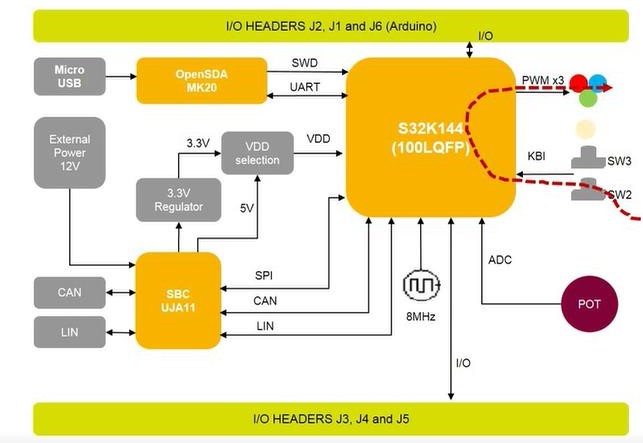
MBDT and FreeMASTER

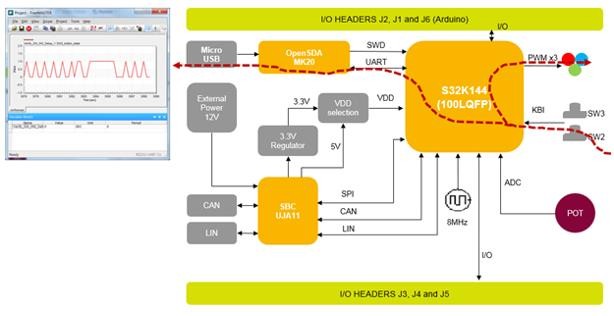
# MBDT Toolbox version 4 was used in lab 1.

Same can be used to run this lab.

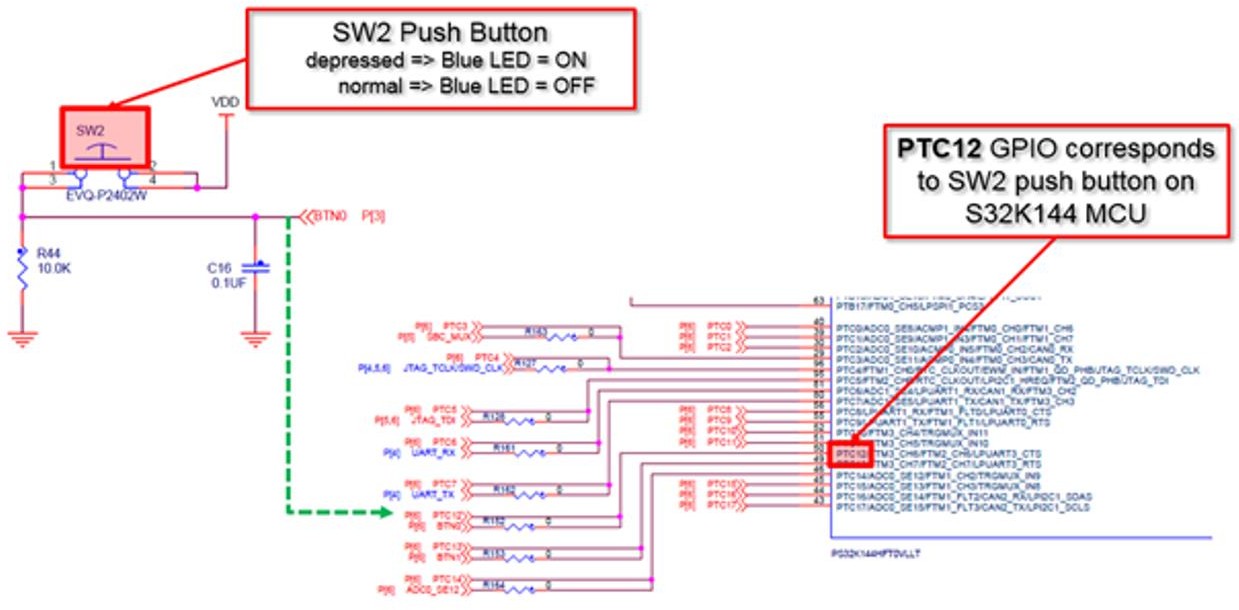
* 1. GPIO Application
     + The simple application consists of:
       - Read the SW2 push button state at each 0.1 second.
       - Switch on/off the Blue LED.
       - Record the value of the push button and display its state via serial communication in FreeMASTER.
     + The application flow is shown next. The code that runs on S32K144 MCU will be pooling at each 0.1 second the state of the SW2 push button and depending on the value read via the associated digital input will update the state of the Blue LED by writing 1 or 0 to the associated digital output.

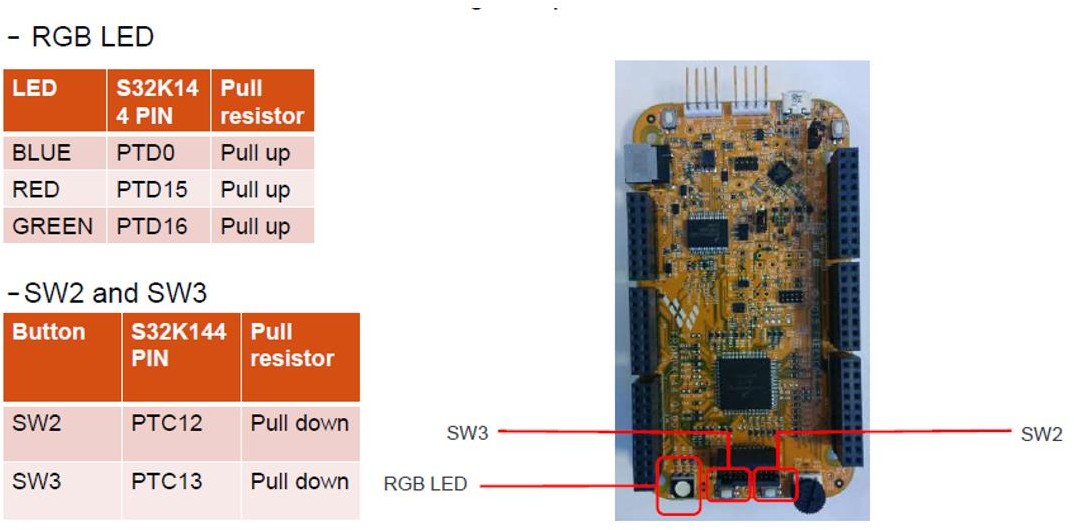


* In the same time the application needs to configure the UART peripheral to send the value of the variable associated with the SW2 push button state to the FreeMASTER where it can be displayed in real time.



* On the S32K144EVB the hardware schematics indicates that the signal from the SW2 push button is routed to the general-purpose pin PTC12. That is the input we are going to use for reading the push button state at each 0.1 second.

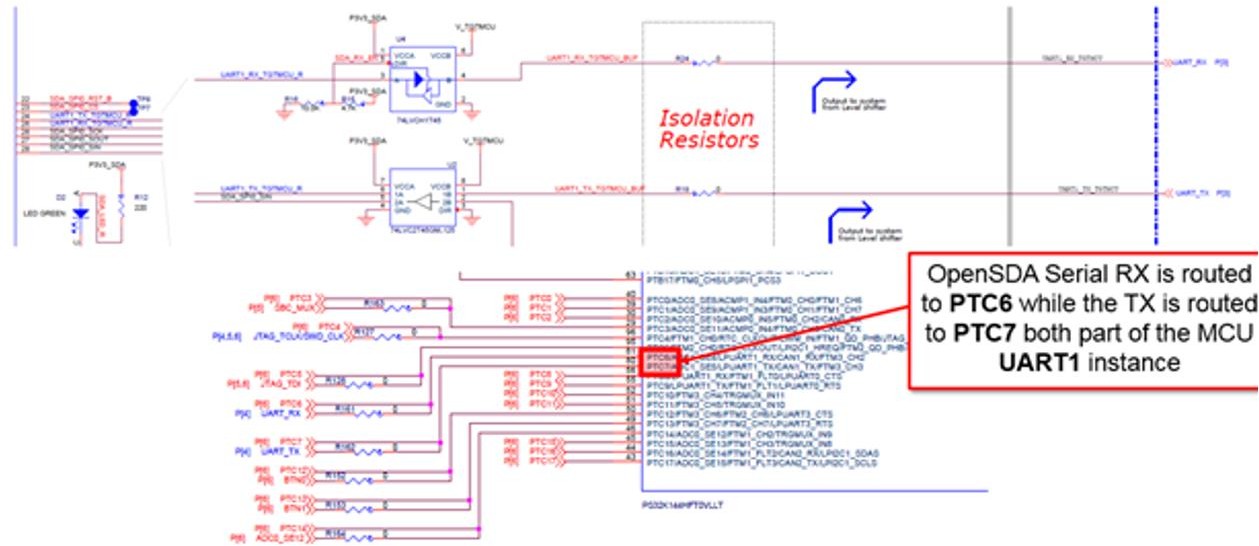




* The Blue LED is associated with the PTD0 general-purpose pin. Each time the SW2 push button will be depressed we need to switch the Blue LED to ON.

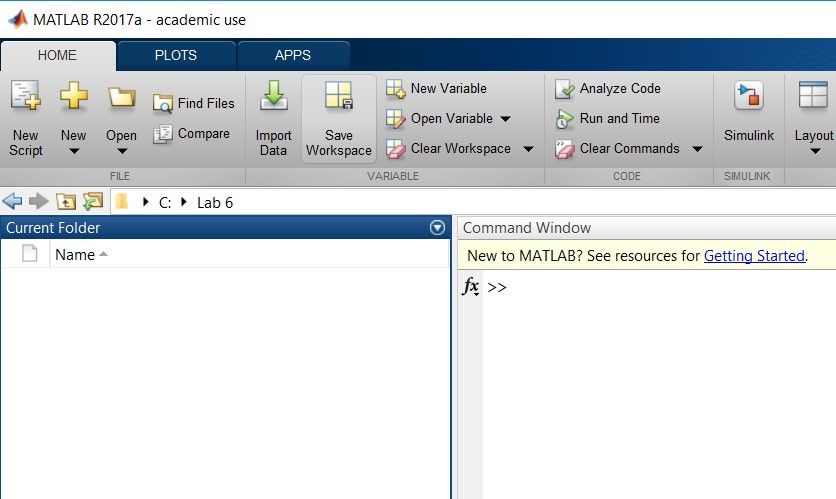


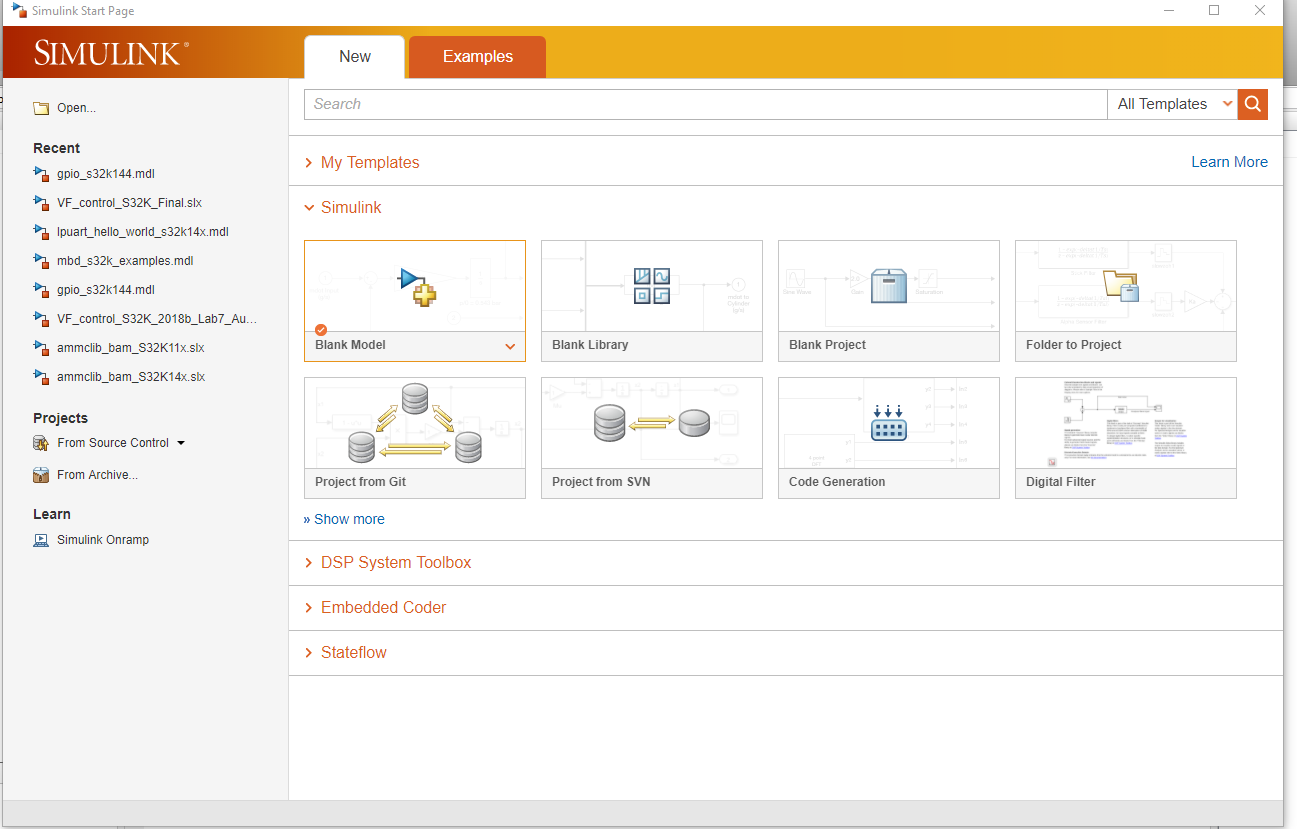
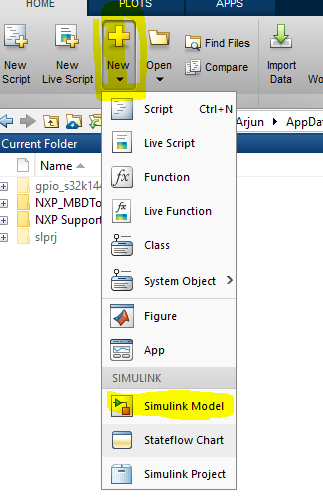
* To communicate with the S32K144 MCU, we are going to use the built-in OpenSDA that will have a dual purpose: to load/flash the application into the MCU flash memory and to assist with serial communication over the virtual COM port mapped on top of physical USB.
* For S32K144EVB the MK20 microprocessor that implements the OpenSDA protocol is connected with the S32K144 MCU via RX/TX assigned for UART1 peripheral instance as it is shown next.



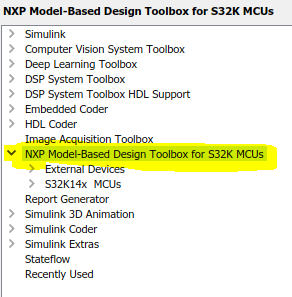
* The entire example can be implemented using standard Simulink blocks from NXP’s Model Based Design Toolbox for S32K1xx Simulink Library.
* Start MATLAB to open a new Simulink model.

G

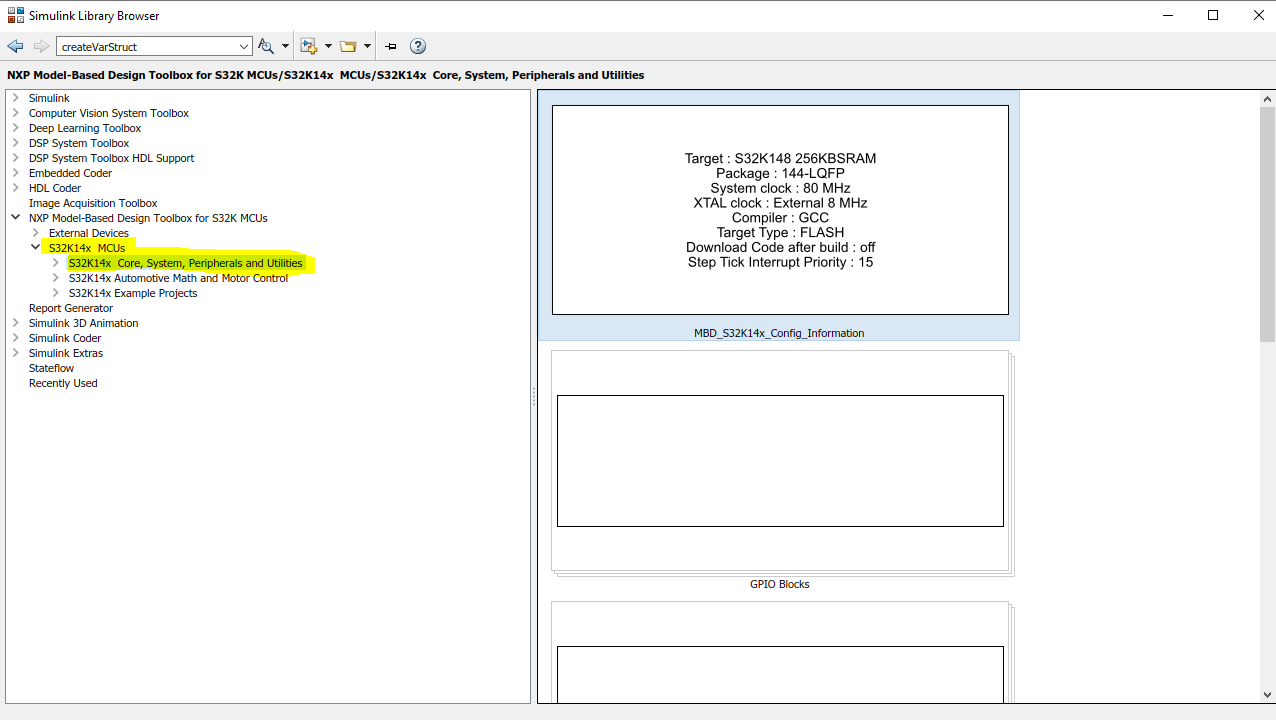


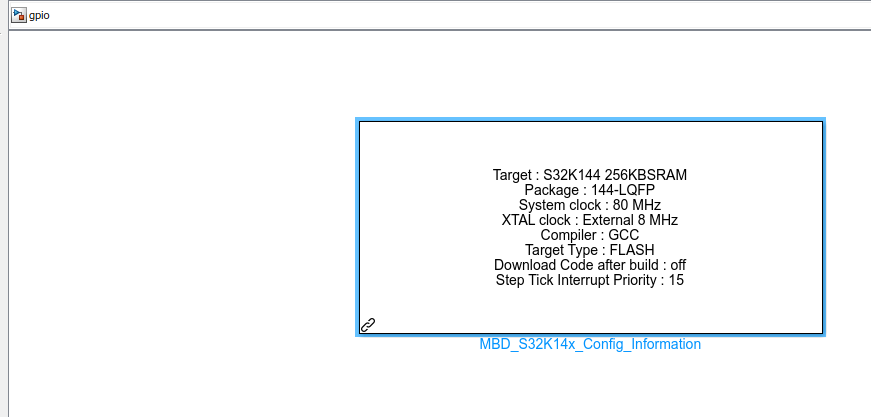
* Click on the Blank Model.
* Alternatively, Choose New -> Simulink Model.
* This generates a blank Simulink model. Do File ->Save as to name it and save to a desired location.
* Click on the Simulink Library Browser

and select the “NXP Model Based Design Toolbox for S32K MCUs”.

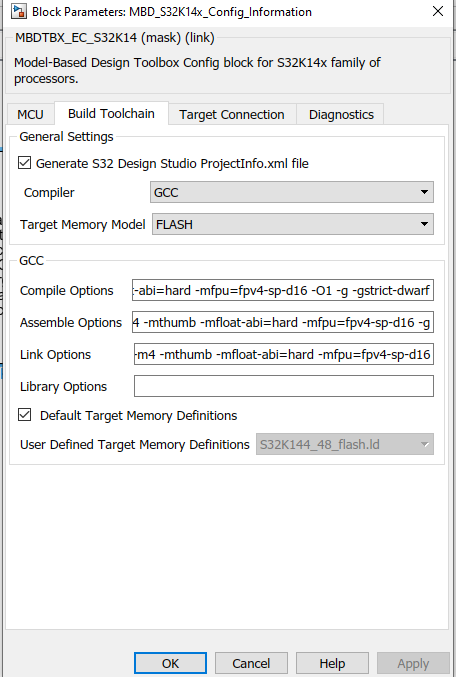


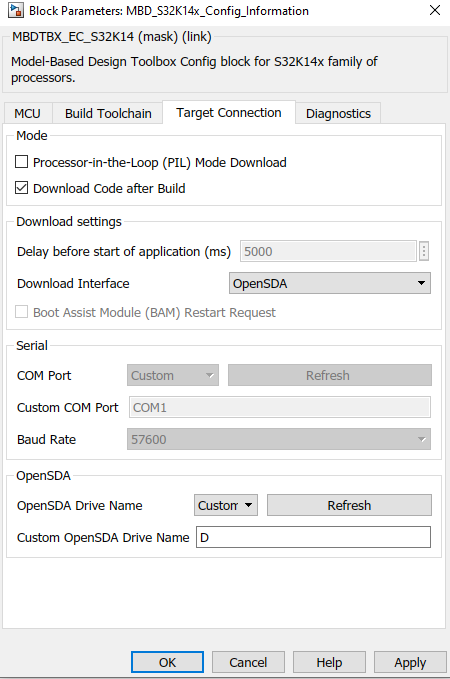
1. **Adding Config Block to the Model**

* Navigate to **S32K14x MCUs** 🡪 **S32K14x Core System Peripheral and Utilities** and **Right Click** on **MBD\_S32K14x Config\_Information block** as in below figure and select ***‘Add block to model’***



* Double click on “MBD\_S32K14x\_Config\_Information” block.
* Set “MCU”, “Build Toolchain”, “Target Connection” parameter as follows

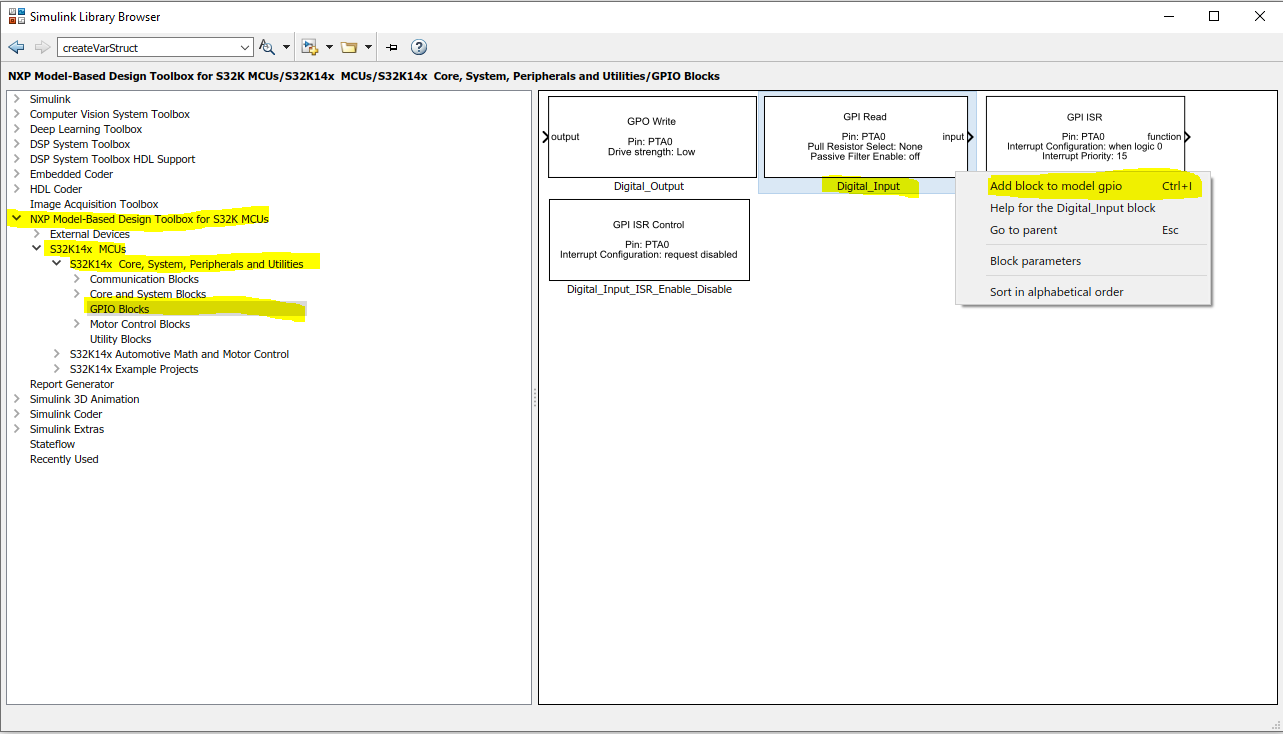




# Y

1. **Adding GPIO Read/Write Block to the model.**

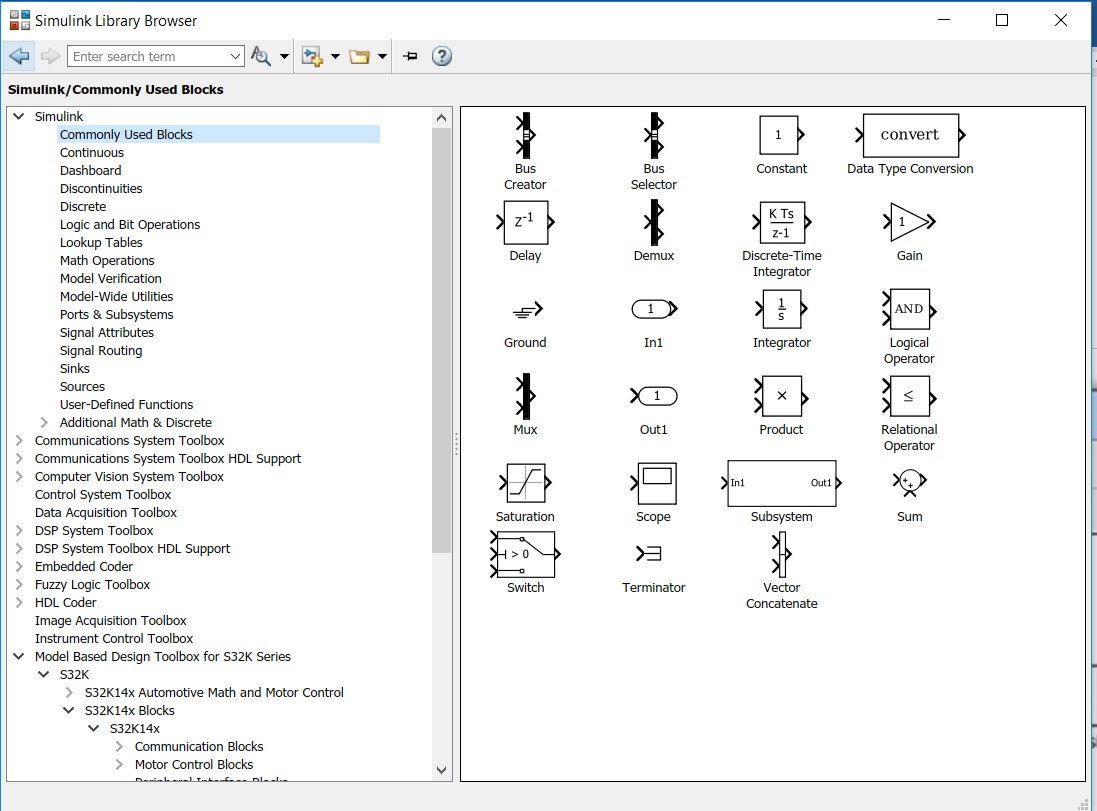
* Navigate to NXP Model Based Design Toolbox for S32K MCUs 🡪 S32K14x MCUs 🡪 S32K14x Core, System, Peripheral, and Utilities 🡪 GPIO Blocks 🡪 Digital Input (Right click 🡪 Add block to model) as in below figure.



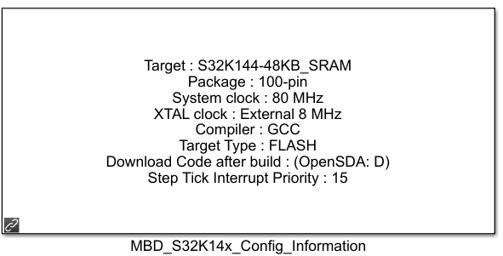
* Similarly add **Digital\_Output** Block to the model.

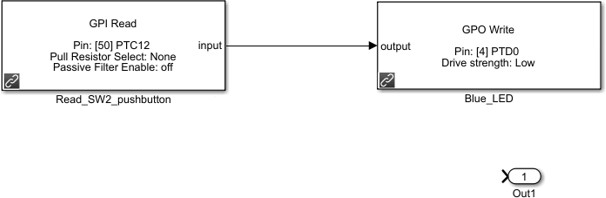
1. **Configure the Digital Input/ Output blocks.**

* Double click on **Digital Input** and Select **Input Pin** PTC12 and click Apply, then OK.
* Double click on **Digital Output** and Select **Output Pin** PTD0 and click Apply, then OK.
* You can rename the blocks by clicking on the block and selecting the name. Rename the **Digital Input** to **SW2 Push** Button and **Digital Output** to **BLUE Led**.
* Connect these blocks.
* Select Simulink “Commonly Used Blocks” from “Simulink Library Browser”.



* Right click on “Out1” and select “Add block to model GPIOLAB”.

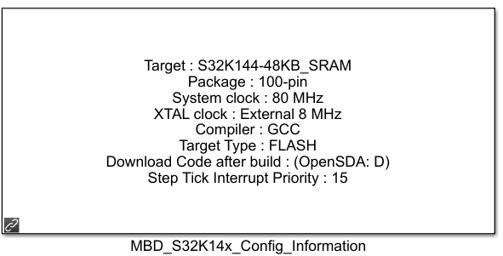


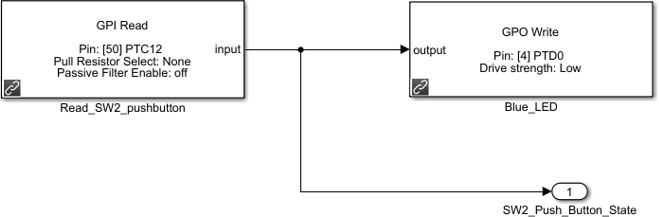


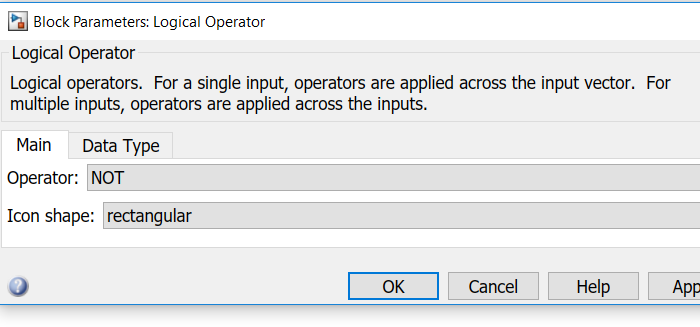
* Rename the “Out1” block as “SW2\_Push\_Button\_State”.

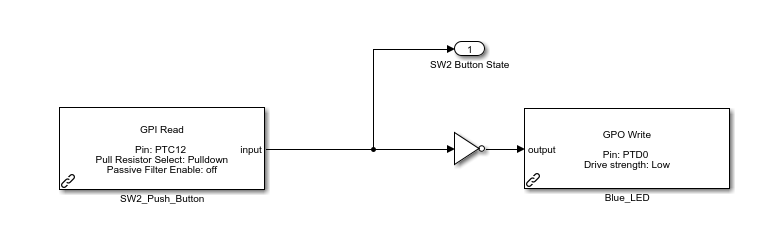


* Connect the Input of the “GPI Read” to the “SW2\_Push\_Button\_State”.

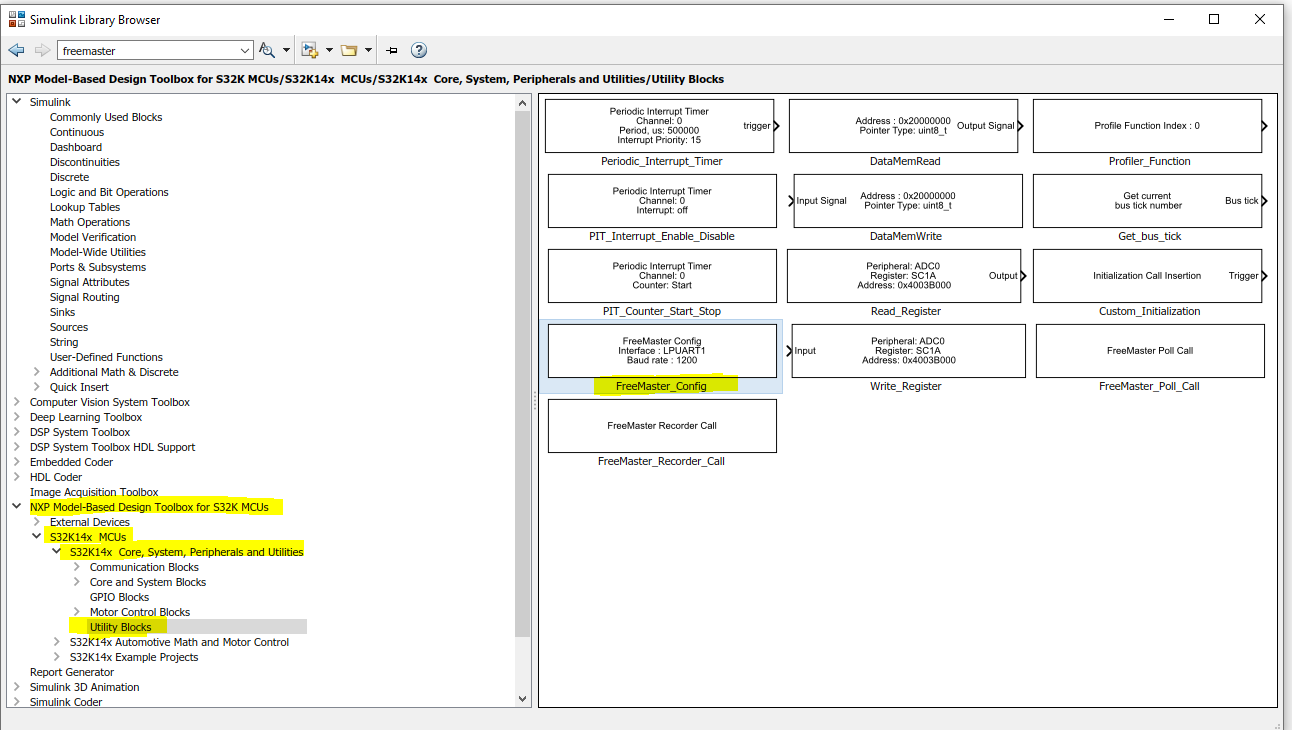


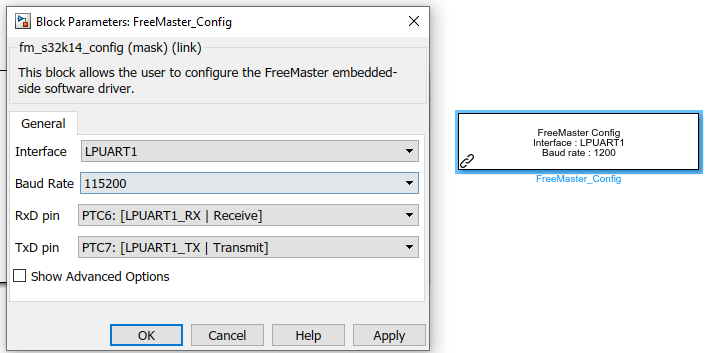


* Click on “Logical Operator” block from “Simulink Commonly Used Blocks” to change the “AND” operator to a “NOT”. If you prefer, you can change Icon Shape to “distinctive.”
* Move the “Logical Operator” block between the digital input and digital output. Finally, it looks as follows:



1. **FreeMaster Configuration**

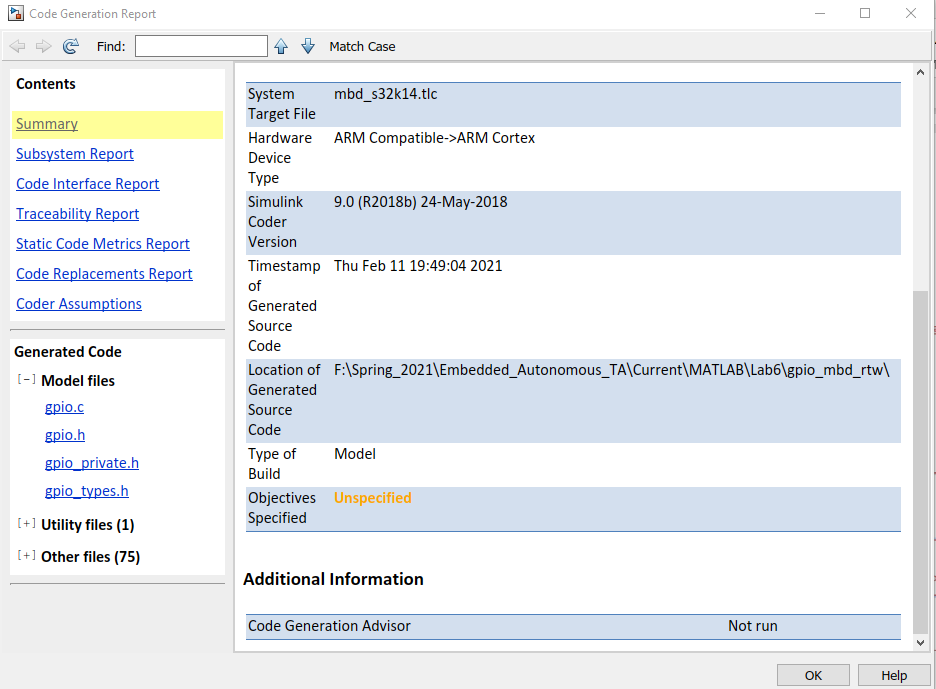
* Add FreeMaster\_Config box to the model and configure it as the image in the next page.
* Right click on “



* Click Apply, then OK.
* After this we can **connect the board** to the PC, Verify the “**Download Code After Build**” in the **configuration block** is **checked** and then **Build** the project now.



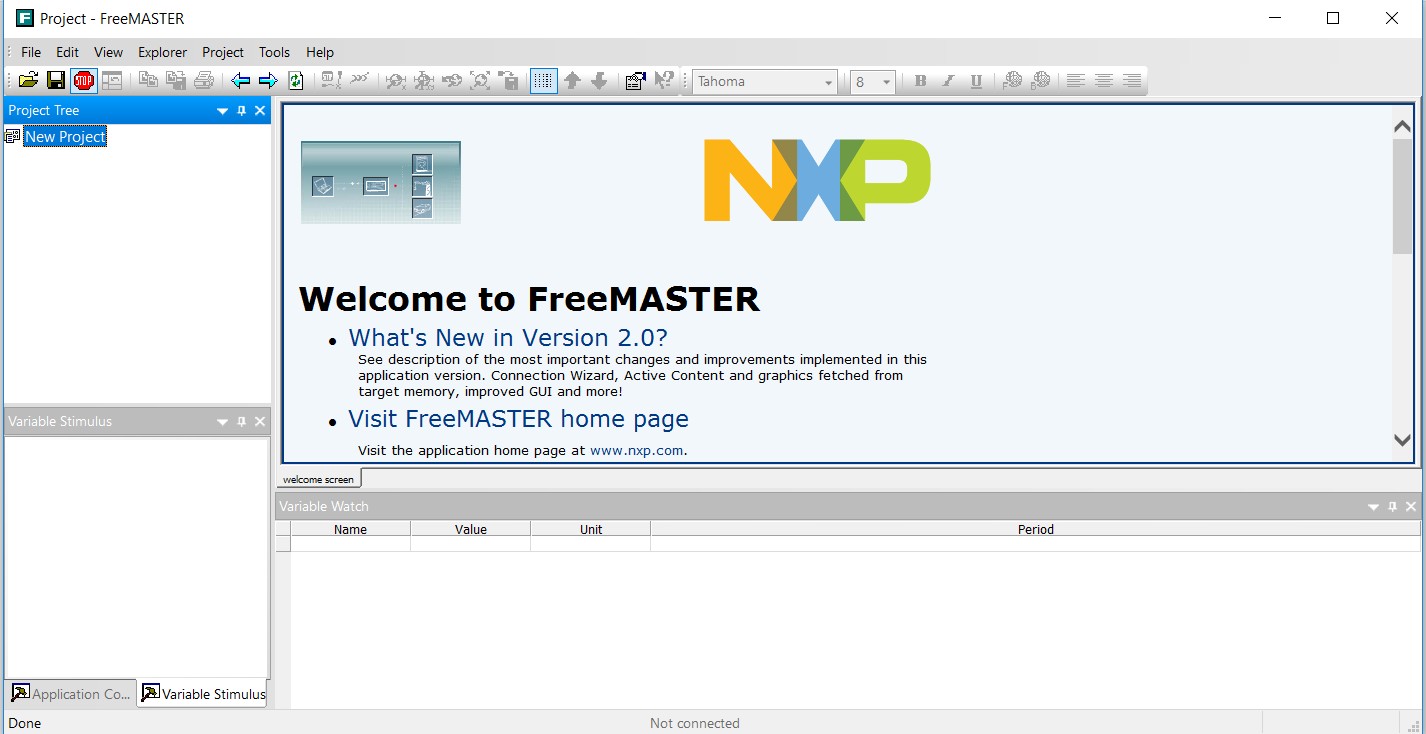
The Code Generation Report pops up and observe the location of Generated Source Code. This folder with contain all the codes generated for your application. Now you can also investigate the folder where there will be a directory named like **project name\_mbd\_rtw** this folder contains an **project\_name.mot** file. This mot file is the file which is the output of your build and this can be copied to the board drive (eg: D:) for downloading the application to the board. For now it happens automatic and you don’t have to do this step.



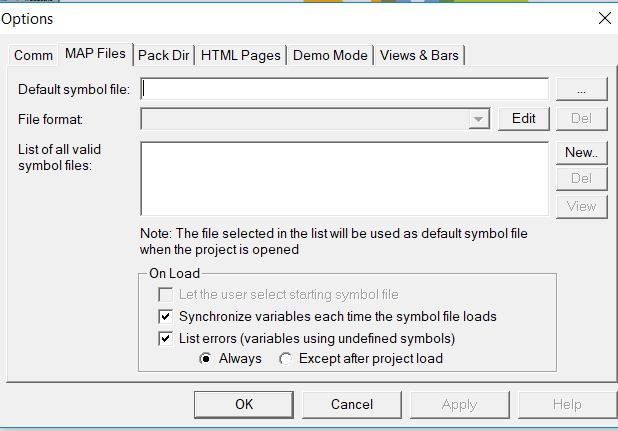
* The blue LED toggles when you push SW2.

**Viewing the Variables in the Application via FreeMASTER**

* Start FreeMASTER software.

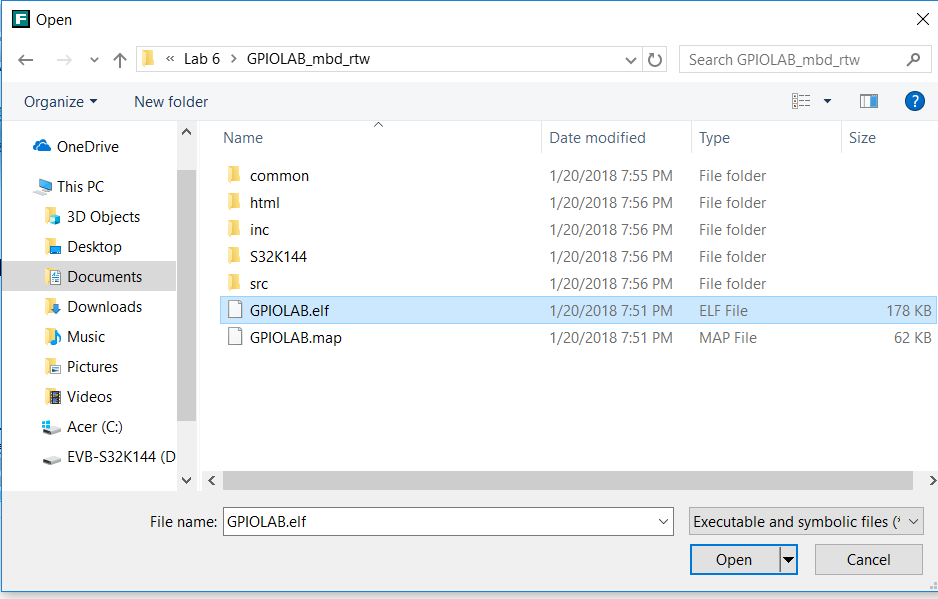


* Select **Project** 🡪 **Options** 🡪 **MAP** **Files**.

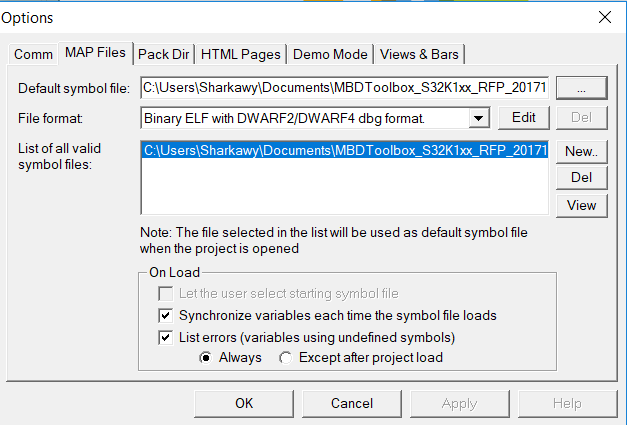




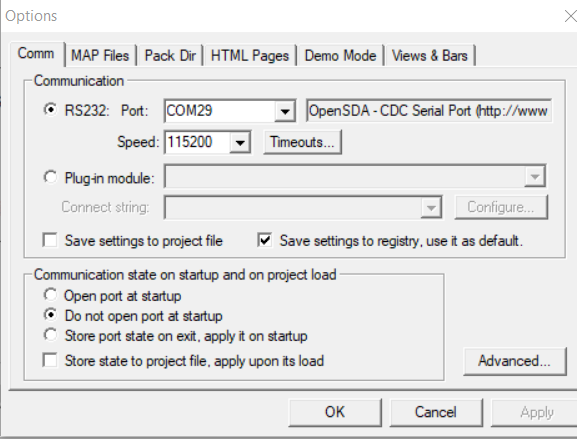
* Click on and locate the **“project\_name.elf”** that you have already generated in the **location of generated code**.



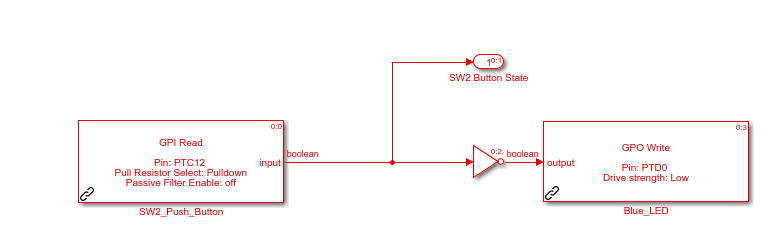
* Click open than OK.

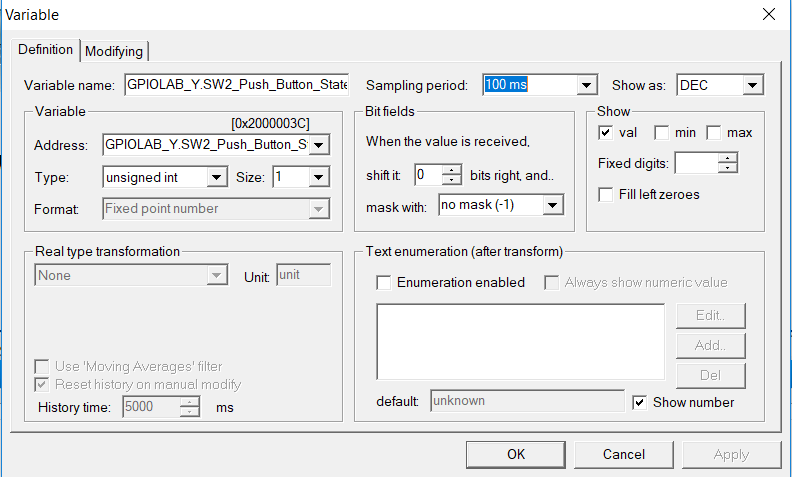


* Check your com port at the device manager.
* Select “Comm” and set your communication by selecting RS232 🡪 Port select the COM port of your MCU and speed as 115200.

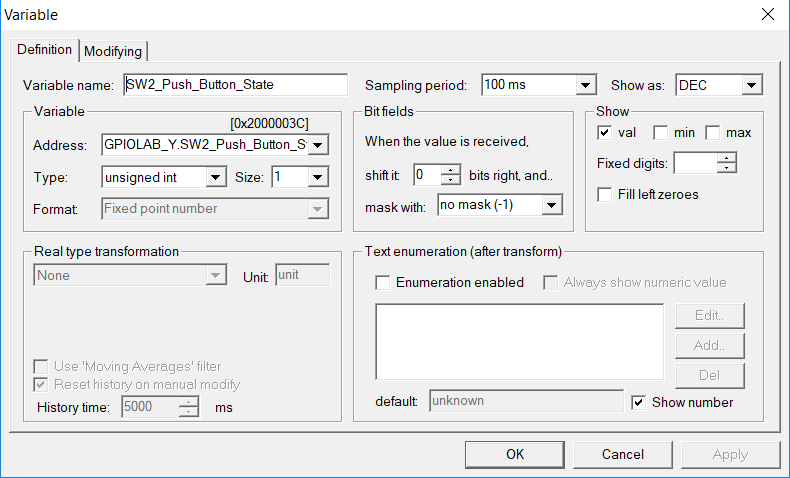


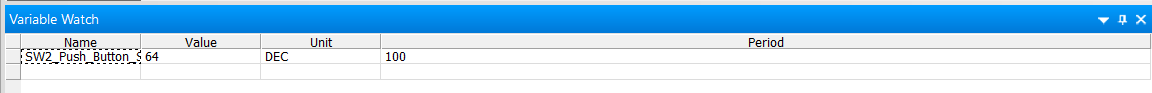
* Click OK and then select “**Tools**” then “**Connection Wizard**” to establish the communication.
* In connection wizard select: - Use Direct connection to on-board USB port, click next.
* Select COM port, Baud rate 115200, Click next. And then Finish.
* Create a new watch variable by double click on the Variable Watch window.
* In the address, select drop down menu, see the value **your\_project\_name\_Y.your out window name** in the matlab simulink. Eg: below the Output is named as SW2 Button State. So the value to be selected in address will be **gpio\_Y.SW2ButtonState.**



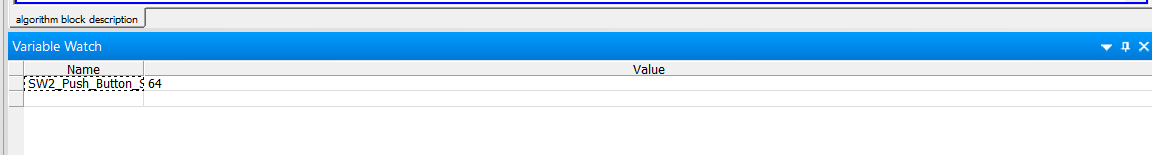
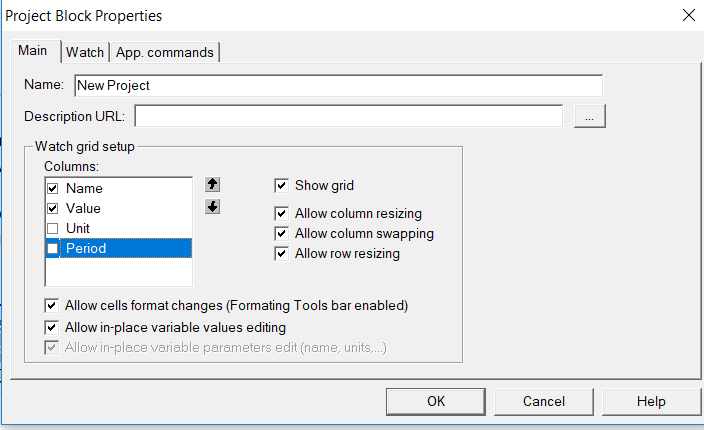


* Select Sampling rate 100ms and Change the variable name to SW2\_Push\_Button\_State.

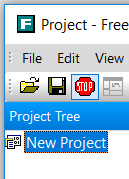




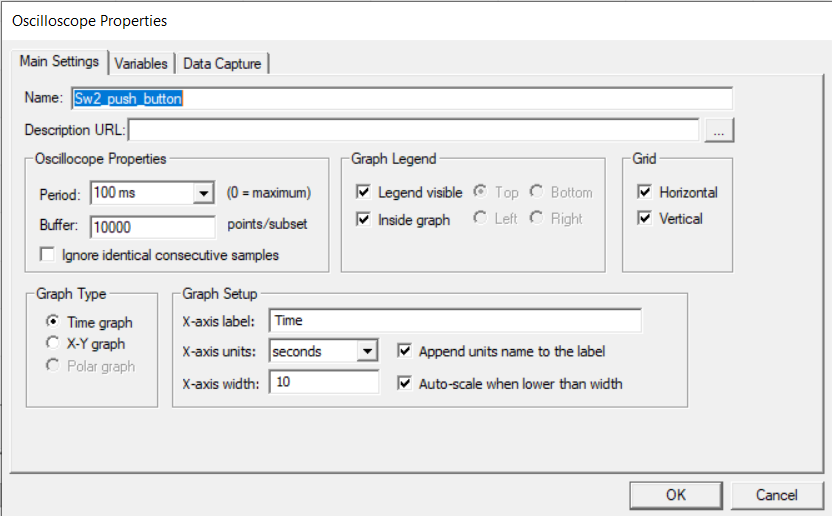
* Right Click on the “SW2\_Push\_Button\_State” variable and select “Watch Properties”. Open **Main** Tab



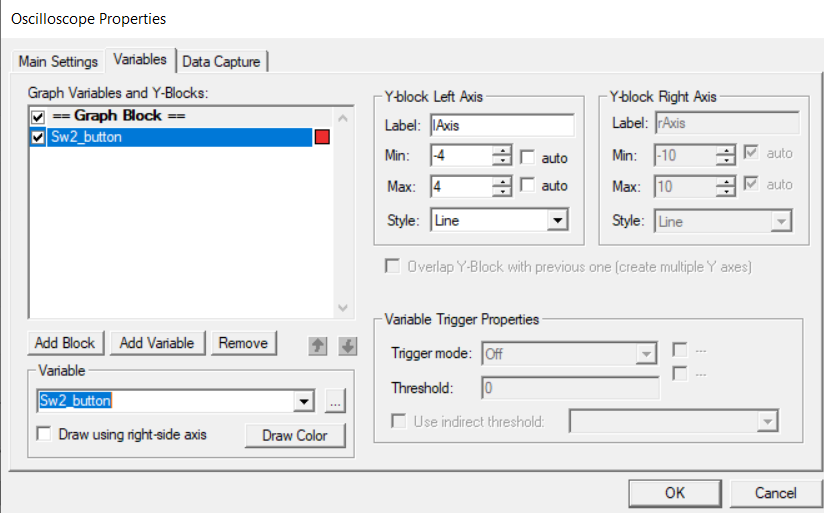
* Save FreeMASTER project as “GPIOLAB”.



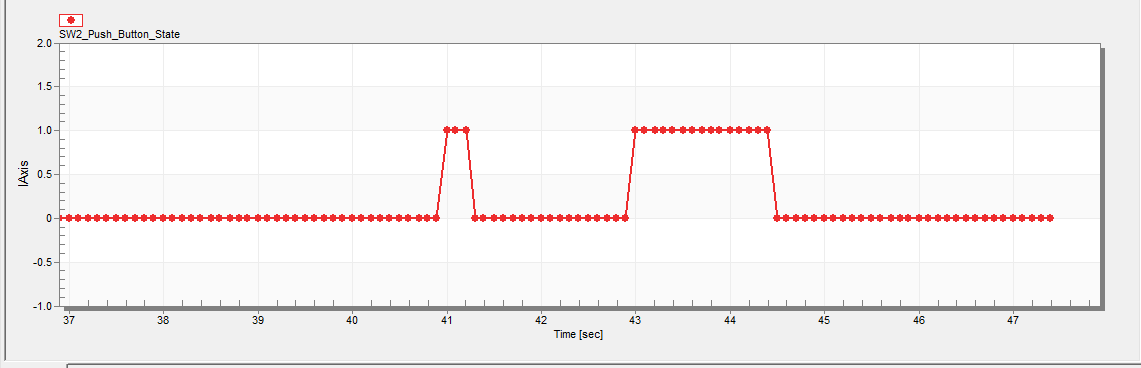
* Right Click on “New Project” and select “Create Oscilloscope”.



* Set the name as “SW2\_Push\_Button\_State” and the period as 100 ms.
* Select “Variables” Tab. Click on add variables and then select the variable for the oscilloscope. Do not disturb the Graph Block.



* Select the “SW2\_Push\_Button\_State” from the variable drop down.
* Click OK.
* Push SW2 several times and watch the oscilloscope at FreeMASTER.



**END OF LAB 6**

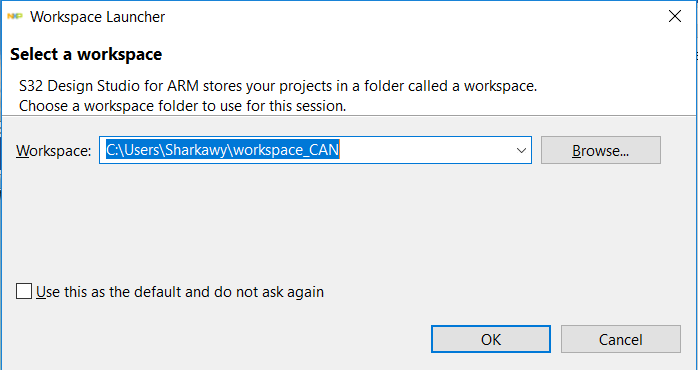
**Follows Optional Portions of the LAB.**

## FlexCAN (Optional part if two boards and two 12V adapters are available)

A FlexCAN module is initialized for 500 KHz (2 usec period) bit time based on an 8 MHz crystal. Message buffer 0 transmits 8-byte messages and message buffer 4 can receive 8-byte messages.

This example is intended for two EVBs to be connected, “Node A” and “Node B”. After Node A is initialized, it transmits an initial message. Node A then loops: wait to receive a message from Node B then transmit one back. After Node B is initialized it loops waits to receive a message from Node A then transmits one back.

* + - Start S32 Design Studio for ARM V1.3 and select a workspace (for example, workspace\_CAN).



* + - Select File, New and S32DS Project from Example.

lmpan X

Import example project

Project

bootloader\_KEA ^

FTM\_PWT

hello

hello clocks hello\_interrupts UART

S32K144

ADC

DMA

Fle N



FIexCAN\_F D FTM

hello hello\_clocks hello\_interrupts LPSPI

LPUART

S32K144 EAR SDK Example Projects

demO\_apps

ADC LOW\_POWER •

Name

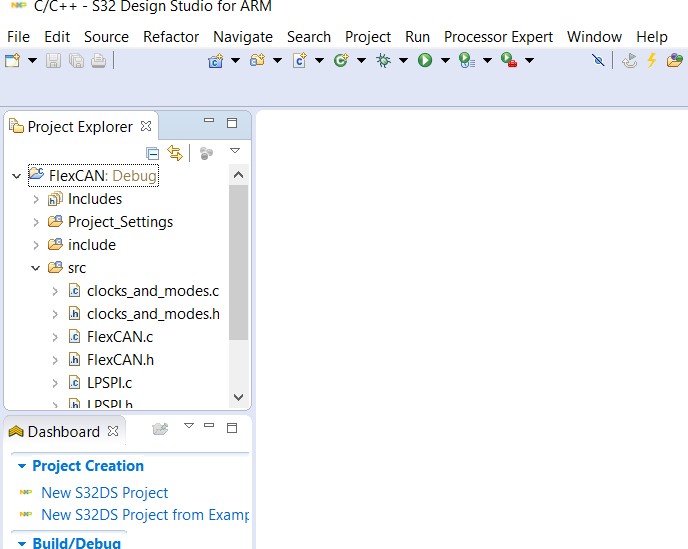
FlexCAN

Description

Simple CAN 2.0 transmit / receive

in|F|ish Cancel

* + - Select FlexCAN and click Finish.

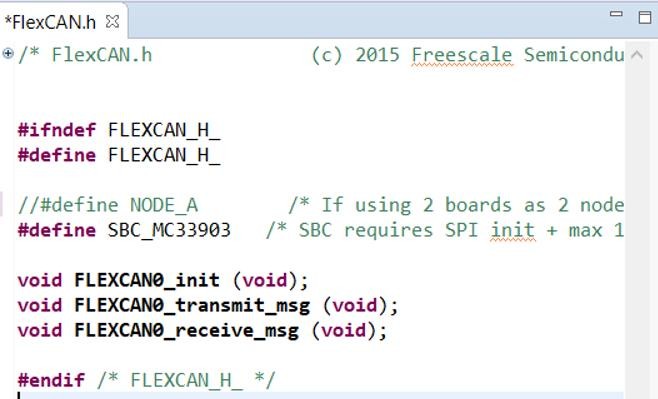


* + - Connect CAN High, CAN Low and ground between the two boards with a cable as shown
    - Move power supply selection jumper to use external 12 V (away from CAN connector per arrows below.

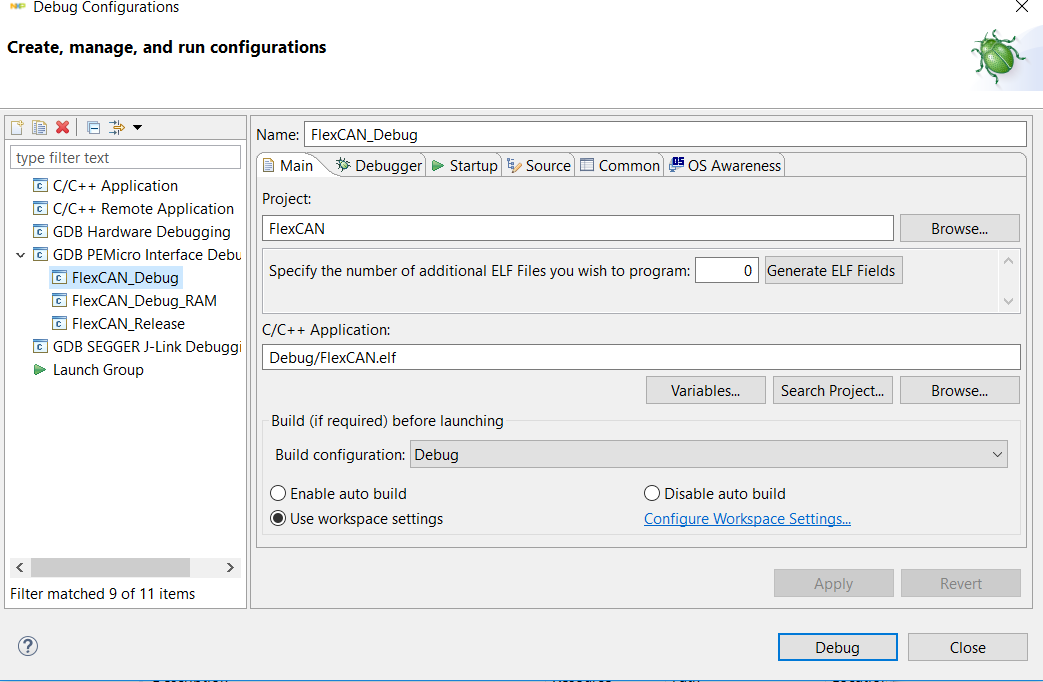


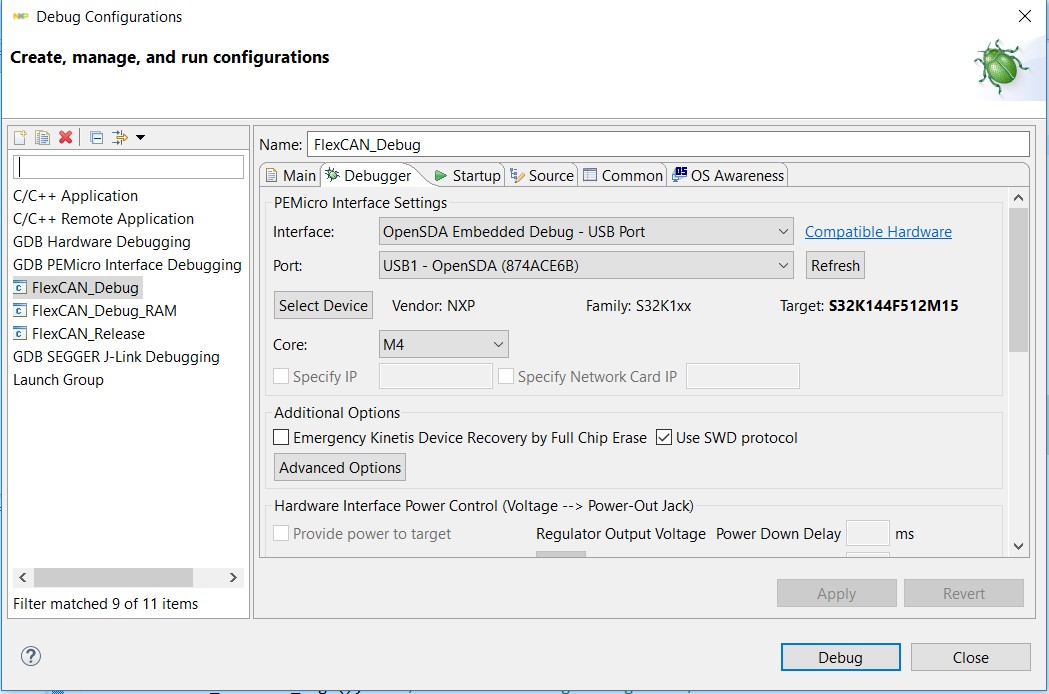
* + - Connect 12 V power supply to both boards
    - Configure code for Node B by commenting out the line in FlexCAN.h as shown below:

//#define Node A.



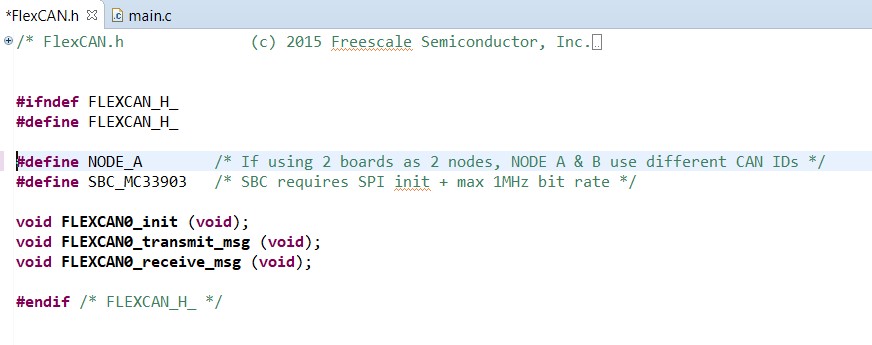
* + - Click on  to build the program.
    - Select Debug Configurations by clicking on the arrow at the left of  to flash program to the “Node B” EVB



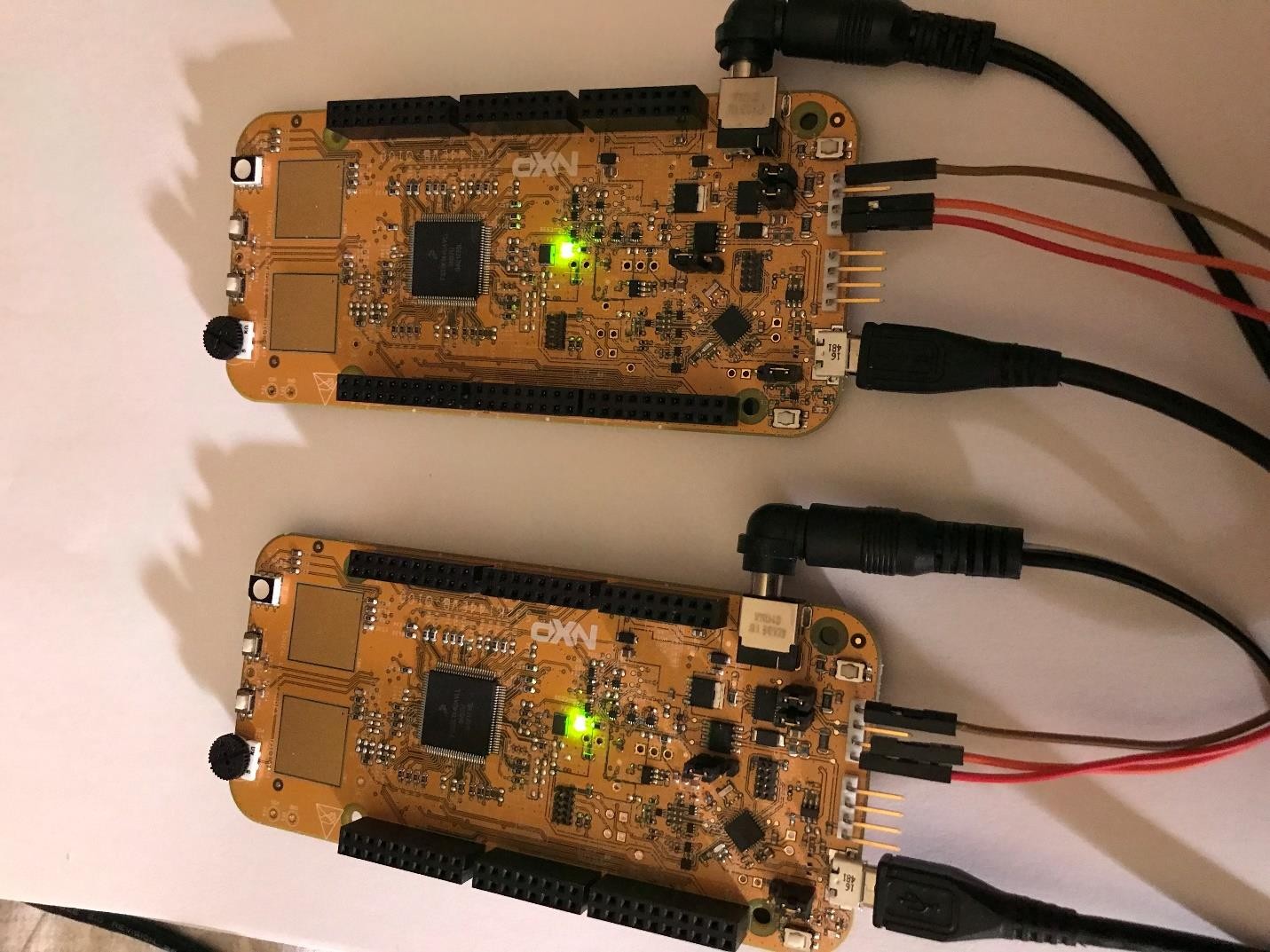


* + - Configure code for Node A by un-commenting out the line in FlexCAN.h as shown below:

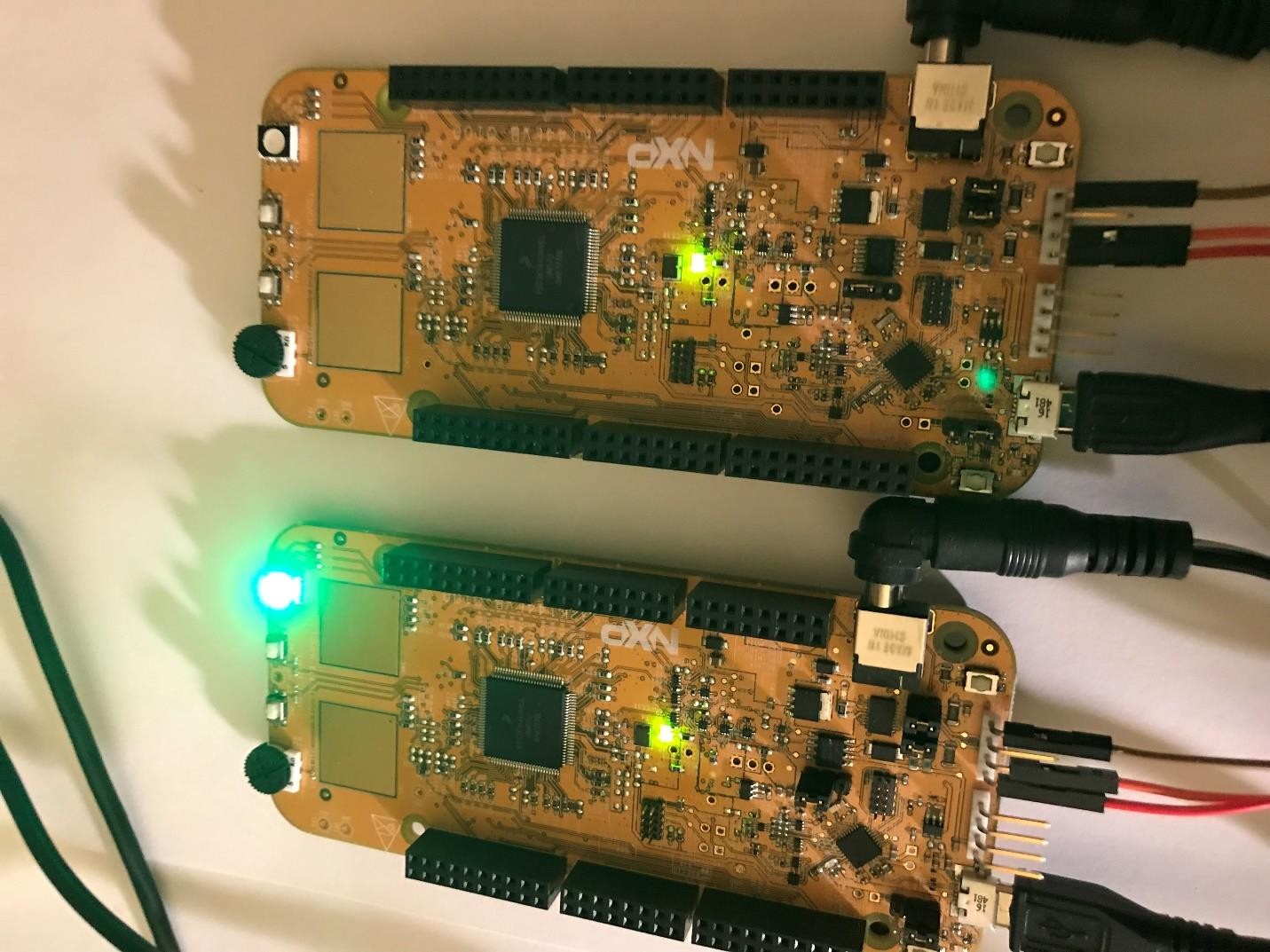
#define Node A



* + - Same as before, build the program and flash it to the “Node A” EVB.
    - With both boards powered, start the program on Node A which starts the transmission sequence.
    - The green LEDs on the boards will toggle flashing every 1000 CAN transmitted and received messages.

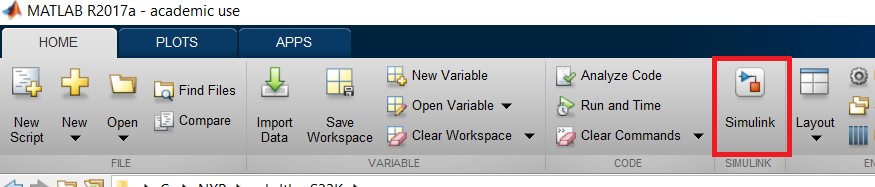




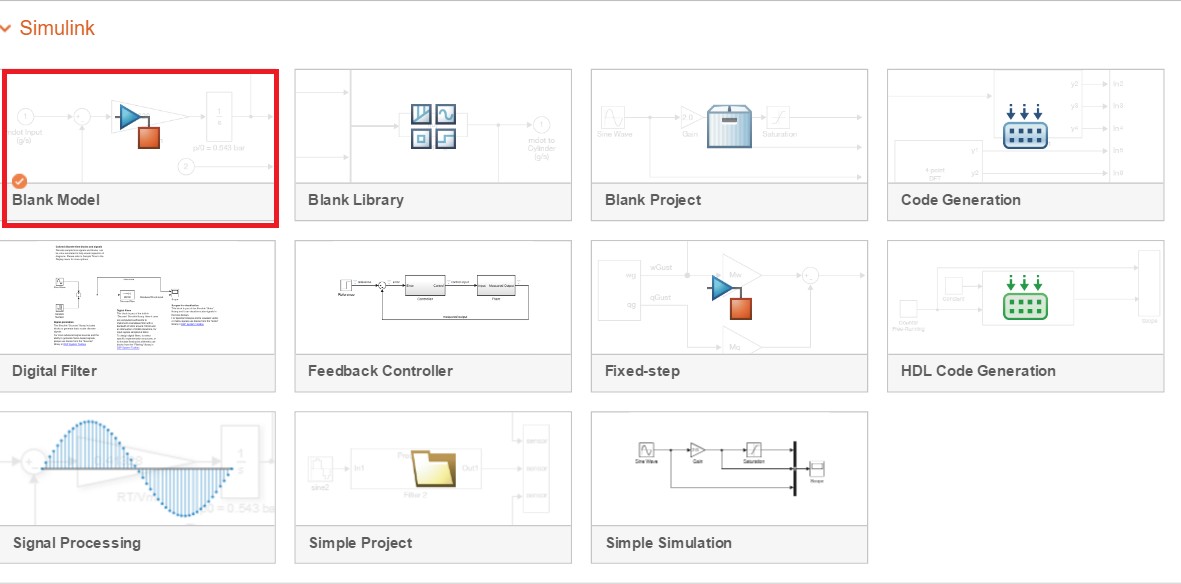


## Receiving a Data Pattern over the I2C (Optional part if two boards are available)

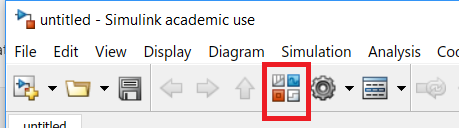
* Two S32K144 evaluation boards are needed for this part. The two boards are connected in a Master-Slave configuration using I2C protocol.
* From MATLAB (R2017a is used here) open Simulink by clicking on the Simulink icon on the top left.



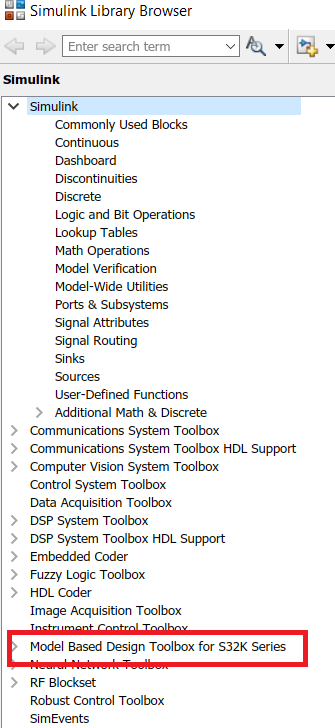
* The Simulink start up page opens. Select the Blank Model option.

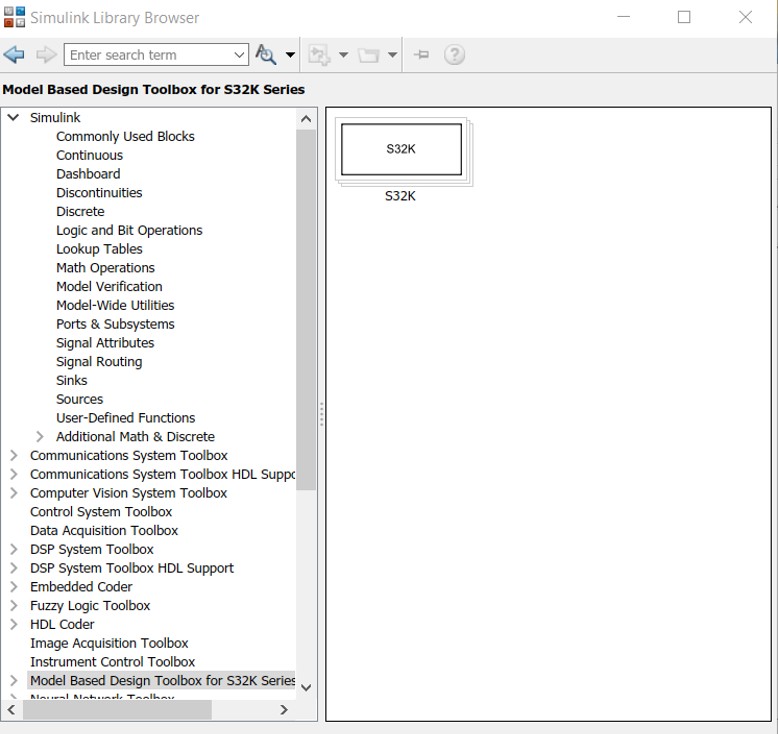


* In the opened blank Model select the Library Browser option.

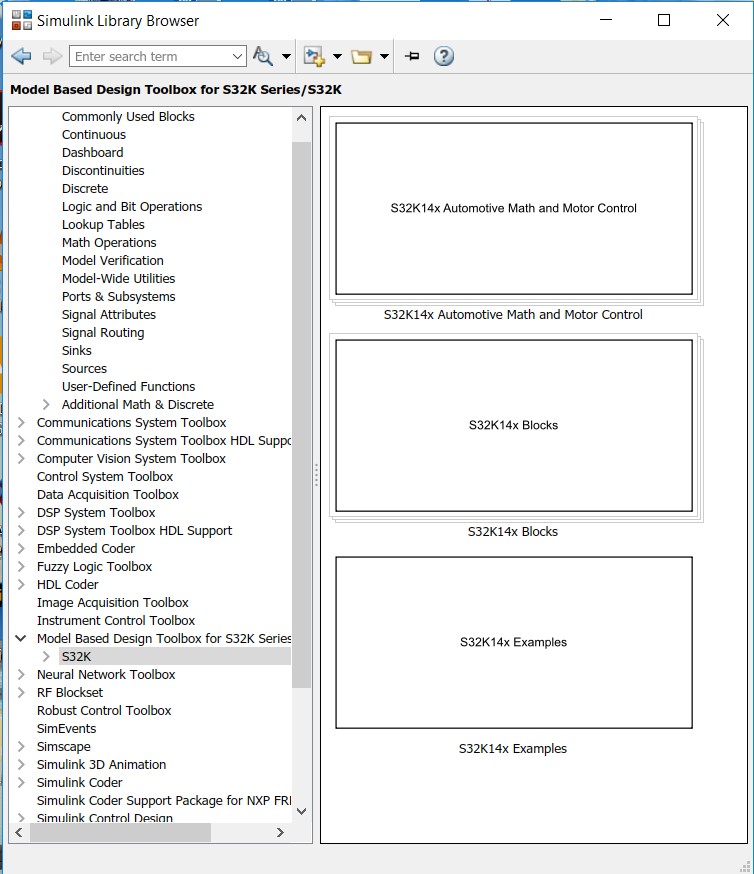


* From the opened list of libraries, select the Model Based Toolbox for S32K Series.





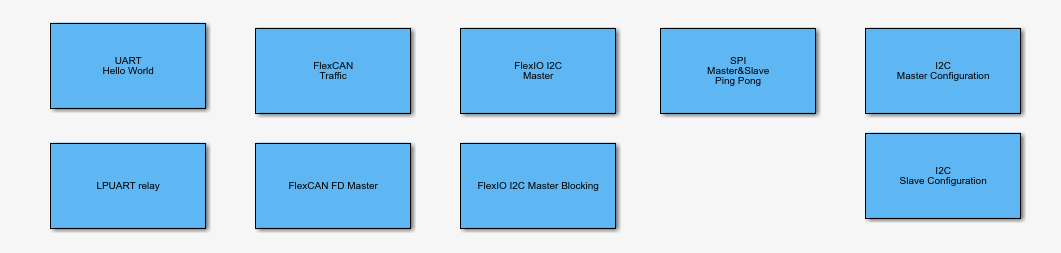
* + Expand the Model Based Design Toolbox for S32K series and double click on the Examples option.



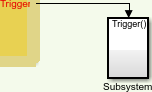
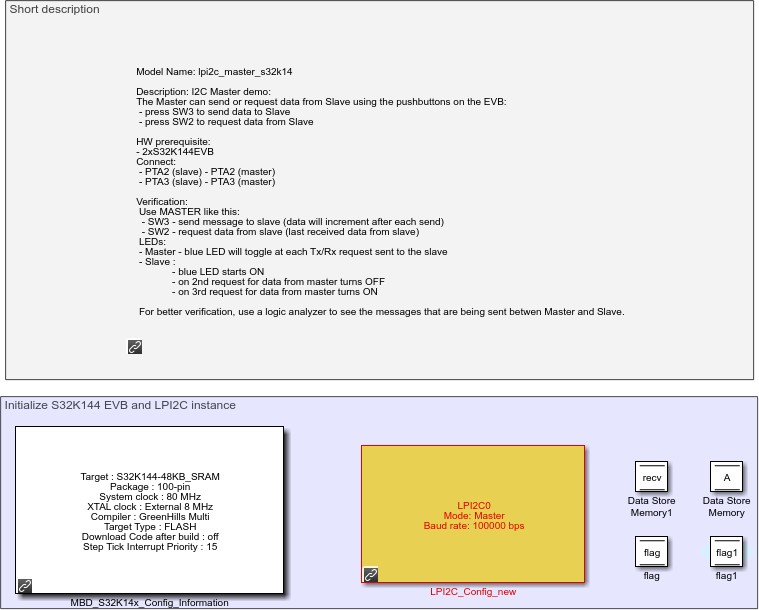
* + From the opened examples window, double click on the communications option.



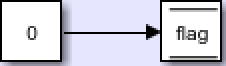
* + From the opened Communications window, select the I2C Master Configuration model and build it by selecting the suitable configuration settings.



* The Simulink lpi2c\_master\_s32K14 model will open. Follow the next steps to run the example.
* Be sure that the S32K144 board is connected to the PC using the USB cable.
* Double click on MBD\_S32K14x\_Config\_Information block and setup the Download Config parameters according with your PC and HW setup.



LPl2E0



Data initialization

SW2

SW I

0

0

GP I Read

GP I Read



On SW-i press SEND data to Slave

On SW2 press REQUEST DATA from Slave

Digitei\_lnput\_iSR

Trigger()

Digit aI\_Inpuit\_TS R 1

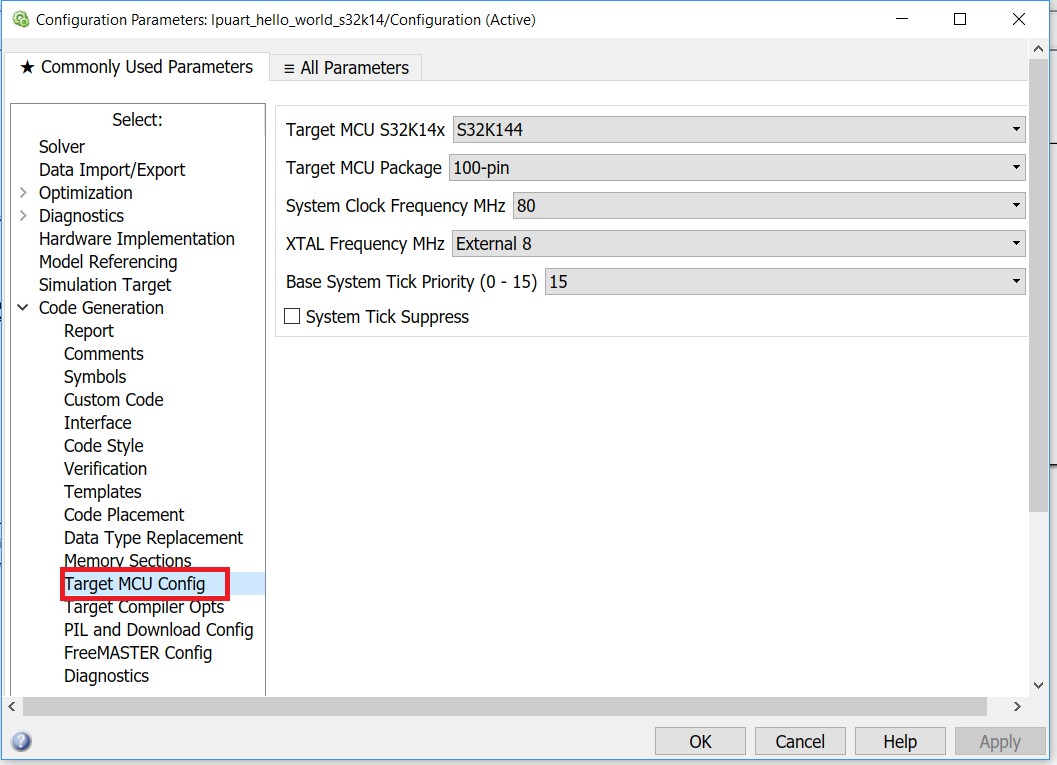
Trigger(

Subsyem2

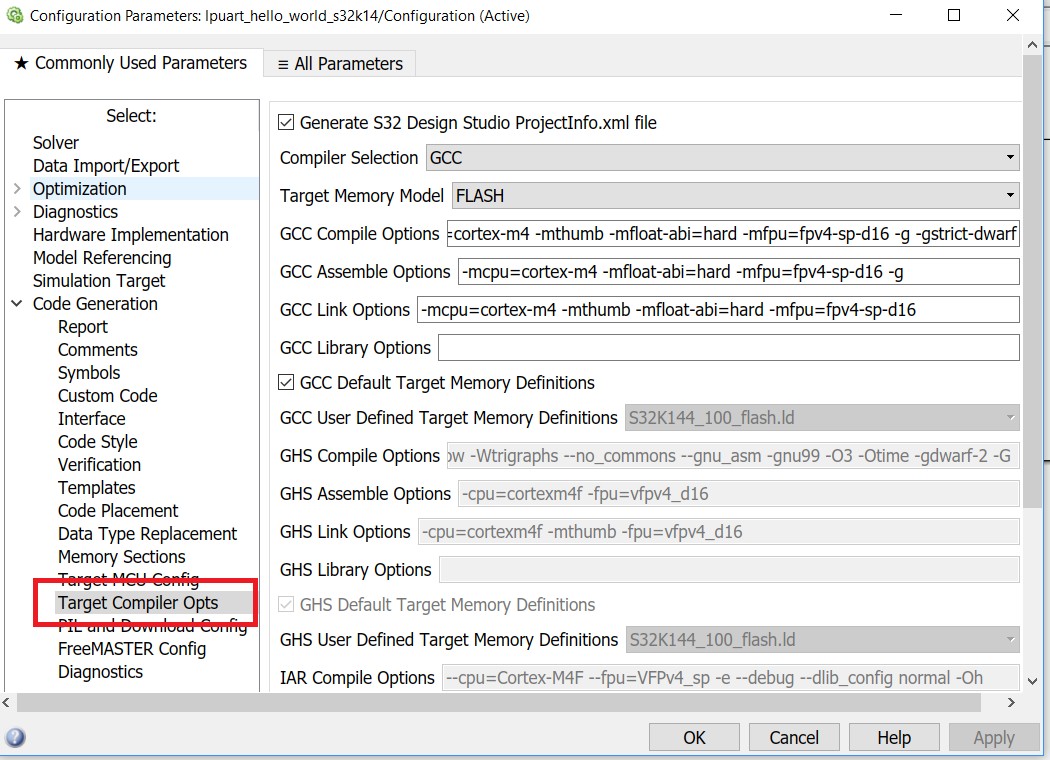
L ”I 2C\_l S R

Toggle blue LE D at TX and 'X evant

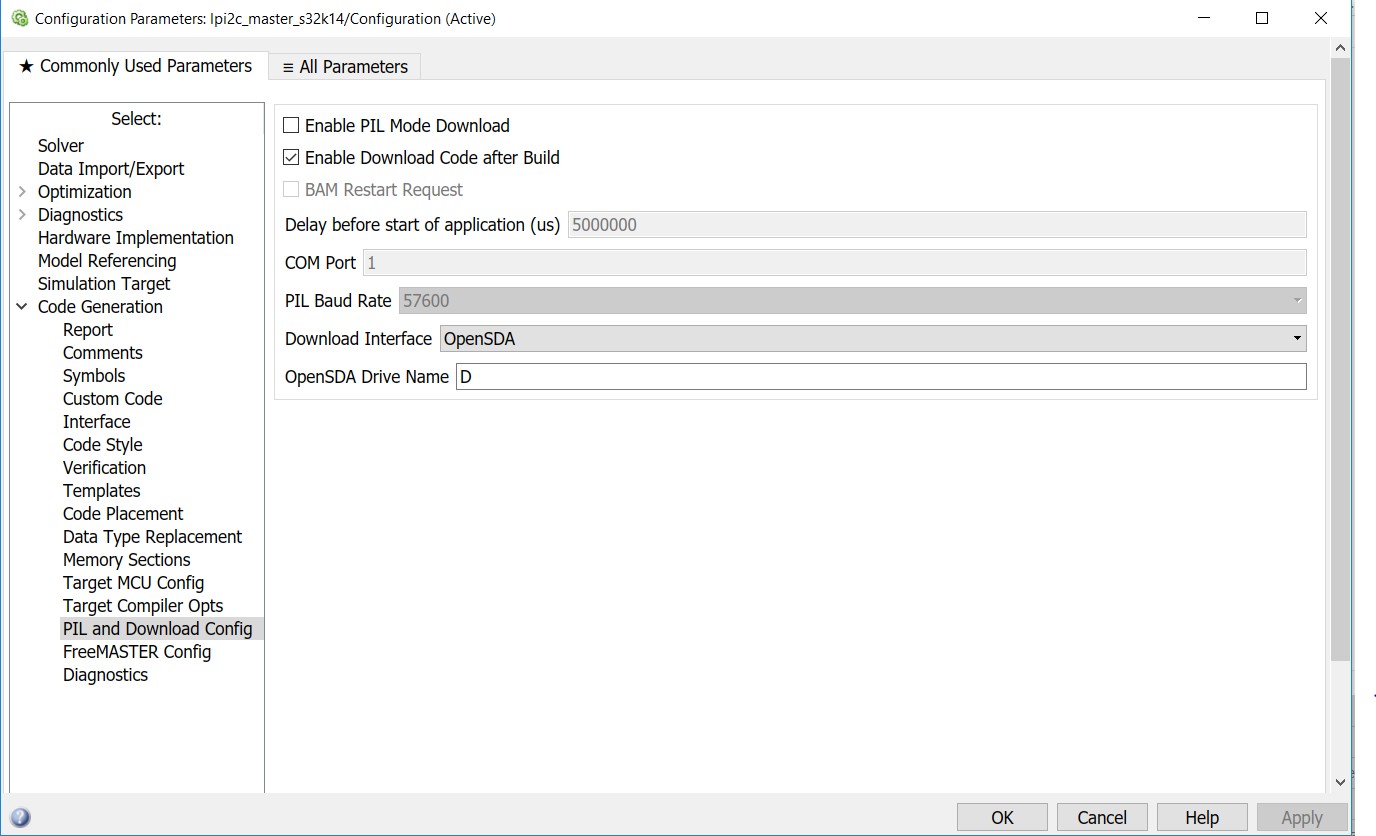
Target MCU Config:



Target Compiler Opts:



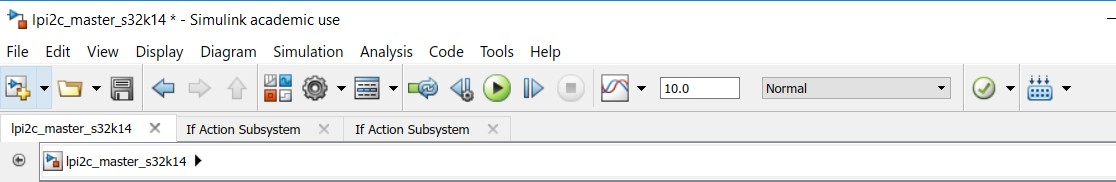
PIL and Download Config:



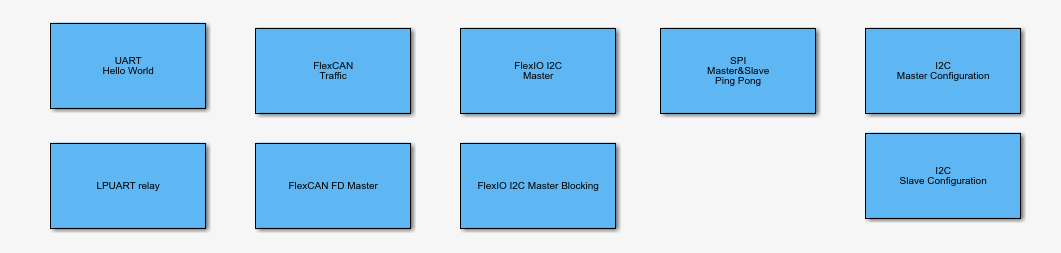
* + Be sure that you have the correct OpenSDA Druve Name (for example, D)



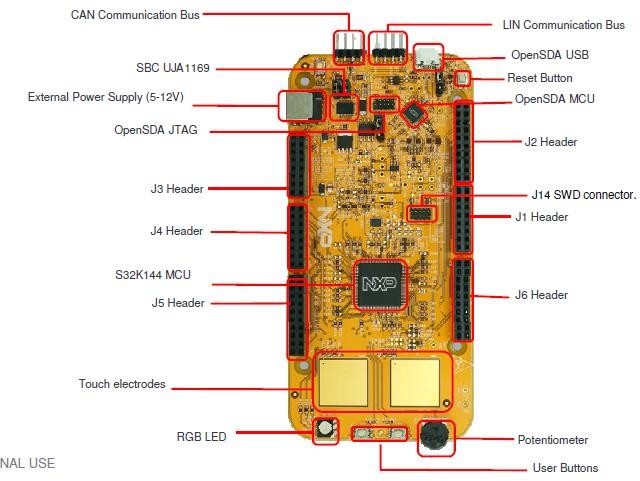
* + Apply and close this window. Press Build Model button and wait until the code is generated, compiled and downloaded to the evaluation board.



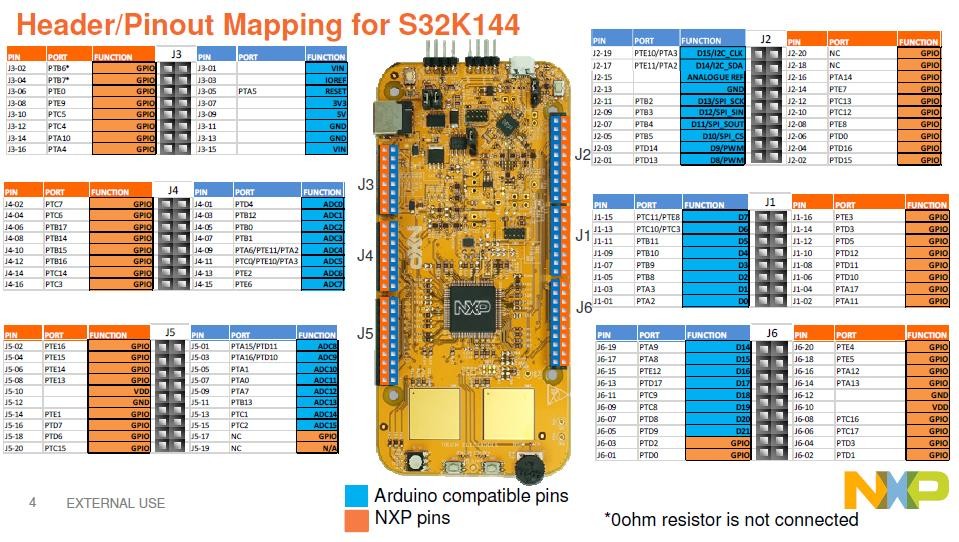
* + Once the Model is built, the Blue LED should be on the S32K144 board which indicated that it is successfully programmed to be a Master to which a Slave can be connected using I2C protocol.
  + Disconnect the board and connect another one through the USB cable.
  + Follow the same steps as before and select the I2C Slave Configuration model.



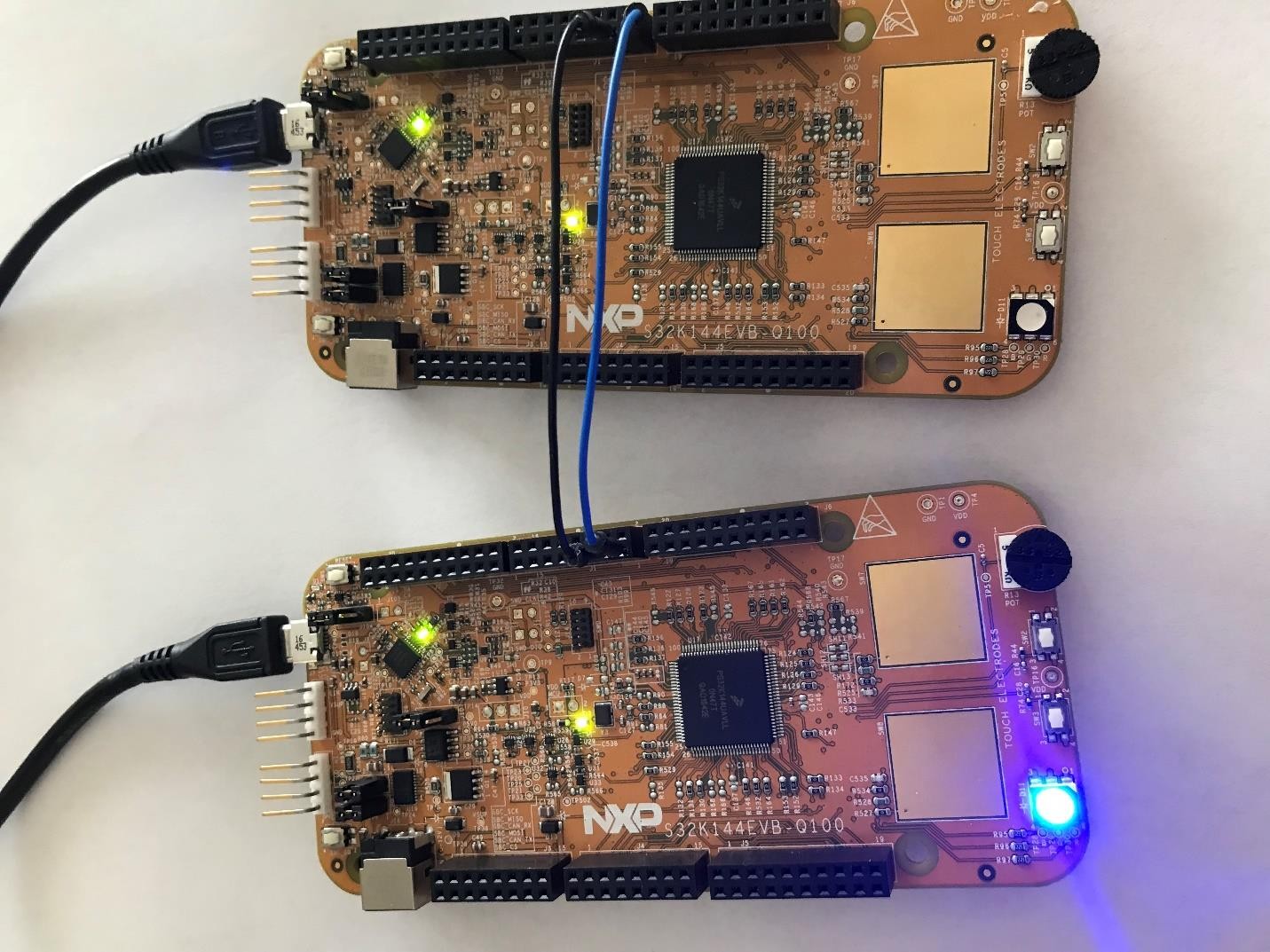
* + As before, make sure the project configuration settings are correct then build the slave model.
  + Once the Slave model is built, the GREEN LED glows on the Slave board.
  + Keep the slave board connected to the PC then connect the slave board to the master Board through the slave PTA2 (J1,1) pin to master PTA2 (J1,1) pin and slave PTA3 (J1,3) pin to master PTA3 (J1,3) pin.



SW3 SW2



* + One of the wires acts as a Serial Data Pin and the other wire acts as a Serial Clock Pin.



* + Once the connections are made, connect the master board to the PC.
  + Now the two boards are connected in a Master-Slave configuration using I2C protocol.
  + Pressing SW3 on the Master board sends a message from master to slave (data will increment after each send. blue LED will be toggled on each step). Pressing

SW2 on the Master Board request data from slave (last received data from slave).

* + The blue LED on the Slave board starts ON. On 2nd request for data from master the blue LED turns OFF. On 3nd request for data from master the blue LED turns ON.