
16
17  /*****************************
18  /**
19  /*****************************
```

2.3 Compile and program the HW by clicking the button.

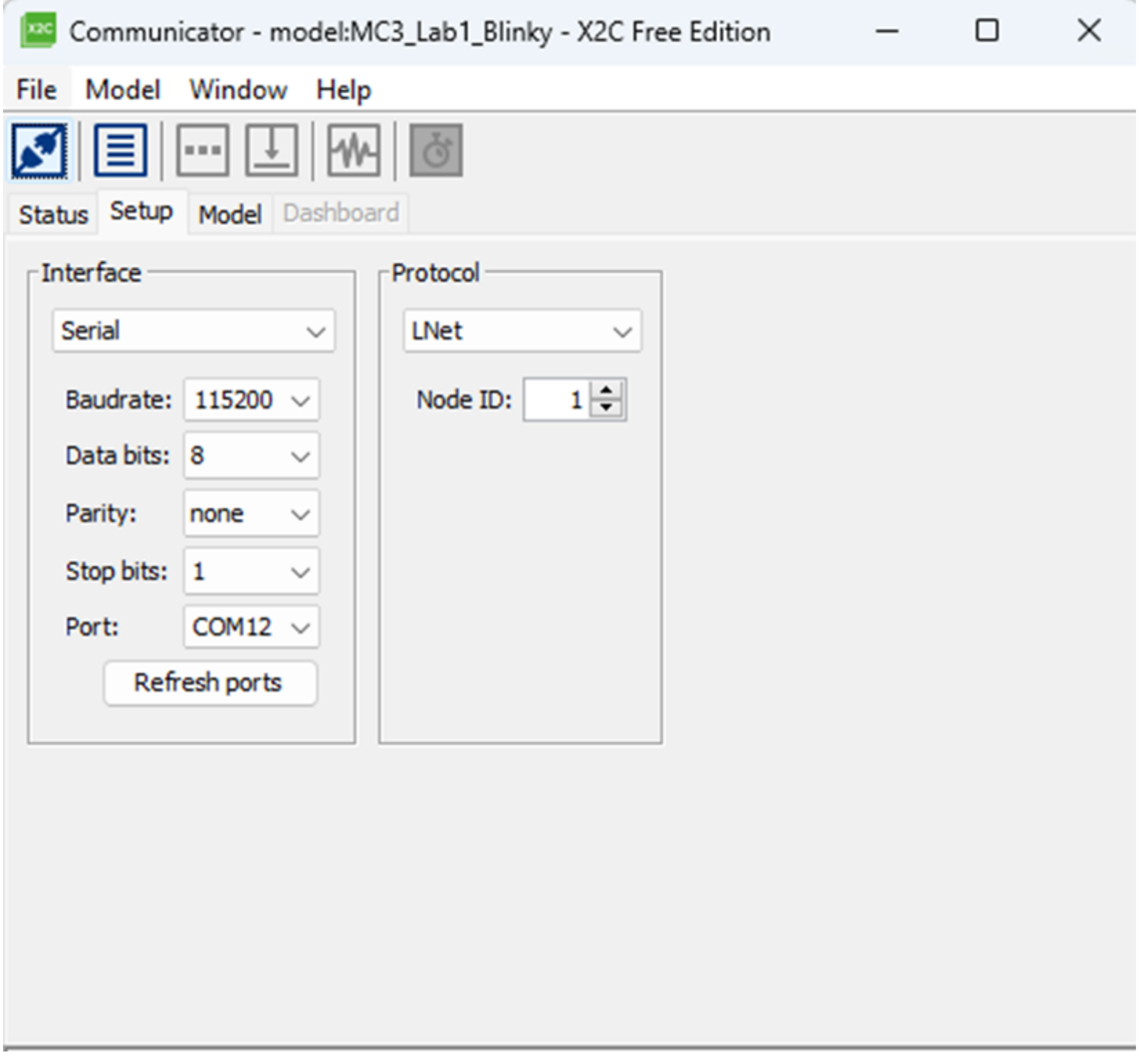

Search Results
Output ×
Git - pwm.c [mc_foc_sl_fip_dspic33ck_mclv
Kits ×
MPLAB® Code Configurator ×
MC3_Lab1_Blinky (Build, Load, ...)

```
UDID5 = 000109

Calculating memory ranges for operation...

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x97ff
configuration memory

Programming/Verify complete
```

2.4 Verify that the LED on the board is blinking. This depends on the state of the switch.

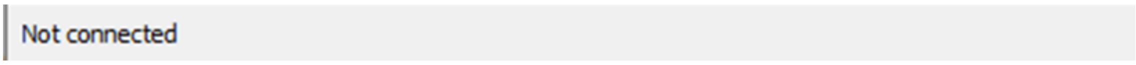
3 Use X2C Communicator to Verify the Operation

3.1 Verify the **Serial** connection settings on the **Setup** tab.



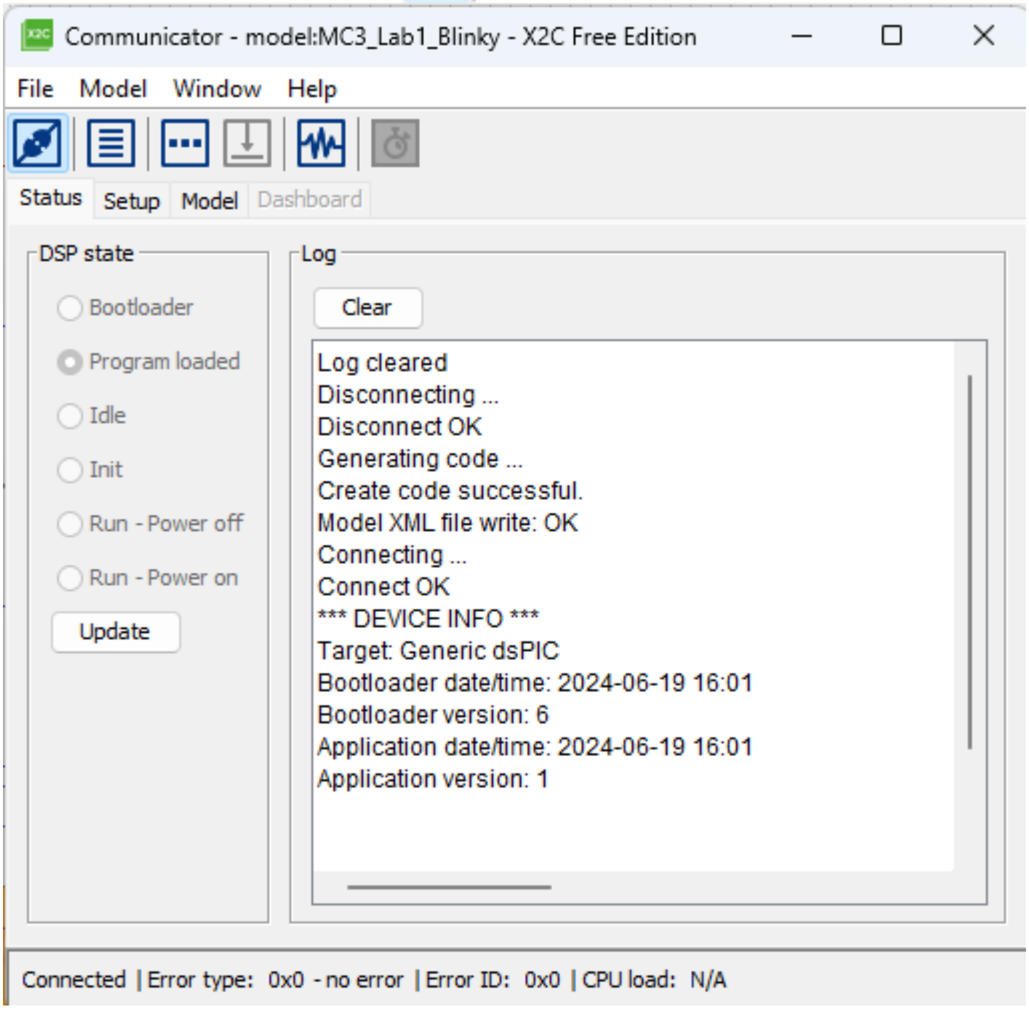
Interface
Serial
Baudrate: 115200
Data bits: 8
Parity: none
Stop bits: 1
Port: COM12

Protocol
LNet
Node ID: 1



Not connected

3.2 Connect to the HW by clicking the button.



DSP state

○ Bootloader
● Program loaded
○ Idle
○ Init
○ Run - Power off
○ Run - Power on

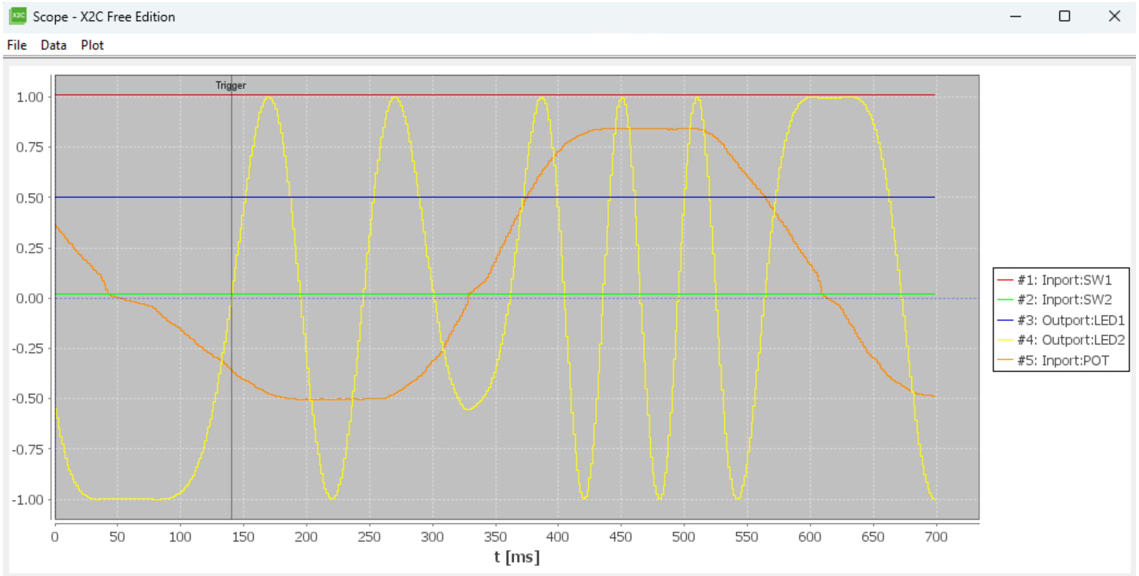
```
Log cleared
Disconnecting ...
Disconnect OK
Generating code ...
Create code successful.
Model XML file write: OK
Connecting ...
Connect OK
*** DEVICE INFO ***
Target: Generic dsPIC
Bootloader date/time: 2024-06-19 16:01
Bootloader version: 6
Application date/time: 2024-06-19 16:01
Application version: 1
```

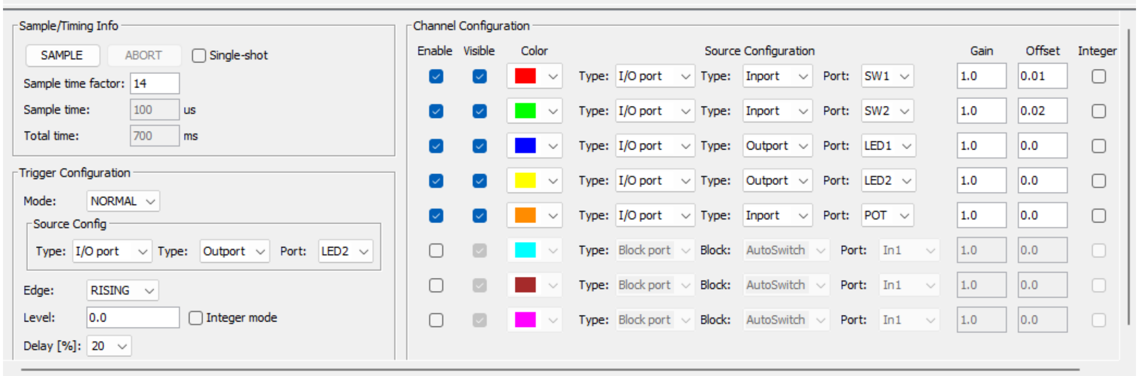
Connected | Error type: 0x0 - no error | Error ID: 0x0 | CPU load: N/A

3.3 Verify the time stamps of **Application date/time**.

3.4 Start **Scope** with the button.

3.5 Use **Scope** to check the signals and verify the HW behavior.





Sample/Timing Info

SAMPLE ABORT □ Single-shot

Sample time factor: 14

Sample time: 100 us

Total time: 700 ms

Trigger Configuration

Mode: NORMAL

Source Config

Type: I/O port Type: Outport Port: LED2

Edge: RISING

Level: 0.0 □ Integer mode

Delay [%]: 20

Channel Configuration

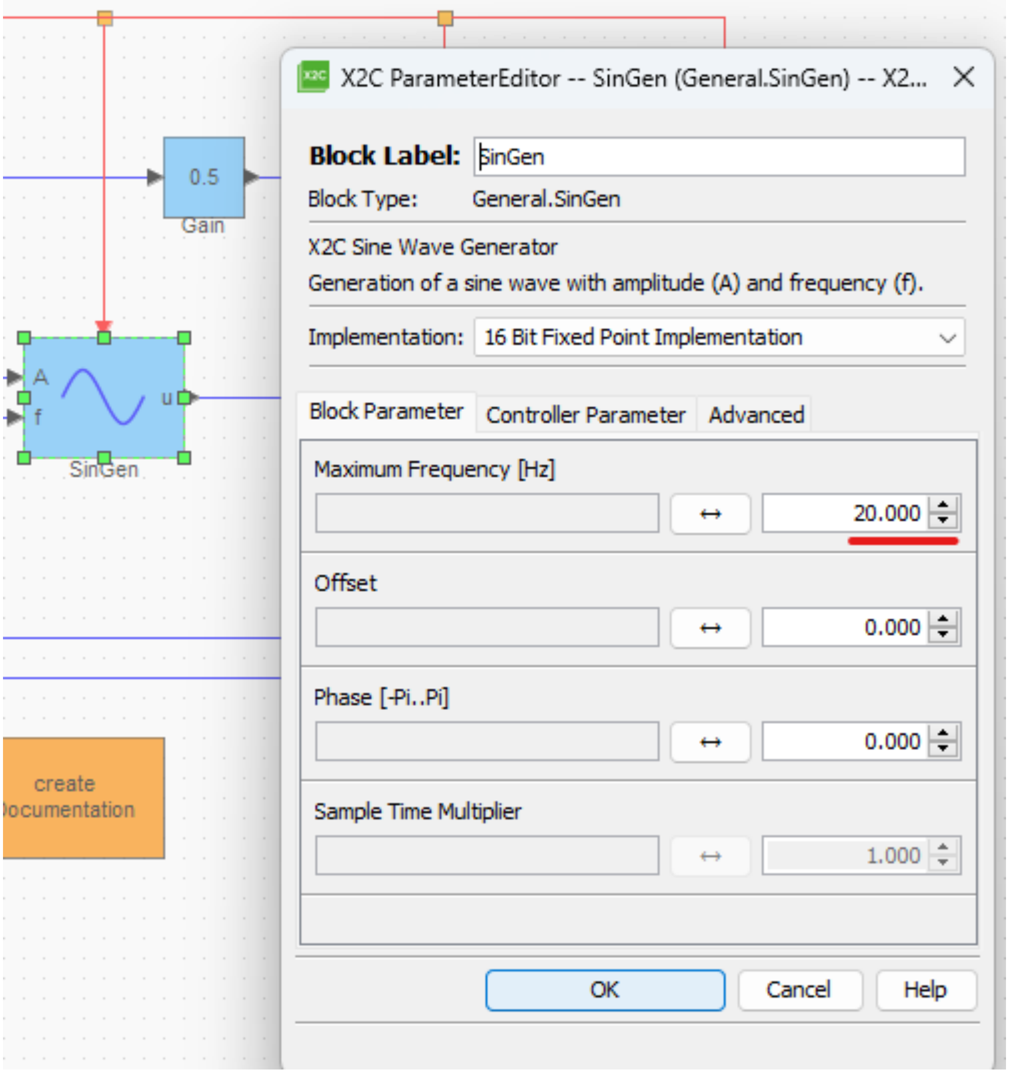
Enable	Visible	Color	Source Configuration			Gain	Offset	Integer
☑	☑	🟥	Type: I/O port	Type: Inport	Port: SW1	1.0	0.01	□
☑	☑	🟩	Type: I/O port	Type: Inport	Port: SW2	1.0	0.02	□
☑	☑	🟦	Type: I/O port	Type: Outport	Port: LED1	1.0	0.0	□
☑	☑	🟨	Type: I/O port	Type: Outport	Port: LED2	1.0	0.0	□
☑	☑	🟧	Type: I/O port	Type: Inport	Port: POT	1.0	0.0	□
□	☑	🟦	Type: Block port	Block: AutoSwitch	Port: In1	1.0	0.0	□
□	☑	🟥	Type: Block port	Block: AutoSwitch	Port: In1	1.0	0.0	□
□	☑	🟪	Type: Block port	Block: AutoSwitch	Port: In1	1.0	0.0	□

3.6 Optional: Change the scope channels and trigger points.

3.7 Navigate back to the model window.

> ❗ Do not click the Open button in MCC again! It would open new Scilab instance that would break the process. The model should be still open in the background. Use the Windows® task bar or **ALT+TAB** to go back to the model window.

3.8 Change the model parameters.



X2C ParameterEditor -- SinGen (General.SinGen) -- X2...

Block Label: SinGen

Block Type: General.SinGen

X2C Sine Wave Generator

Generation of a sine wave with amplitude (A) and frequency (f).

Implementation: 16 Bit Fixed Point Implementation

Block Parameter | Controller Parameter | Advanced

Maximum Frequency [Hz] ↔ 20.000

Offset ↔ 0.000

Phase [-Pi..Pi] ↔ 0.000

Sample Time Multiplier ↔ 1.000

OK Cancel Help

3.9 Note the block parameters are changed in run time on the HW as well.

Remember, the model is connected to the X2C Communicator via RMI. The X2C Communicator is connected to the HW via UART (Lustre Network (LNET) protocol). Therefore, if a parameter is changed in the model, it is transferred to the HW immediately at run time.

X2C Communicator is using LNET protocol to communicate to the HW. LNET is implemented for multiple peripherals and communication layers like USB Communication Device Class (CDC) , Controller Area Network (CAN), and Transmission Control Protocol/Internet Protocol (TCP/IP).

Next

Lab 2 - Torque Mode (/xwiki/wiki/masters/view/masters-2025-lab-manuals/25085/lab2/)

♥ 0 Tags: _$wip_ (/xwiki/wiki/masters/view/Main/Tags?do=viewTag&tag=_%24wip_)

Created by Mark Reynolds (/xwiki/bin/view/XWiki/MarkReynolds) on 2024/12/23 06:58

Information contained on this site regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

No comments for this page

Comment