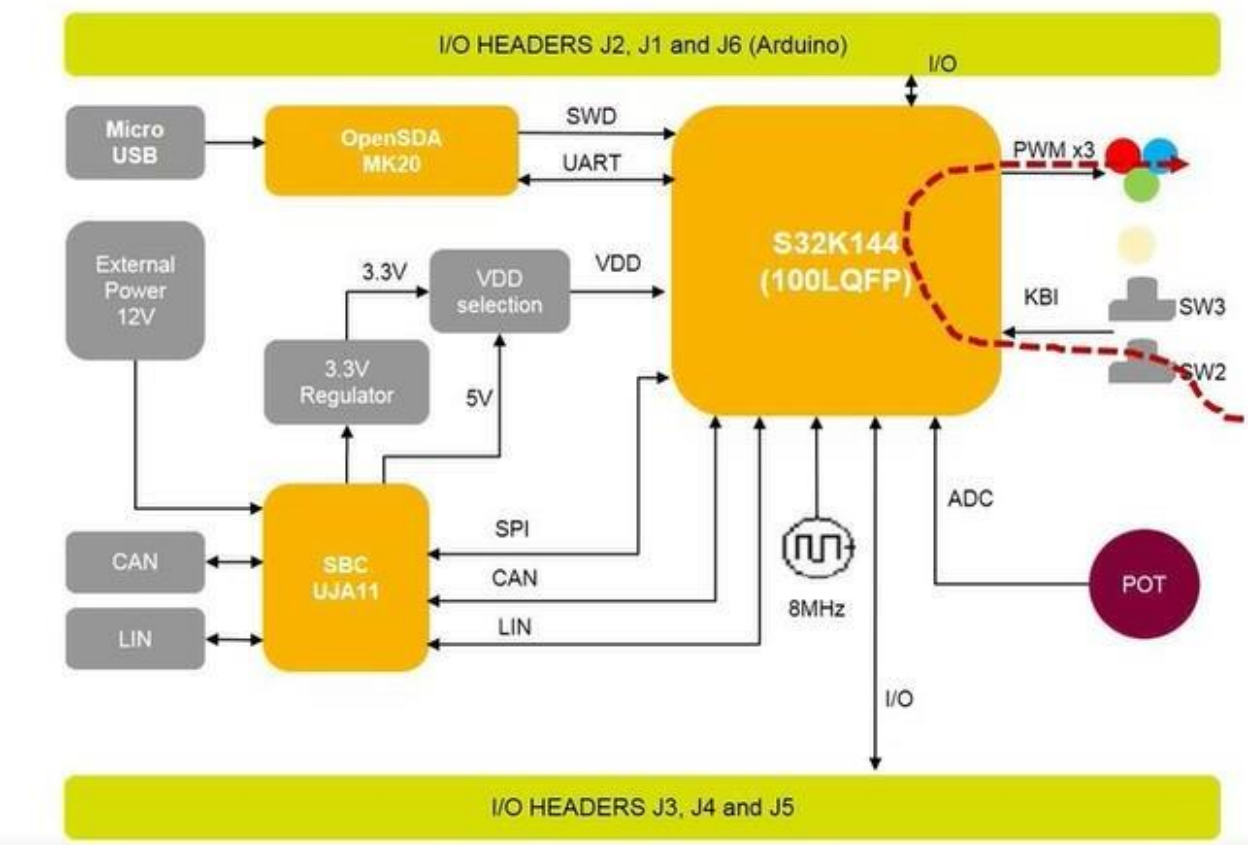
MBDT and FreeMASTER

1. MBDT Toolbox version 4 was used in lab 1. Same can be used to run this lab.

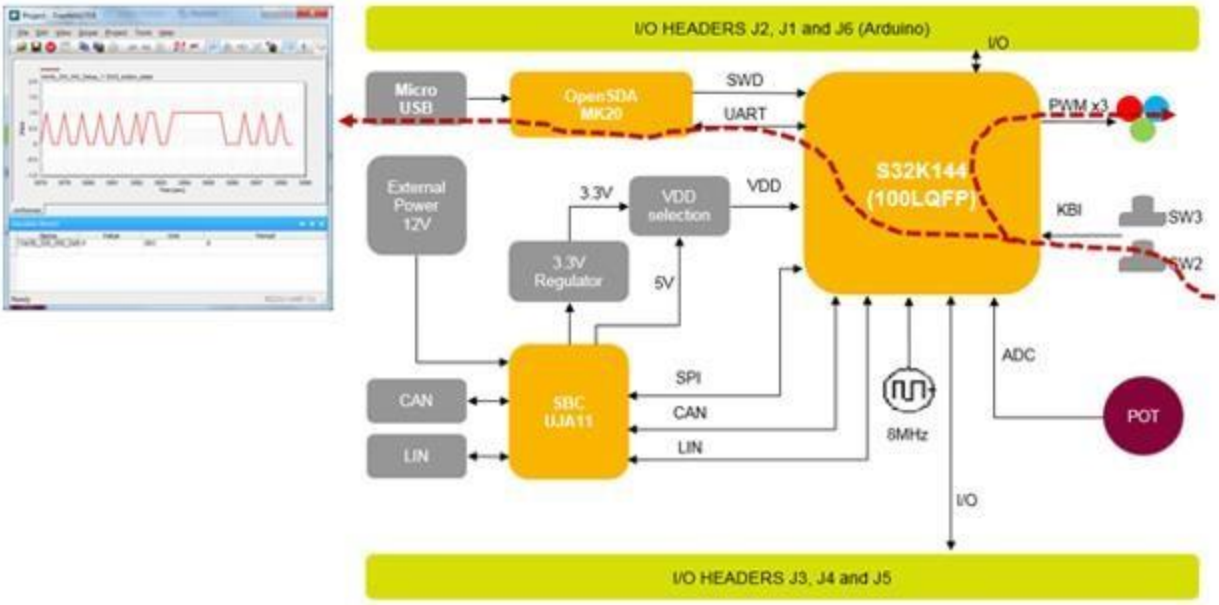
II. GPIO Application

- 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.

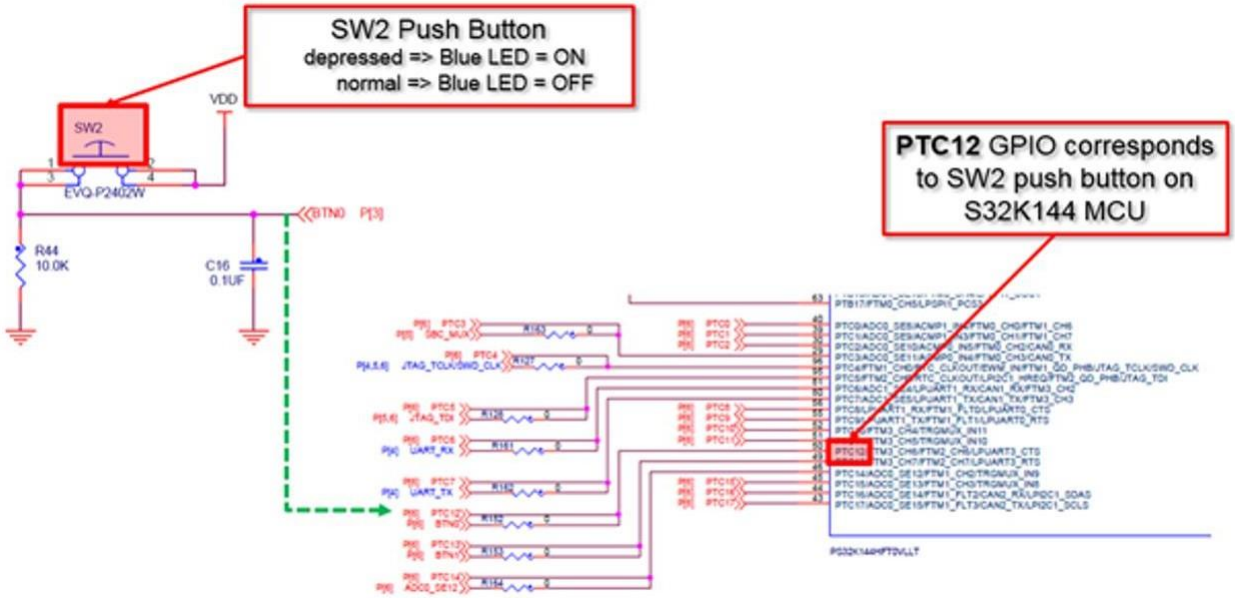
- The application flow is shown next. The code that runs on S32K144 MCU will be pooling at each 0.1 second the state of the SW2 push button and depending on the value read via the associated digital input will update the state of the Blue LED by writing 1 or 0 to the associated digital output.



- 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.

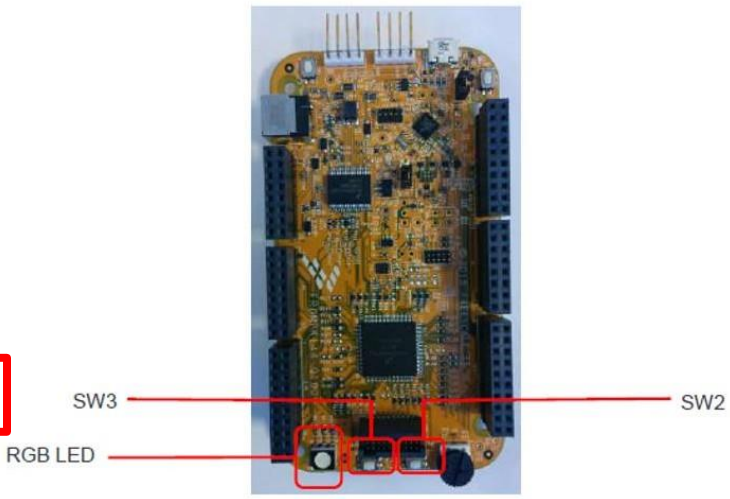


- RGB LED

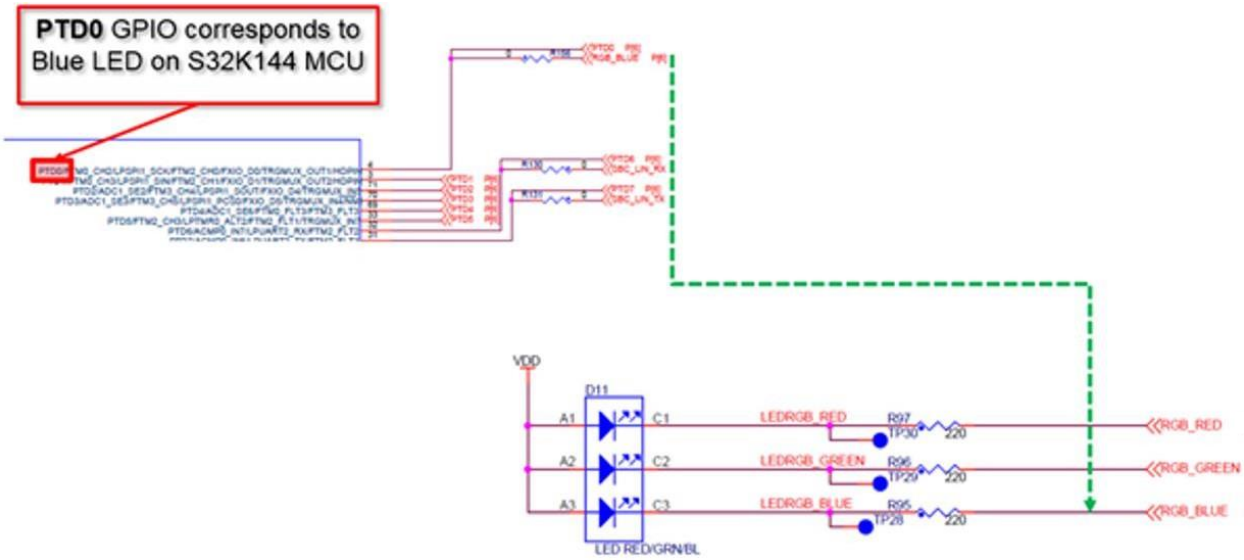
LED	S32K144 PIN	Pull resistor
BLUE	PTD0	Pull up
RED	PTD15	Pull up
GREEN	PTD16	Pull up

- SW2 and SW3

Button	S32K144 PIN	Pull resistor
SW2	PTC12	Pull down
SW3	PTC13	Pull down

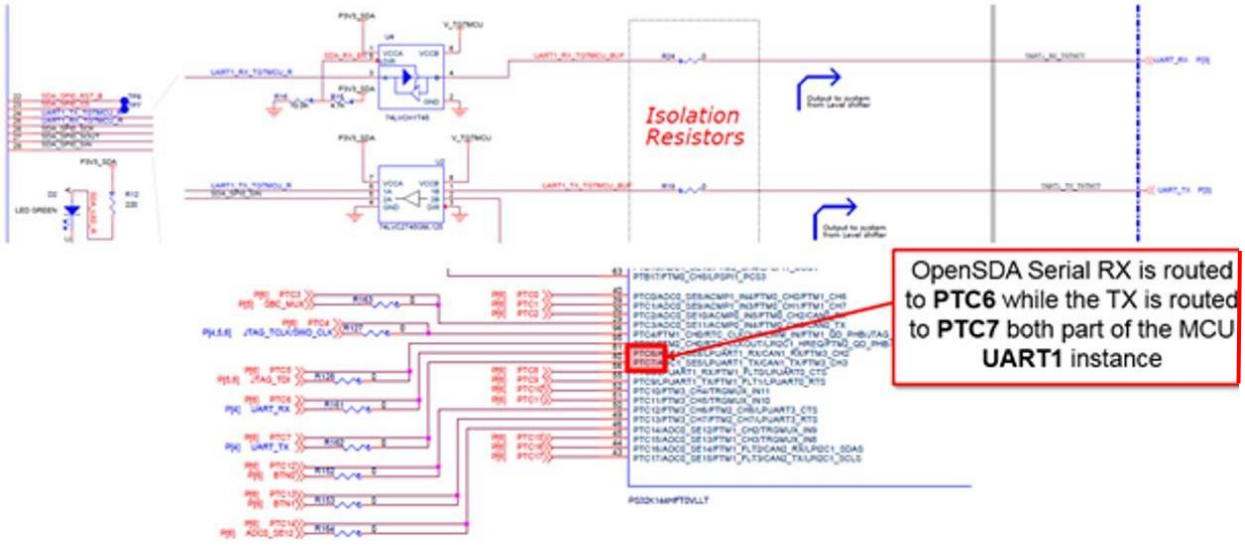


- The Blue LED is associated with the PTD0 general-purpose pin. Each time the SW2 push button will be depressed we need to switch the Blue LED to ON.



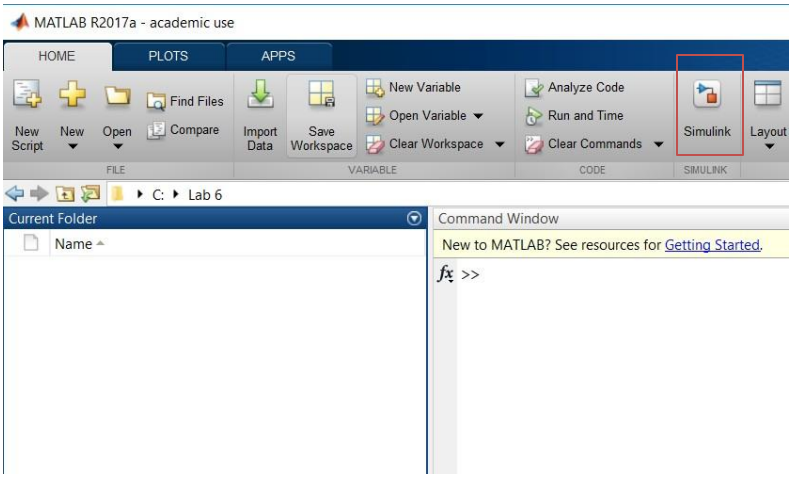
- To communicate with the S32K144 MCU, we are going to use the built-in OpenSDA that will have a dual purpose: to load/flash the application into the MCU flash memory and to assist with serial communication over the virtual COM port mapped on top of physical USB.

- 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.

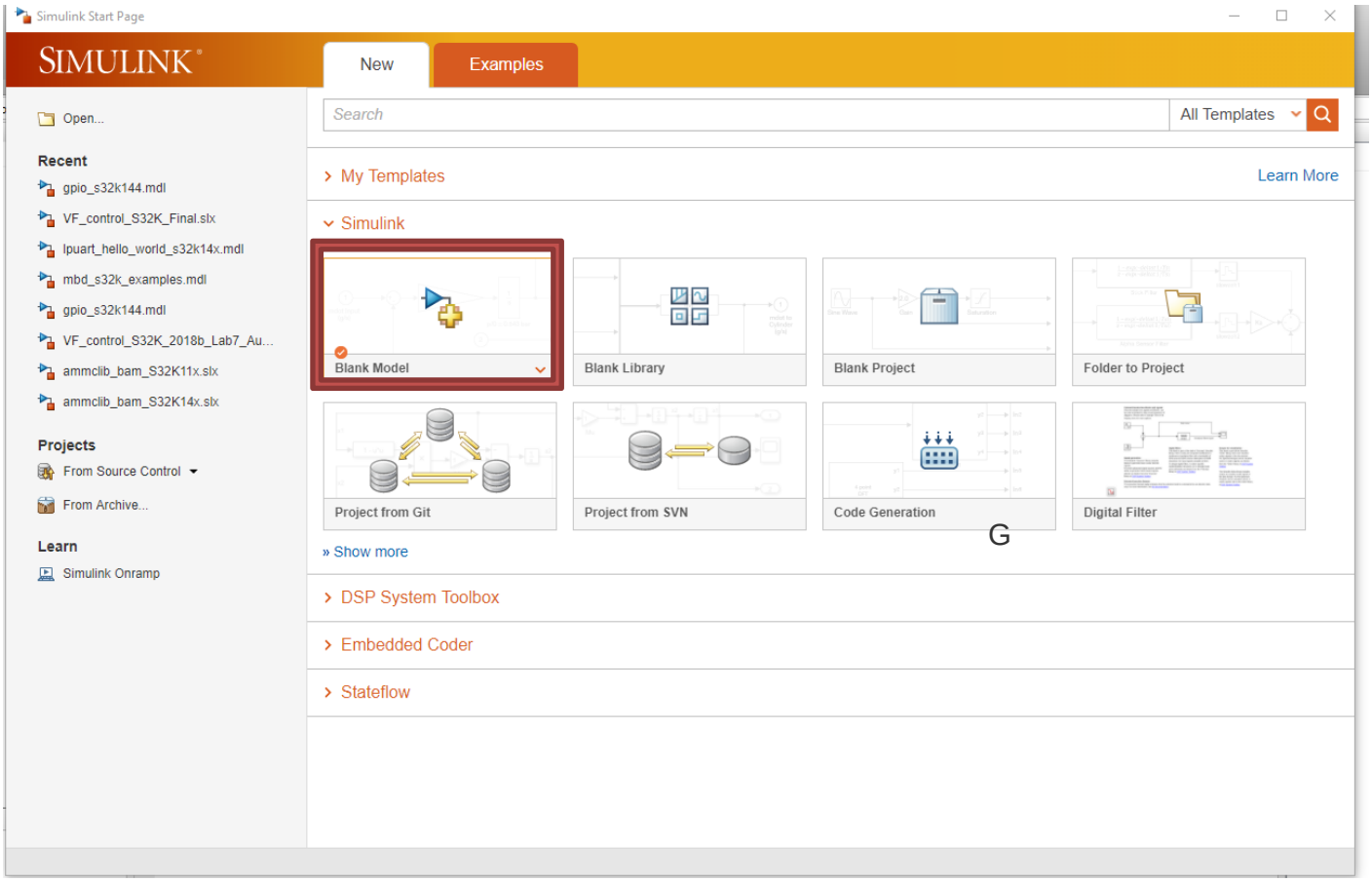


- The entire example can be implemented using standard Simulink blocks from NXP's Model Based Design Toolbox for S32K1xx Simulink Library.

- Start MATLAB to open a new Simulink model.



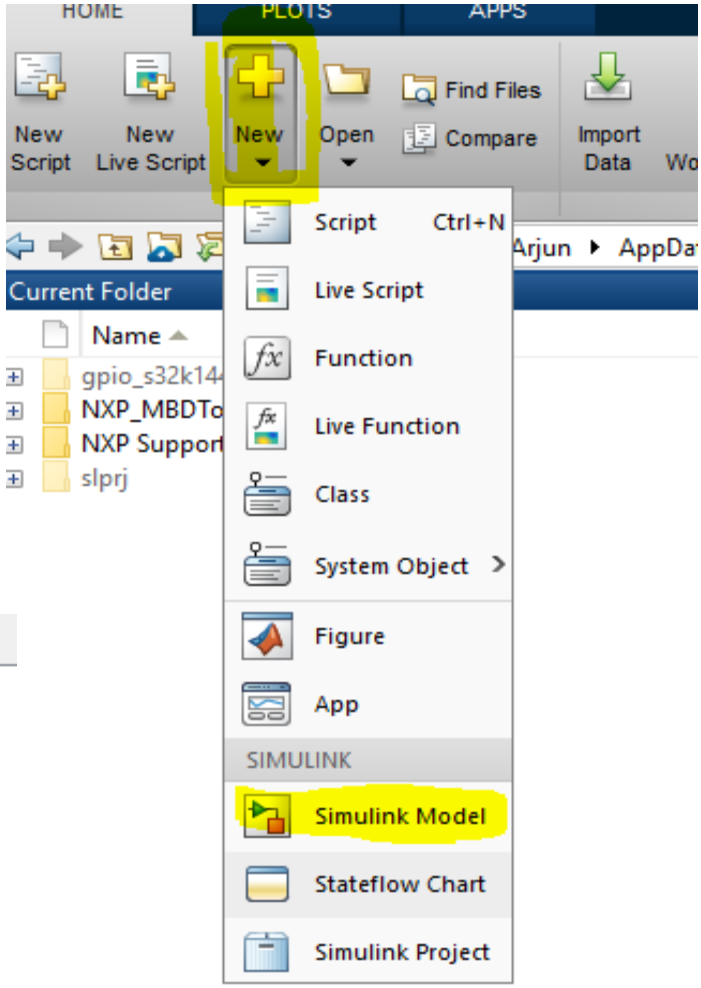
- Click on the Blank Model.

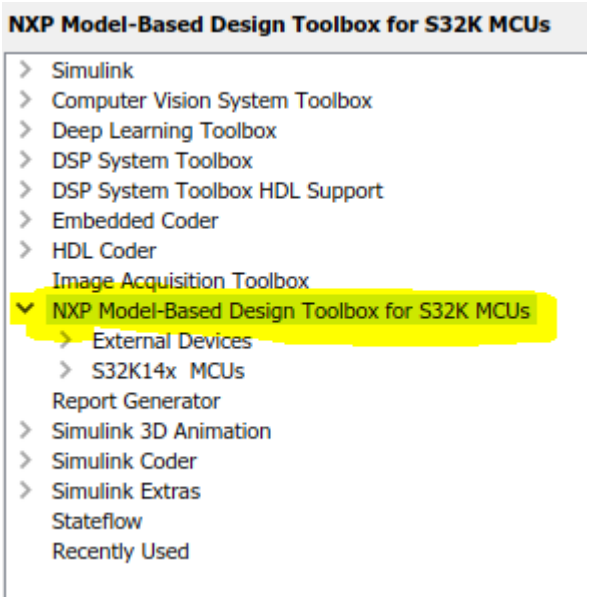


G

- 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 

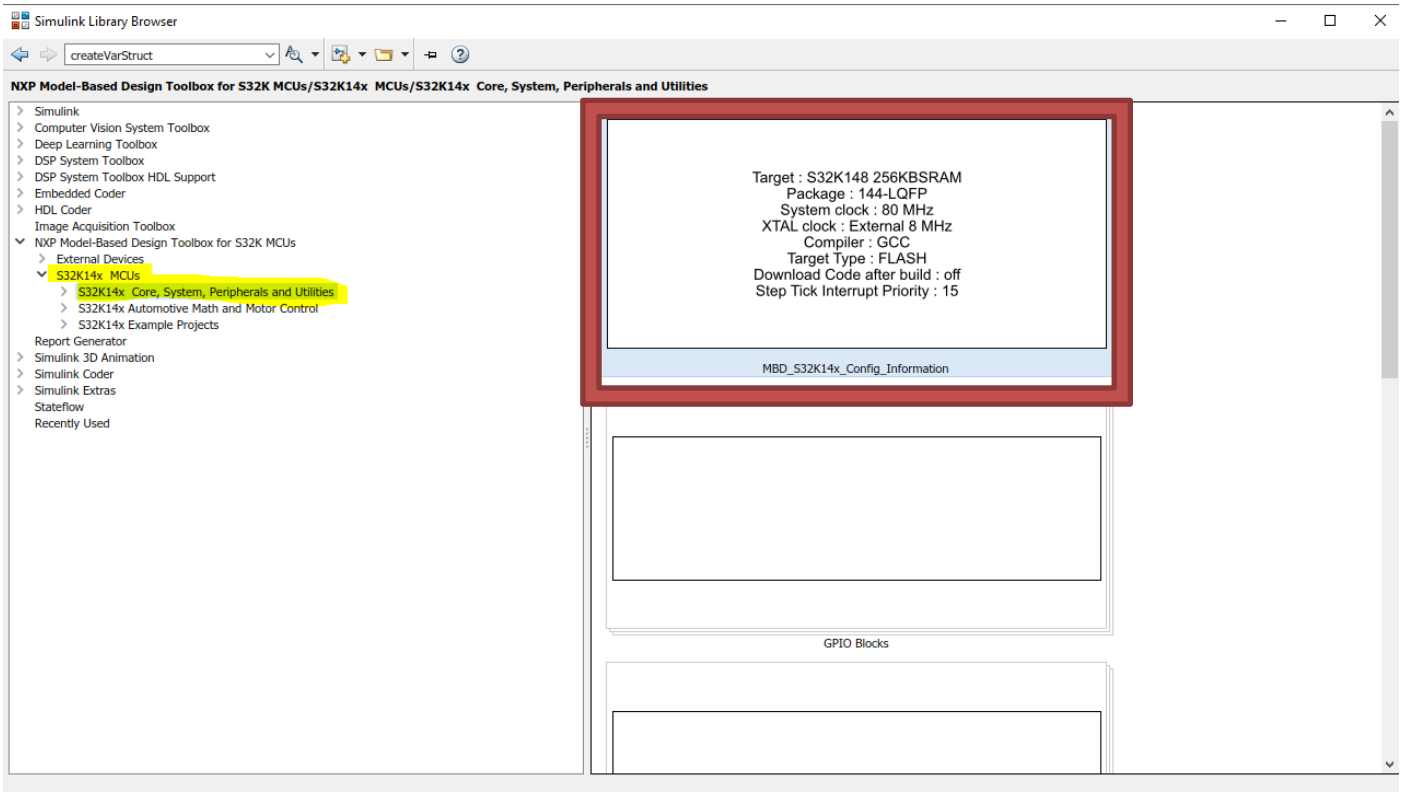
  and select the “NXP Model Based Design Toolbox for S32K MCUs”.

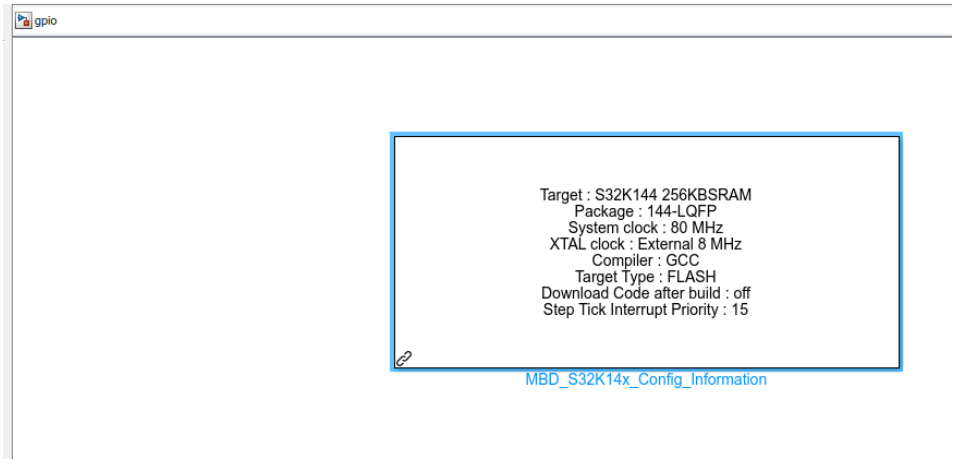




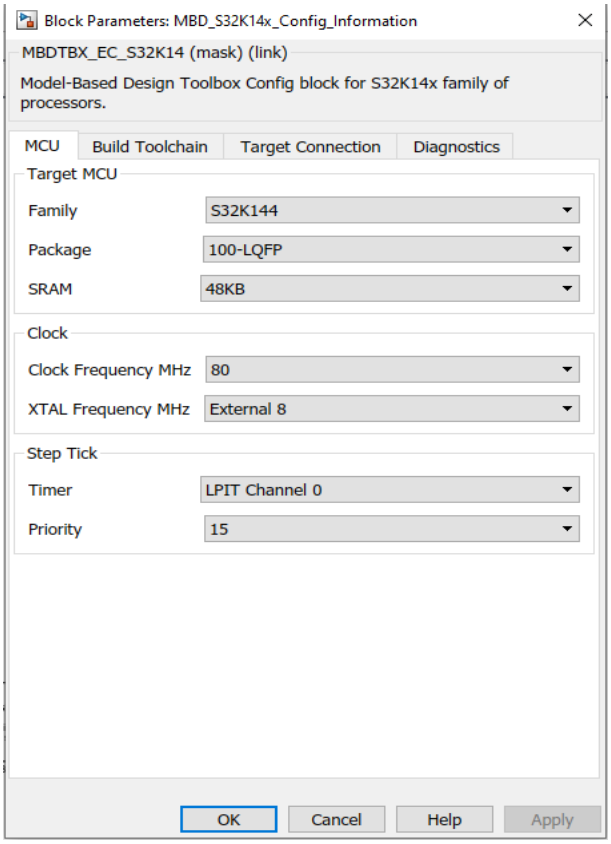
1. Adding Config Block to the Model

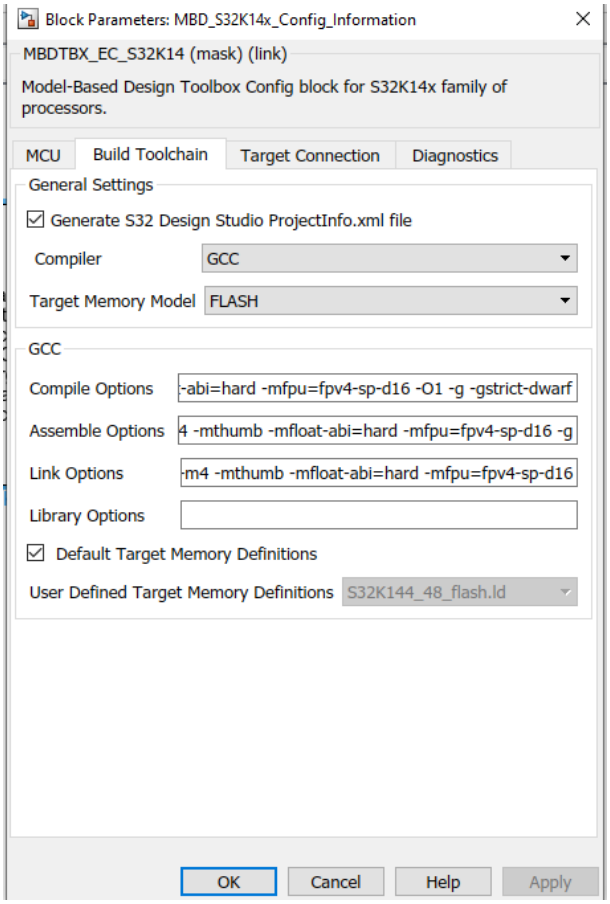
- Navigate to **S32K14x MCUs → S32K14x Core System Peripheral and Utilities** and **Right Click** on **MBD_S32K14x Config_Information block** as in below figure and select ***'Add block to model'***

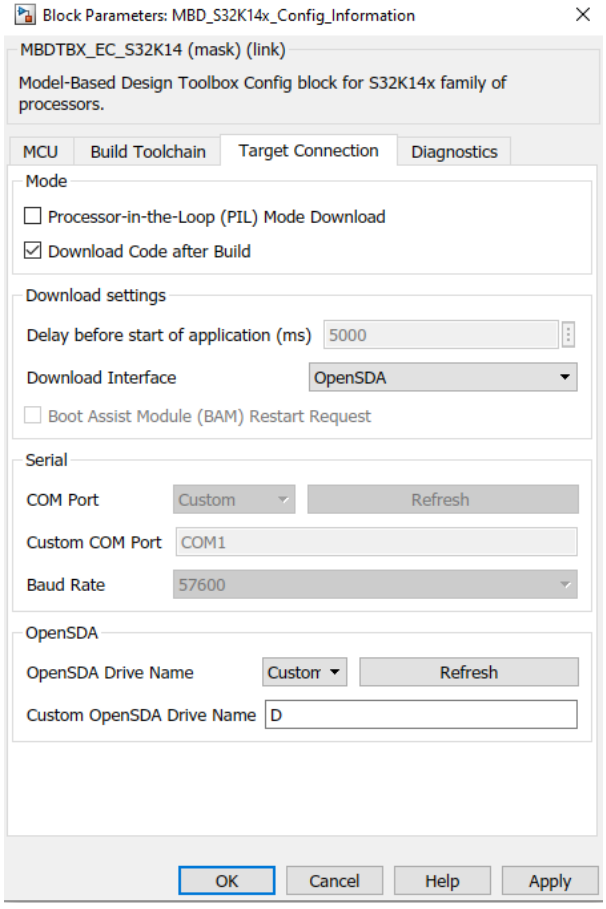




- Double click on "MBD_S32K14x_Config_Information" block.
- Set "MCU", "Build Toolchain", "Target Connection" parameter as follows

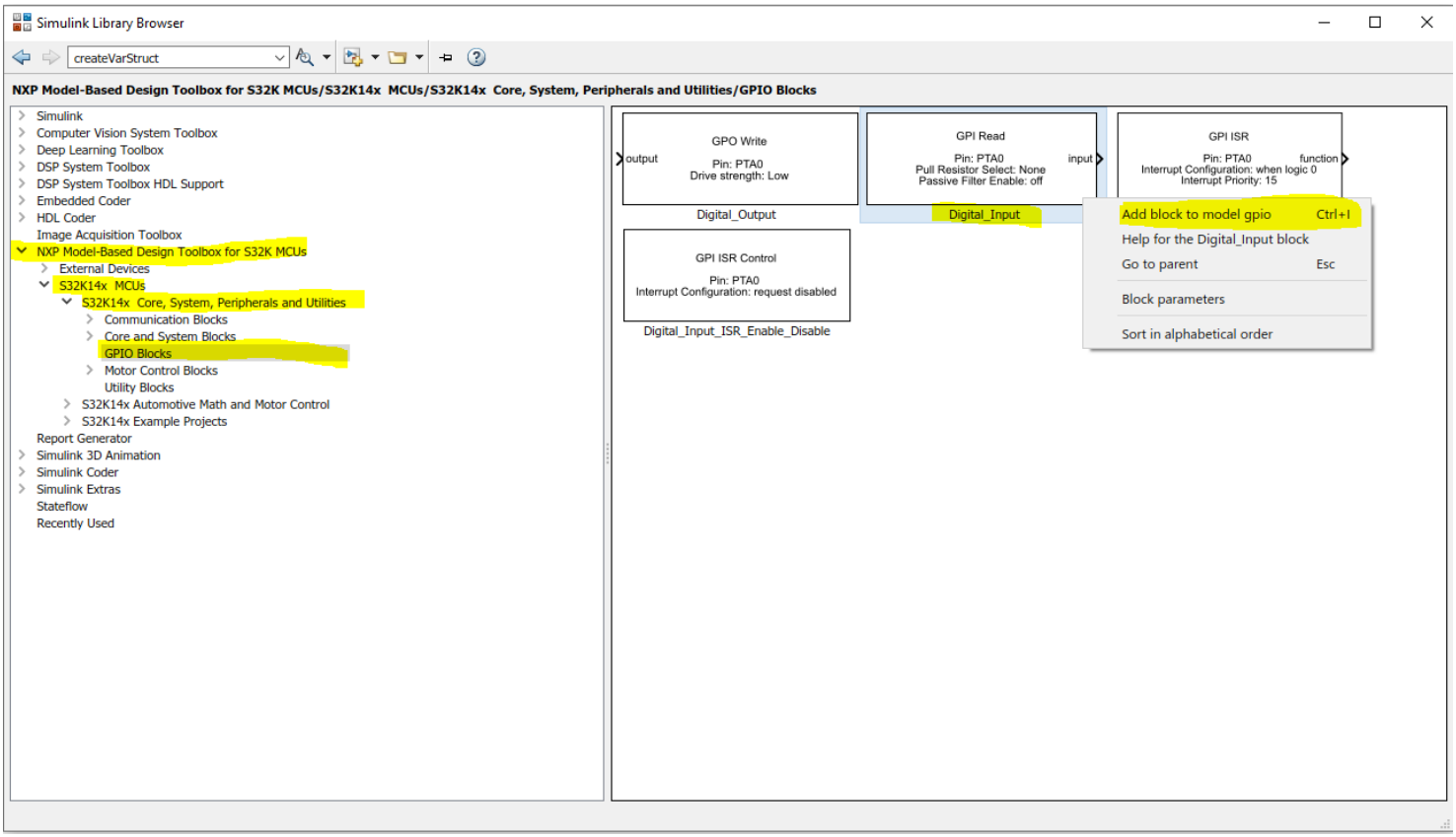

Block Parameters: MBD_S32K14x_Config_Information
MBDTBX_EC_S32K14 (mask) (link)
Model-Based Design Toolbox Config block for S32K14x family of processors.
MCU
Build Toolchain
Target Connection
Diagnostics
Target MCU
Family
S32K144
Package
100-LQFP
SRAM
48KB
Clock
Clock Frequency MHz
80
XTAL Frequency MHz
External 8
Step Tick
Timer
LPIT Channel 0
Priority
15
OK
Cancel
Help
Apply


Block Parameters: MBD_S32K14x_Config_Information
MBDTBX_EC_S32K14 (mask) (link)
Model-Based Design Toolbox Config block for S32K14x family of processors.
MCU
Build Toolchain
Target Connection
Diagnostics
General Settings
Generate S32 Design Studio ProjectInfo.xml file
Compiler
GCC
Target Memory Model
FLASH
GCC
Compile Options
-abi=hard -mfpu=fpv4-sp-d16 -O1 -g -gstrict-dwarf
Assemble Options
4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -g
Link Options
-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16
Library Options
Default Target Memory Definitions
User Defined Target Memory Definitions
S32K144_48_flash.ld
OK
Cancel
Help
Apply


Block Parameters: MBD_S32K14x_Config_Information
MBDTBX_EC_S32K14 (mask) (link)
Model-Based Design Toolbox Config block for S32K14x family of processors.
MCU
Build Toolchain
Target Connection
Diagnostics
Mode
Processor-in-the-Loop (PIL) Mode Download
Download Code after Build
Download settings
Delay before start of application (ms)
5000
Download Interface
OpenSDA
Boot Assist Module (BAM) Restart Request
Serial
COM Port
Custom
Refresh
Custom COM Port
COM1
Baud Rate
57600
OpenSDA
OpenSDA Drive Name
Custom
Refresh
Custom OpenSDA Drive Name
D
OK
Cancel
Help
Apply

2. 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.

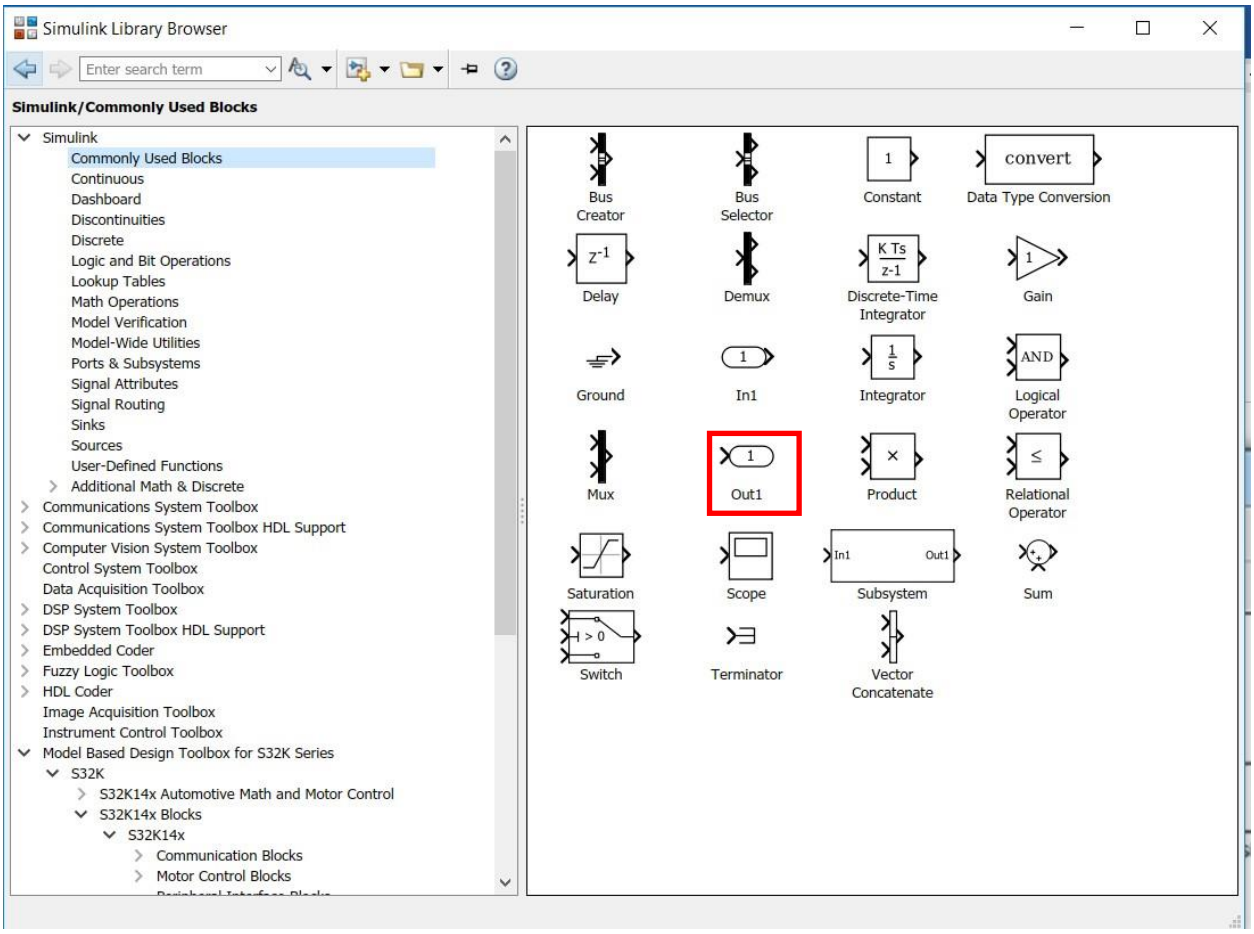


- Similarly add **Digital_Output** Block to the model.

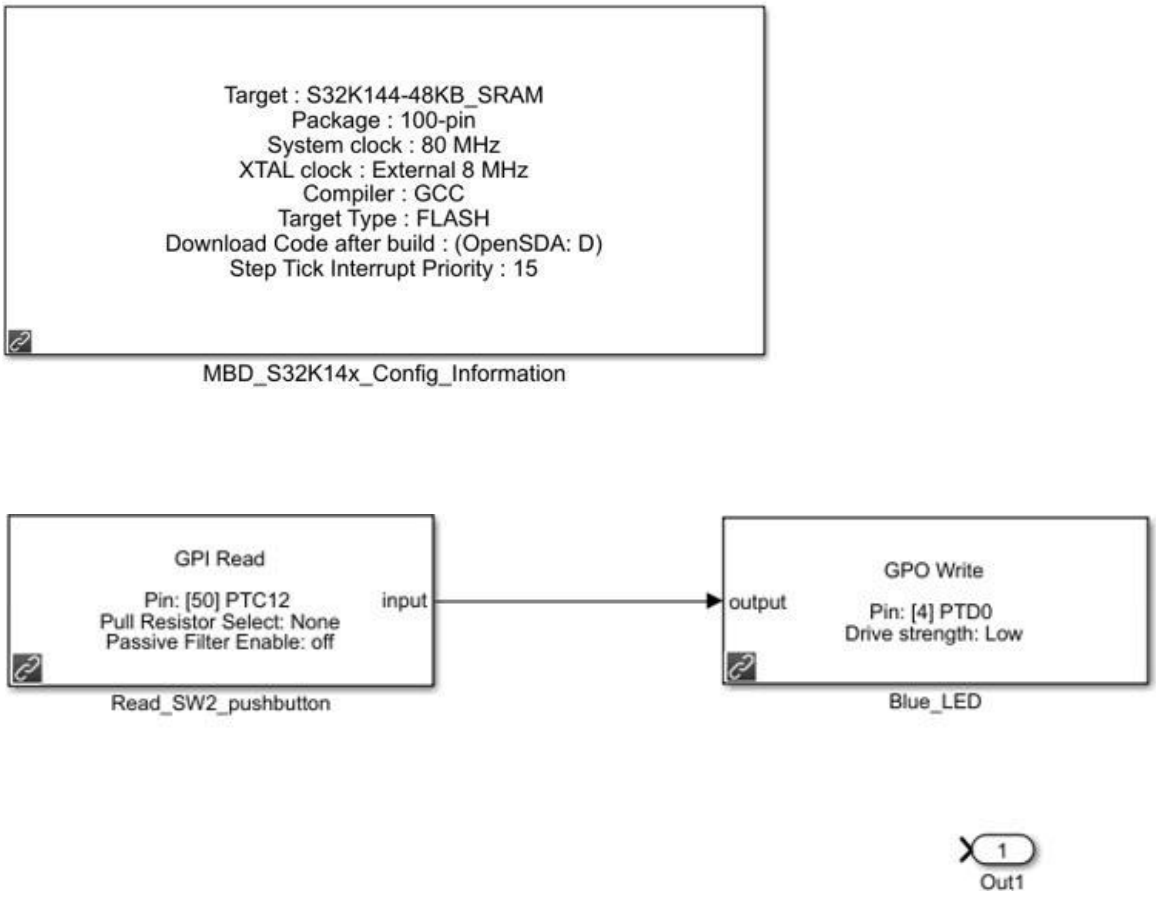
3. Configure the Digital Input/ Output blocks.

- 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”.

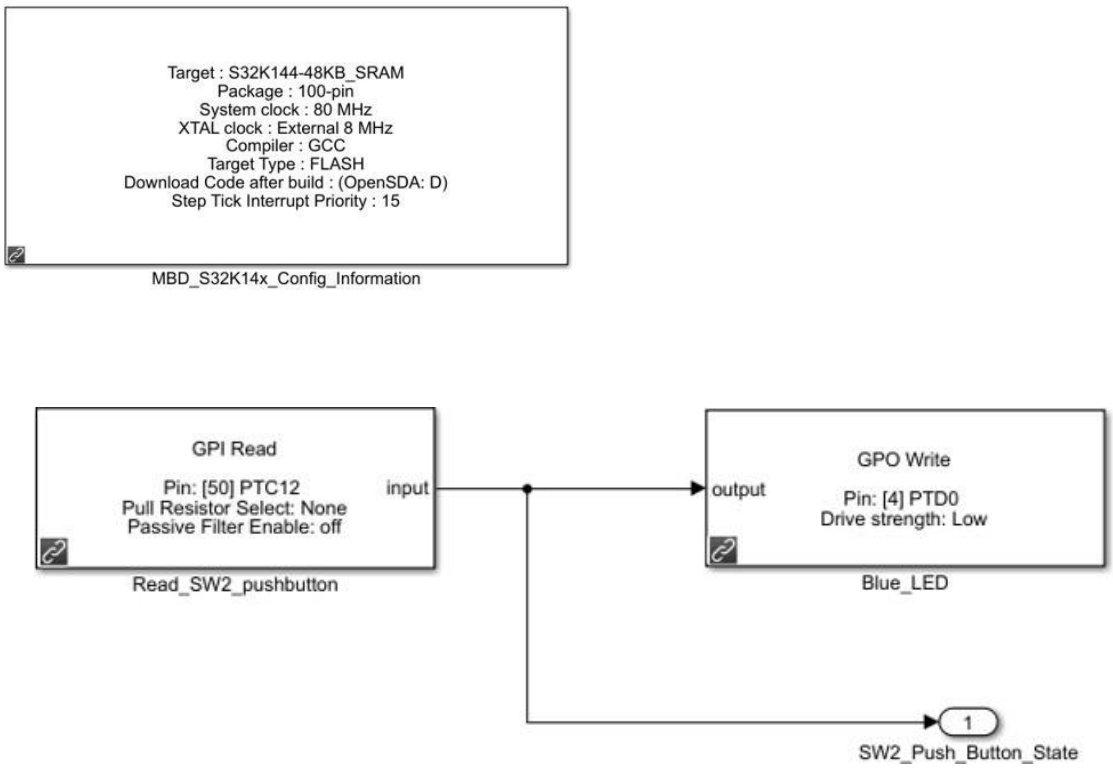


  - Right click on “Out1” and select “Add block to model GPIOLAB”.

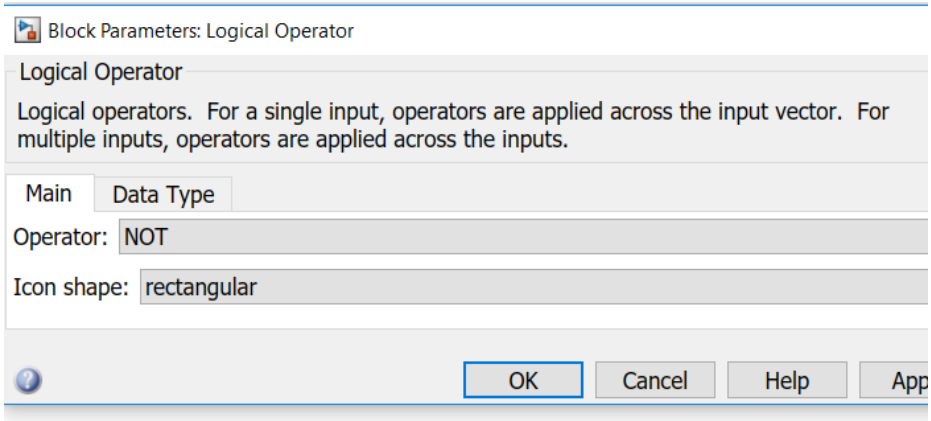


- Rename the “Out1” block as “SW2_Push_Button_State”.

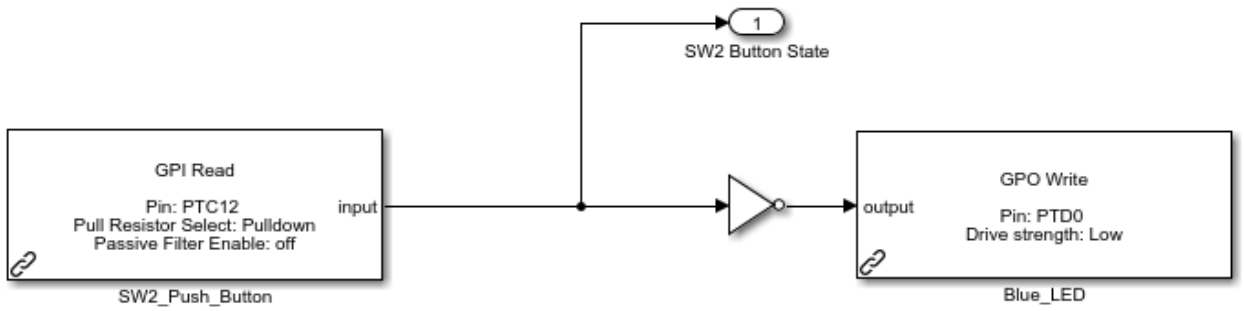
- Connect the Input of the “GPI Read” to the “SW2_Push_Button_State”.



- 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.”

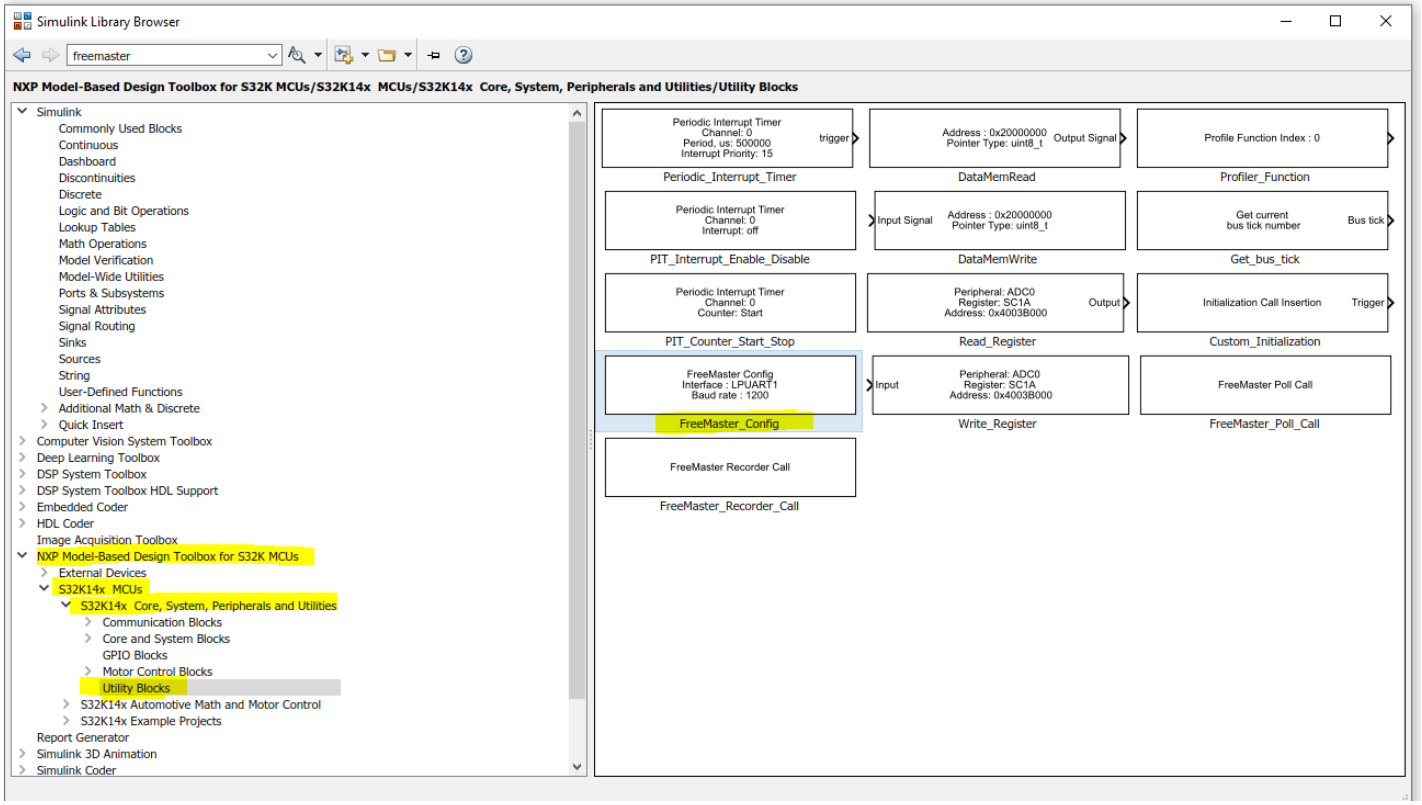


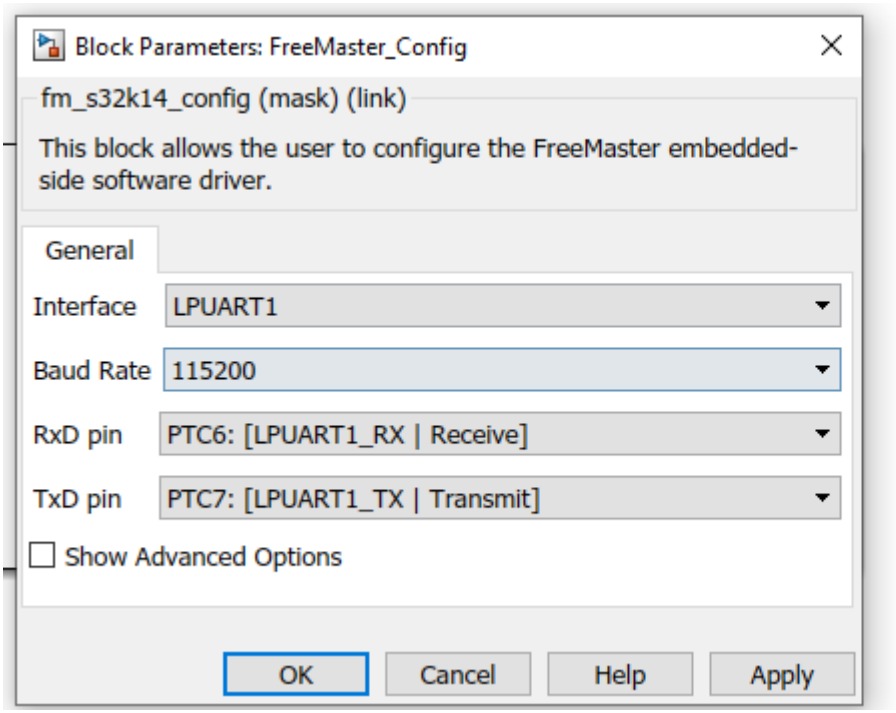
- Move the “Logical Operator” block between the digital input and digital output. Finally, it looks as follows:



4. FreeMaster Configuration

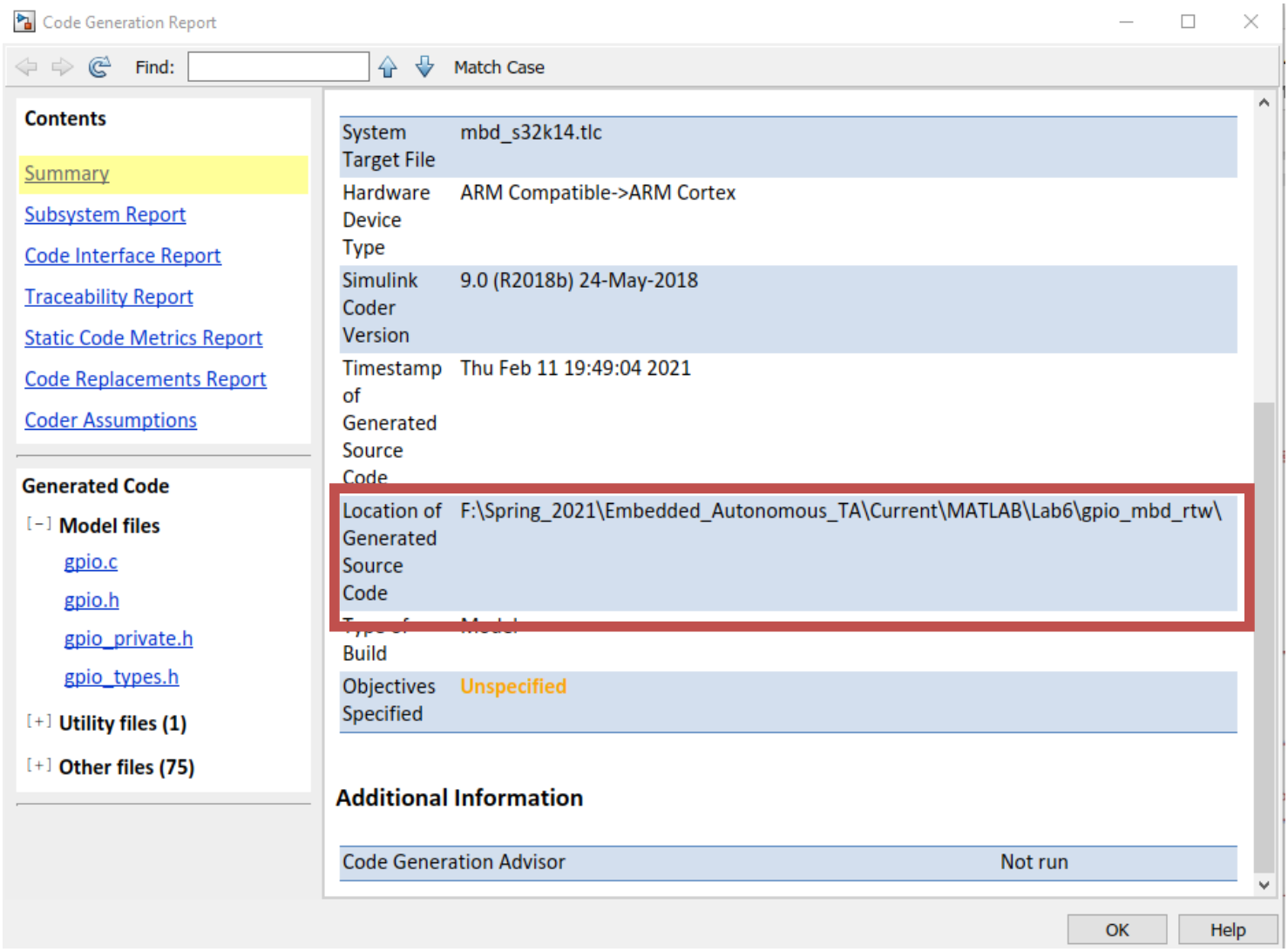
- Add FreeMaster_Config box to the model and configure it as the image in the next page.
  - R





- 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.

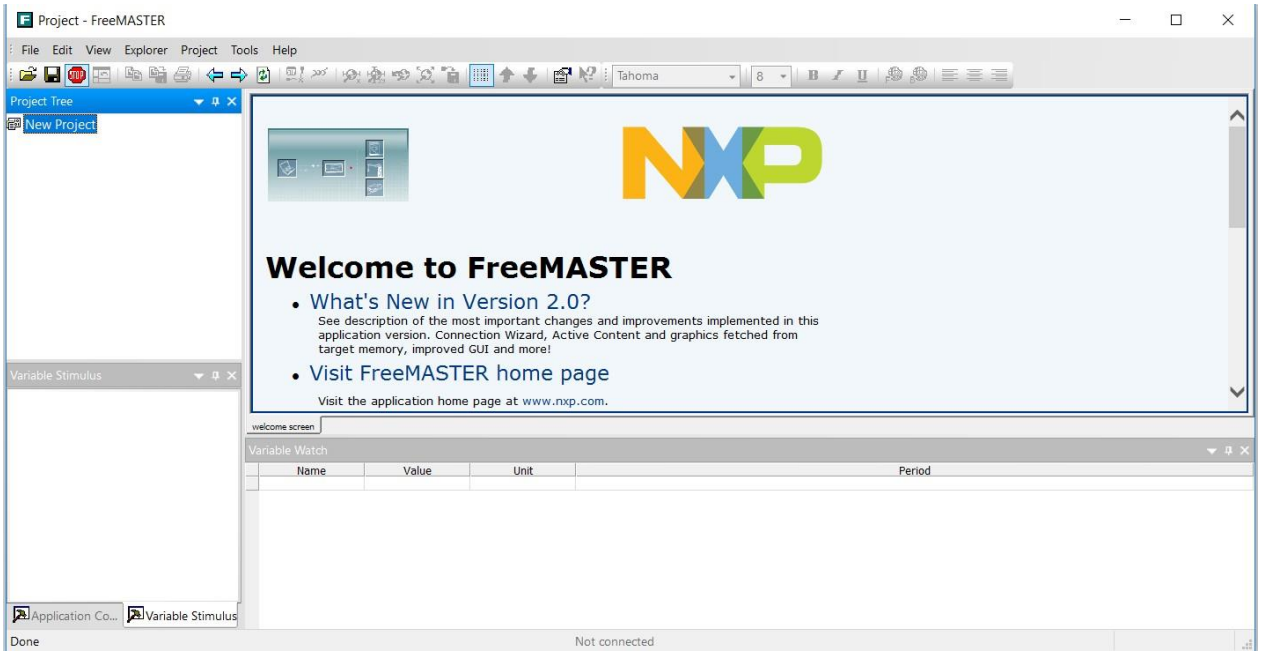
The Code Generation Report pops up and observe the location of Generated Source Code. This folder with contain all the codes generated for your application. Now you can also investigate the folder where there will be a directory named like **project name_mbd_rtw** this folder contains an **project_name.mot** 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.



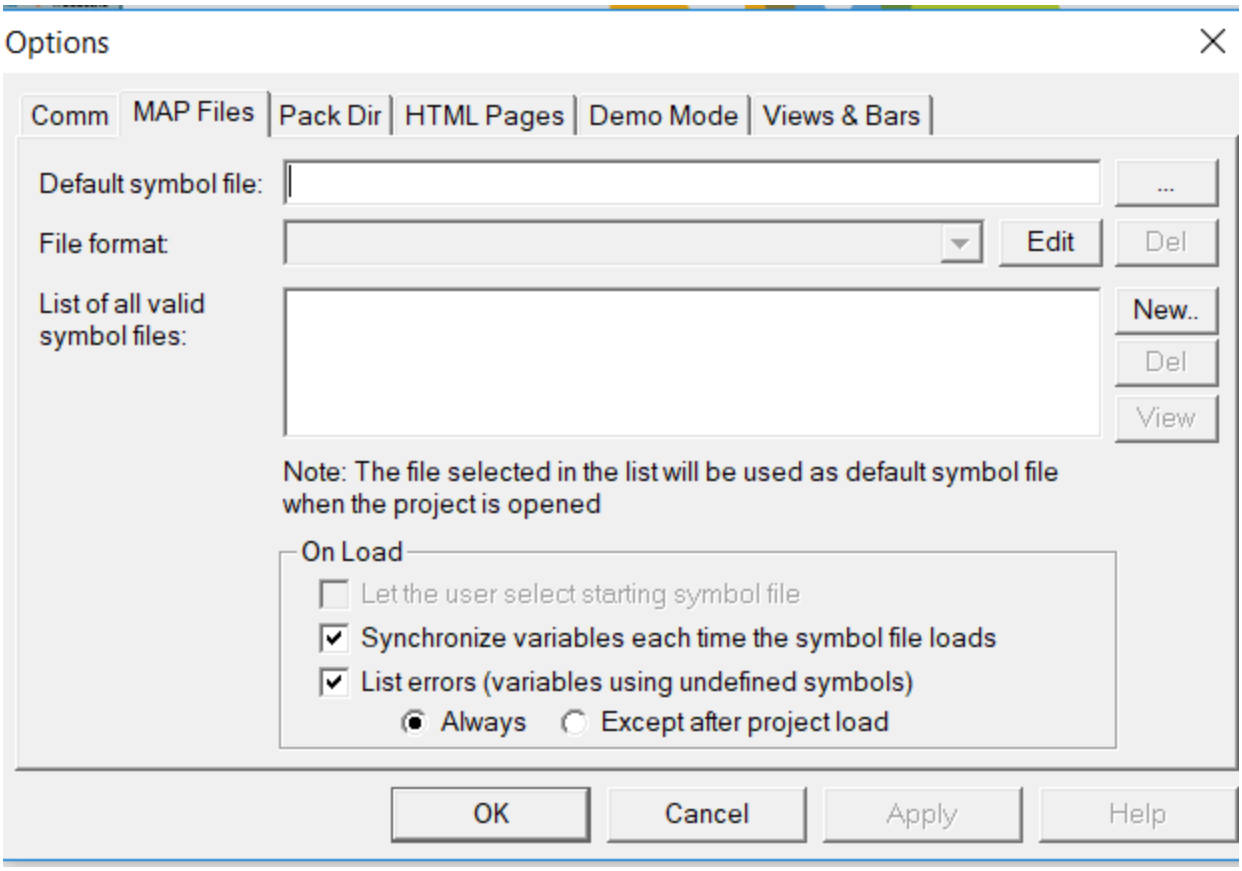
- The blue LED toggles when you push SW2.

Viewing the Variables in the Application via FreeMASTER

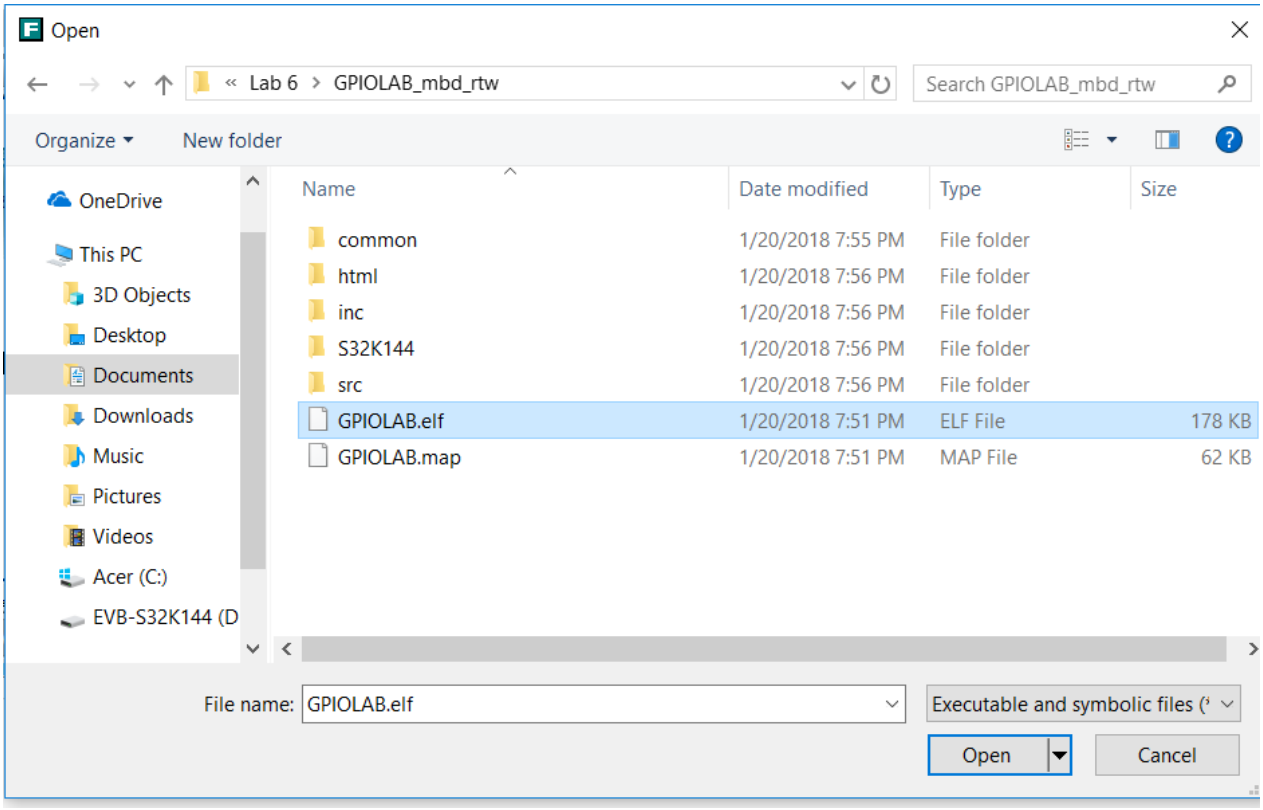
- Start FreeMASTER software.



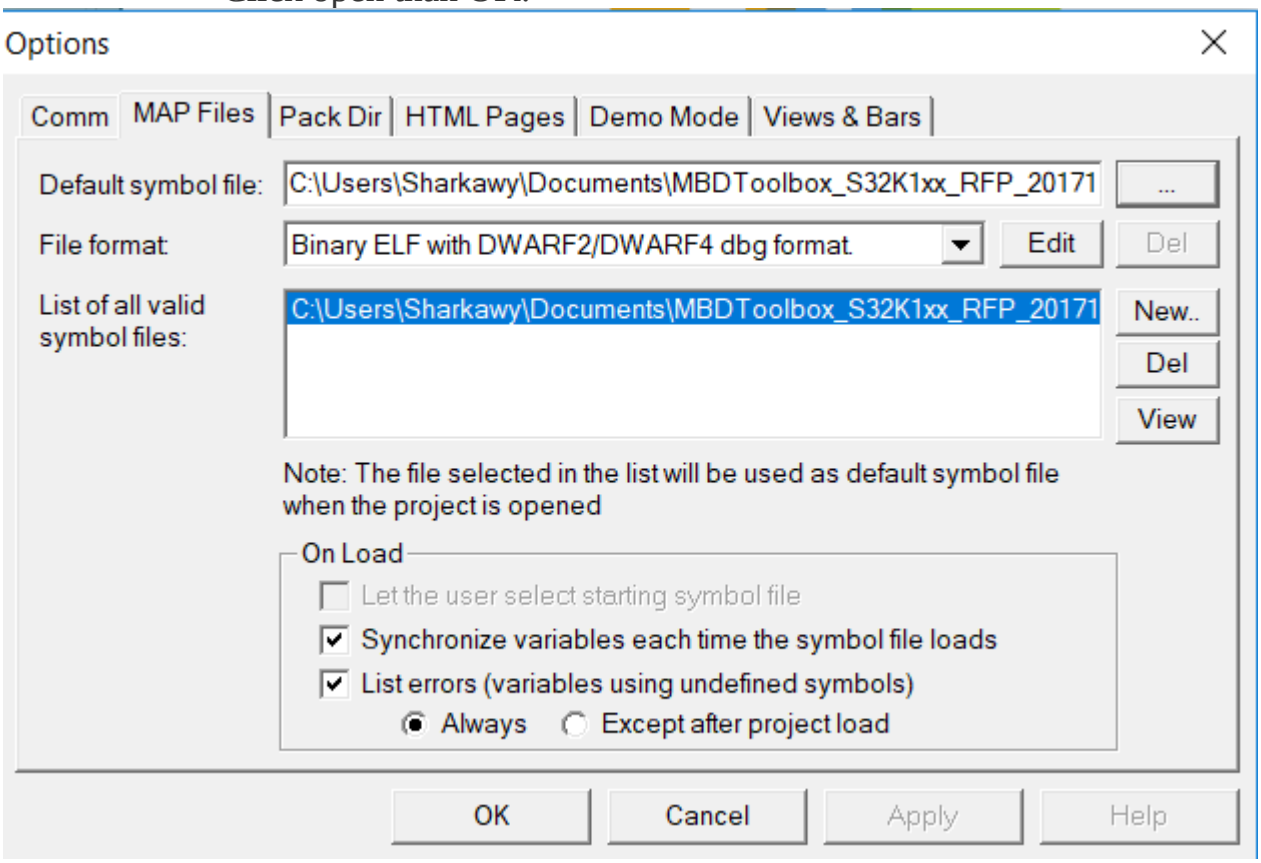
- Select **Project → Options → MAP Files**.



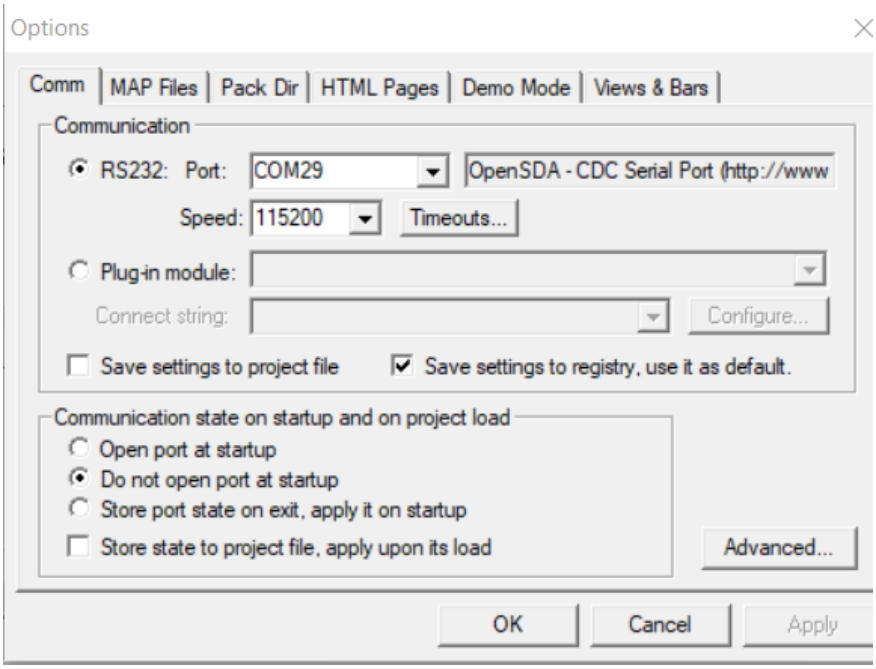
- Click on  and locate the **"project_name.elf"** that you have already generated in the **location of generated code**.



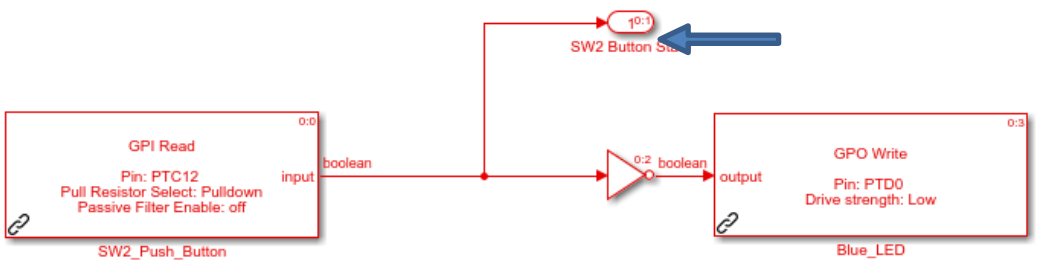
- Click open than OK.

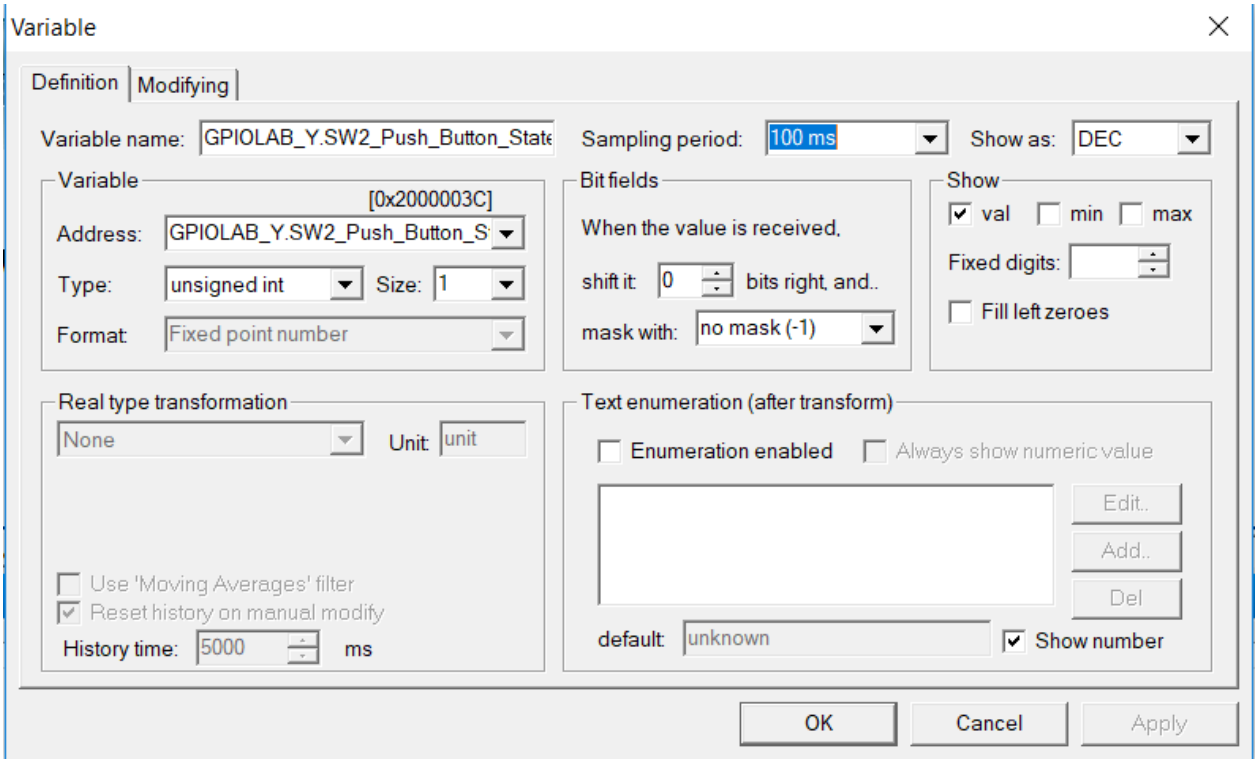


- Check your com port at the device manager.
- Select “Comm” and set your communication by selecting RS232 → Port select the COM port of your MCU and speed as 115200.

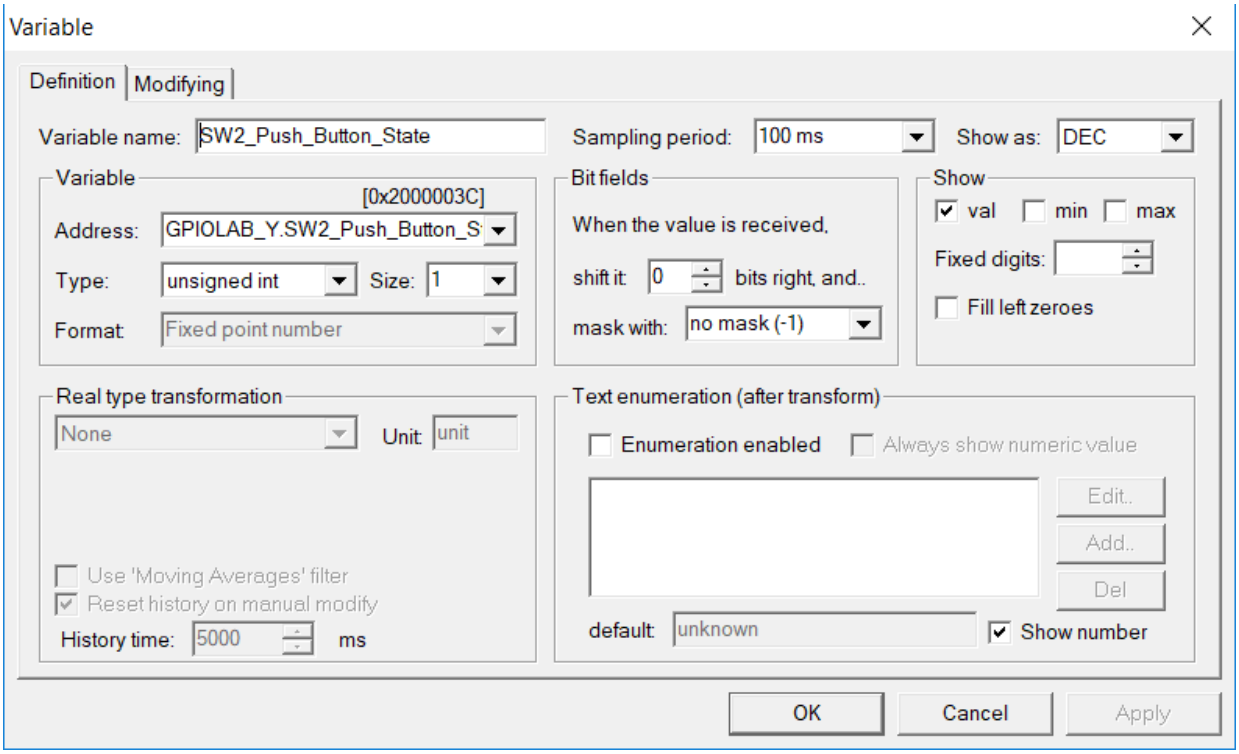


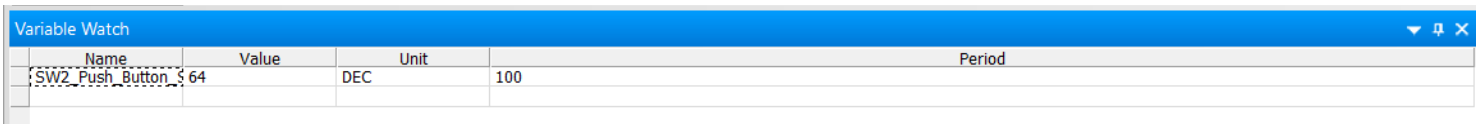
- Click OK and then select “**Tools**” then “**Connection Wizard**” to establish the communication.
- In connection wizard select: - Use Direct connection to on-board USB port, click next.
- Select COM port, Baud rate 115200, Click next. And then Finish.
- Create a new watch variable by double click 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.**



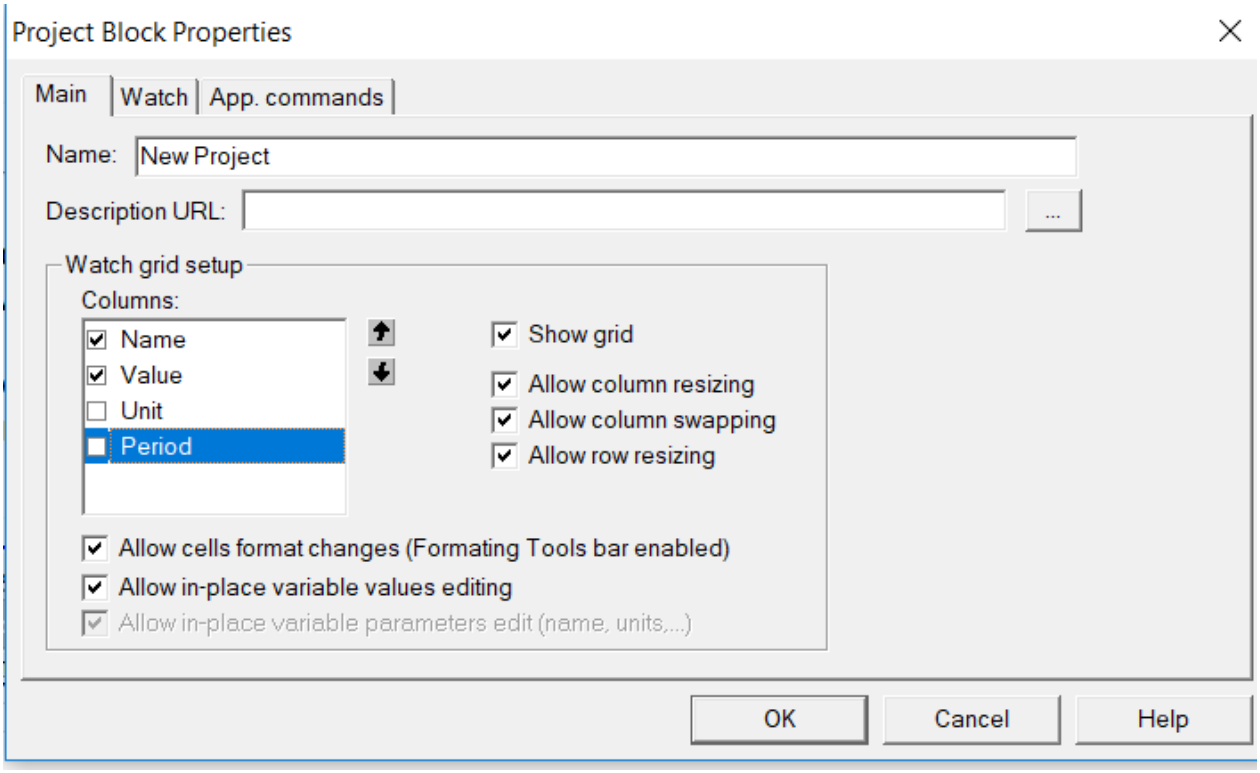


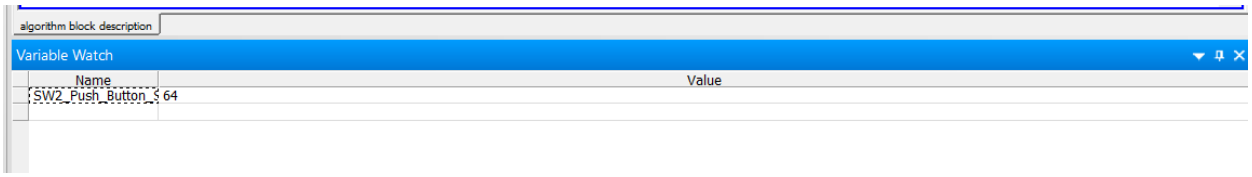
- Select Sampling rate 100ms and Change the variable name to SW2_Push_Button_State.



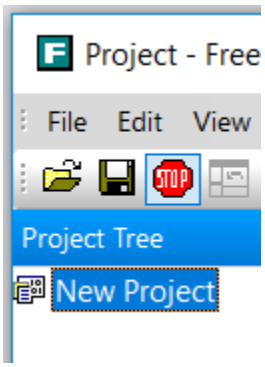


- Right Click on the “SW2_Push_Button_State” variable and select “Watch Properties”. Open **Main** Tab

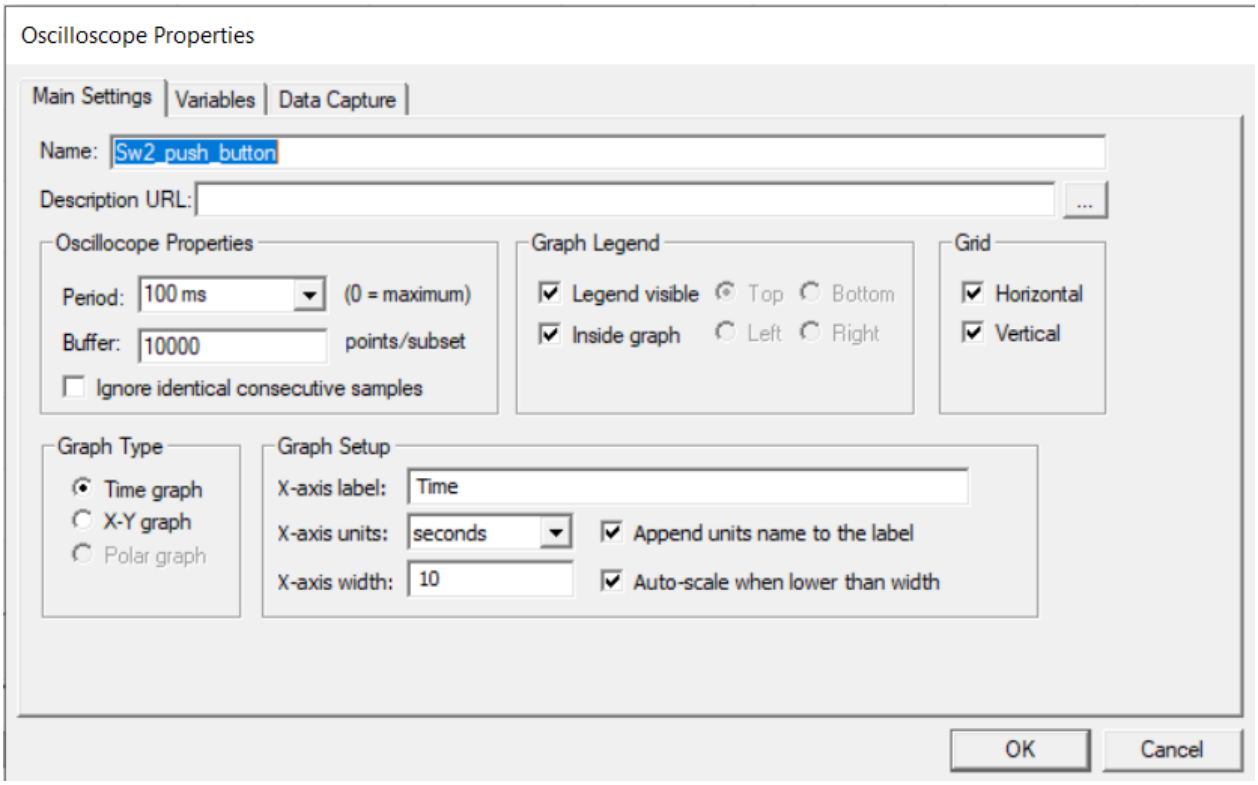




- Save FreeMASTER project as “GPIOLAB”.

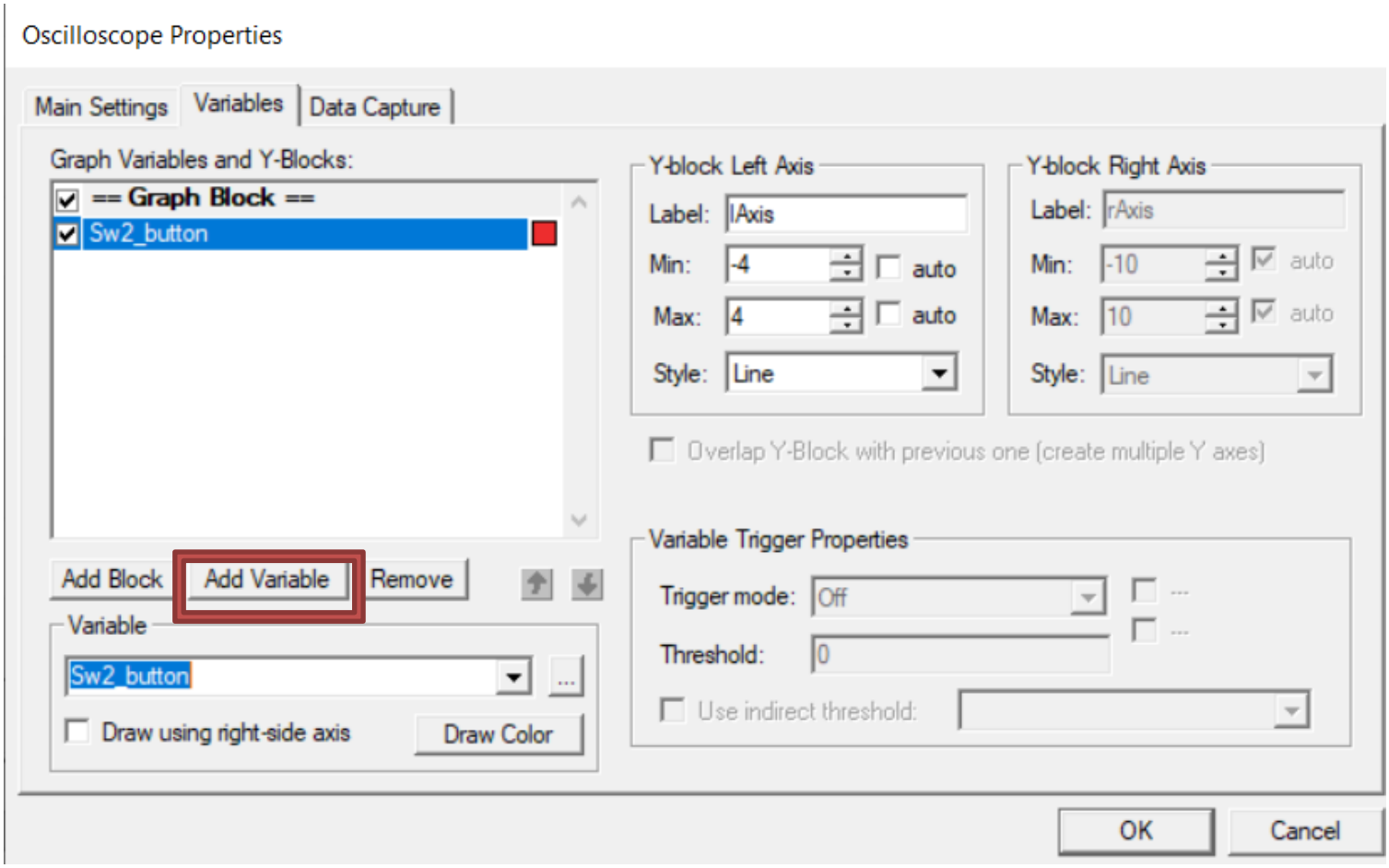


- Right Click on “New Project” and select “Create Oscilloscope”.



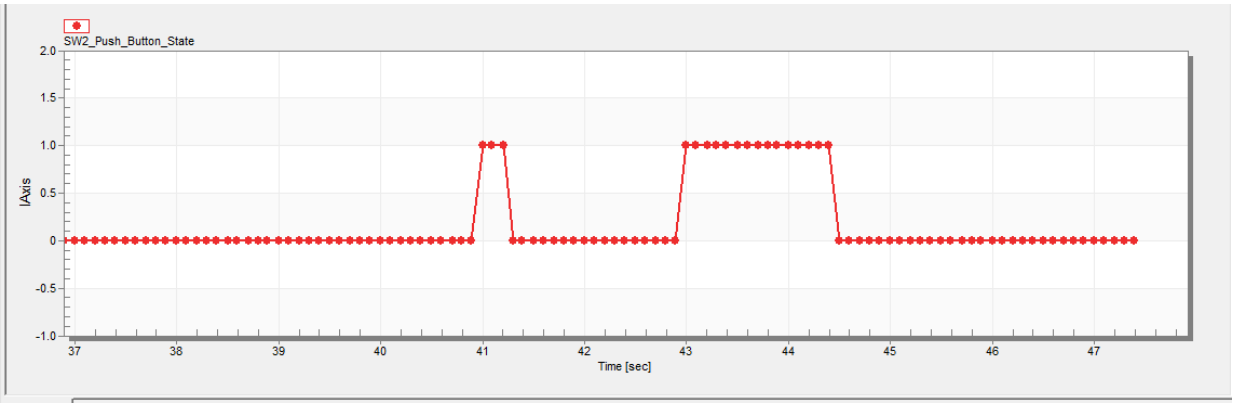
- Set the name as “SW2_Push_Button_State” and the period as 100 ms.

- 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.



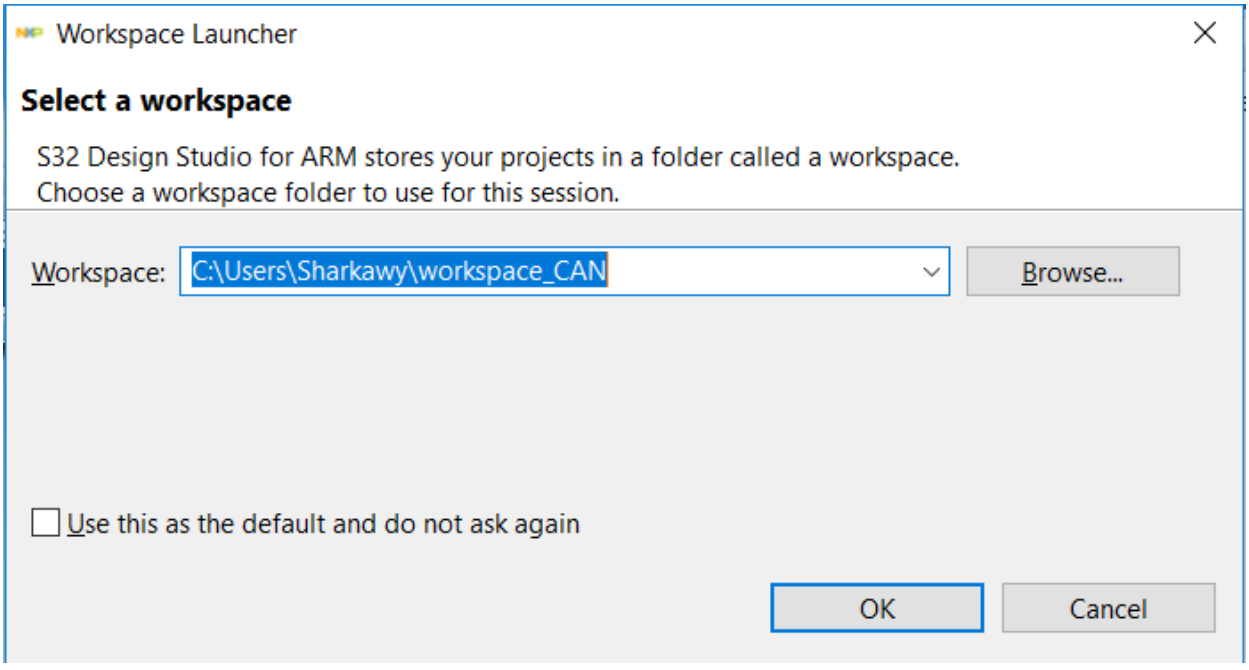
END OF LAB 6
Follows Optional Portions of the LAB.

III. 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.

Impan ☐ X

Import example project

Project

- bootloader_KEA
- FTM_PWT
- hello
- hello clocks
- hello_interrupts
- UART
- S32K144
  - ADC
  - DMA
  - Fle N
  - FlexCAN_F D
  - FTM
  - hello
  - hello_clocks
  - hello_interrupts
  - LPSPI
  - LPUART
- S32K144 EAR SDK Example Projects
  - demO_apps
    - ADC LOW_POWER

Name

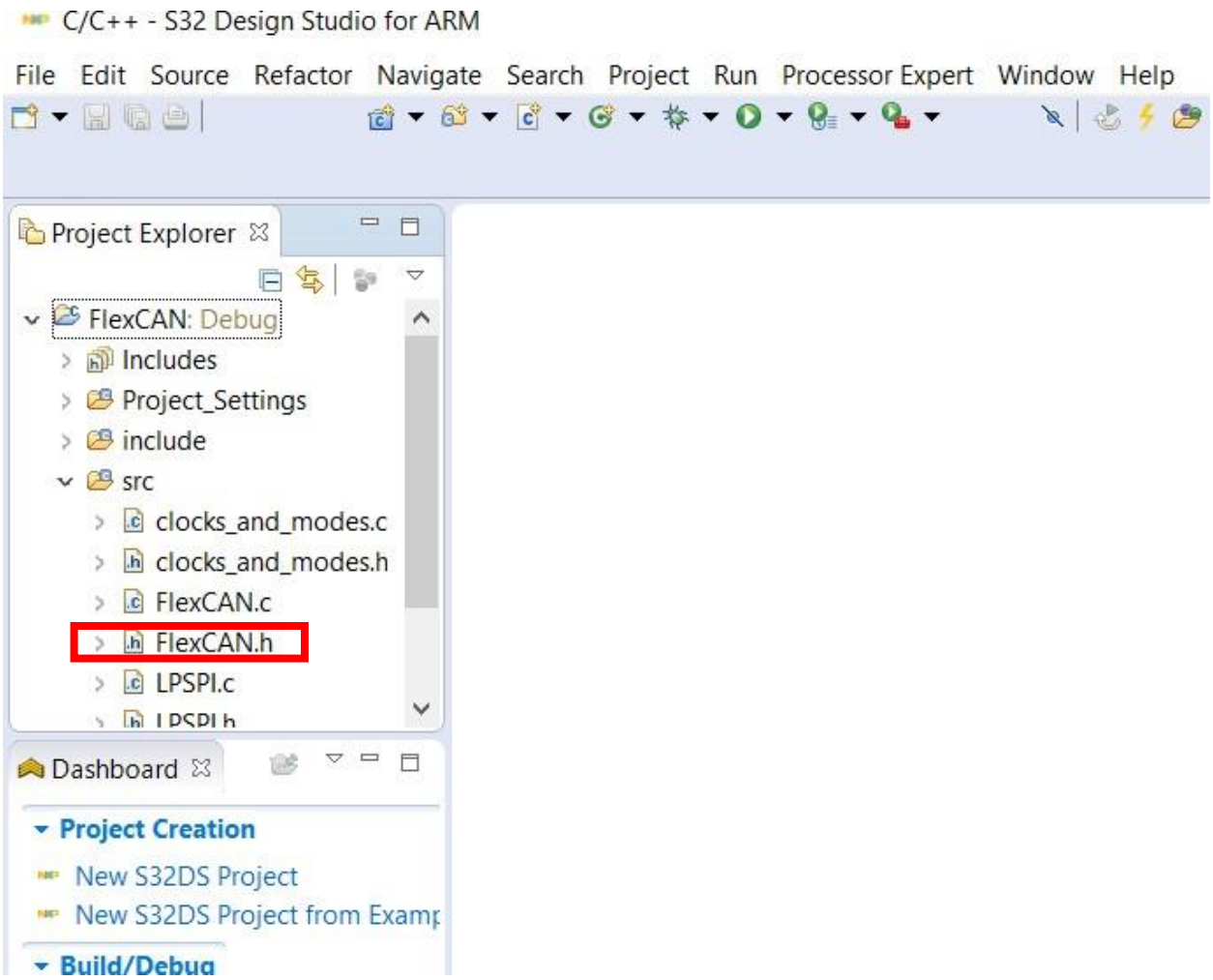
FlexCAN

Description

Simple CAN 2.0 transmit / receive

in|Flish 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.

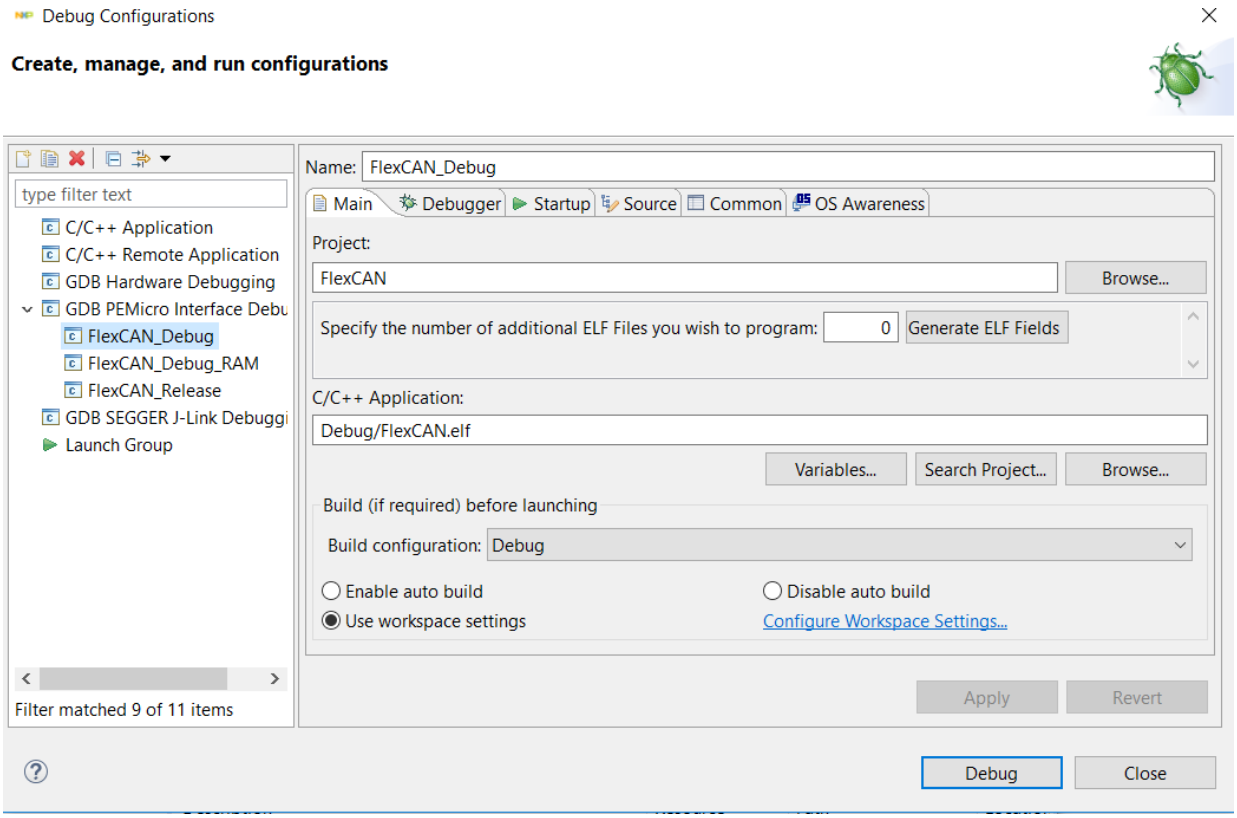
- Connect 12 V power supply to both boards

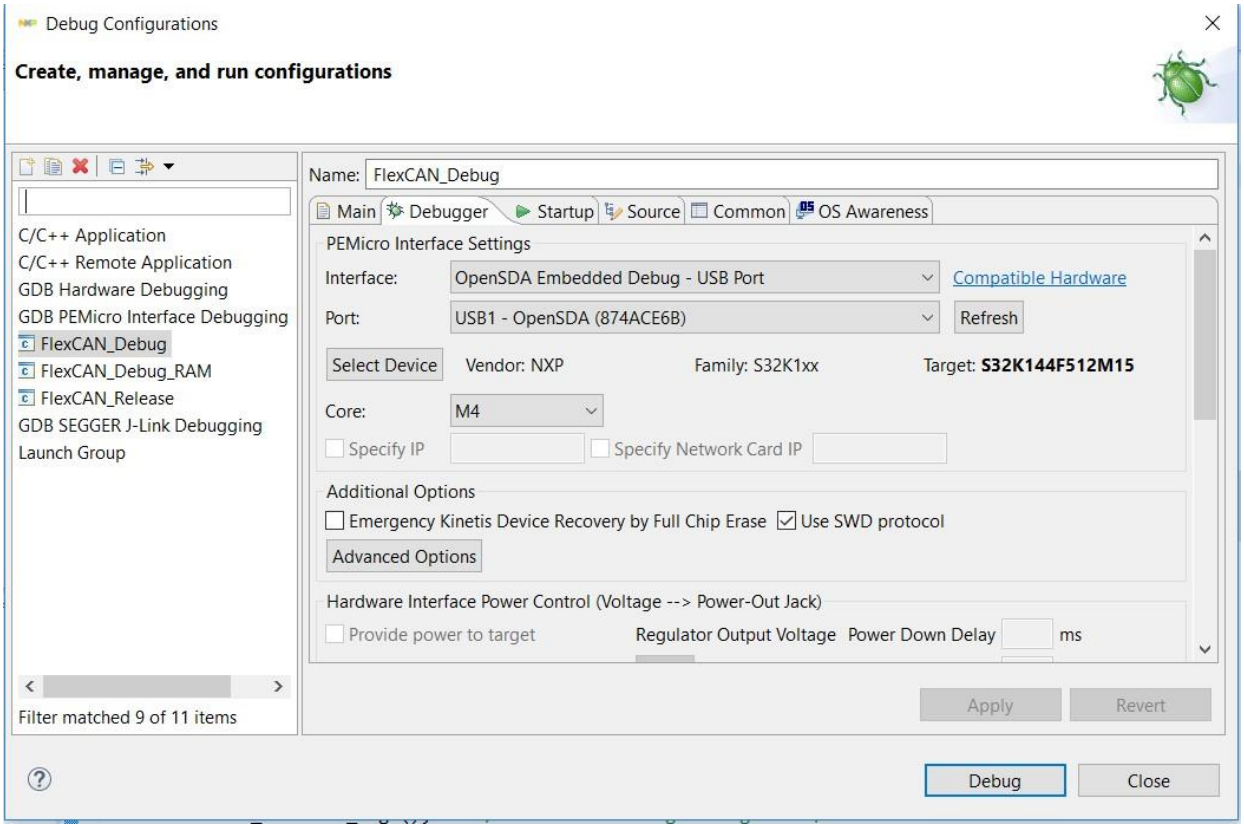
- Configure code for Node B by commenting out the line in FlexCAN.h as shown below:
  //#define Node A.

*FlexCAN.h

```
/* FlexCAN.h                    (c) 2015 Freescale Semicondu

#ifndef FLEXCAN_H_
#define FLEXCAN_H_

//#define NODE_A          /* If using 2 boards as 2 node
#define SBC_MC33903    /* SBC requires SPI init + max 1

void FLEXCAN0_init (void);
void FLEXCAN0_transmit_msg (void);
void FLEXCAN0_receive_msg (void);

#endif /* FLEXCAN_H_ */
```

- 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

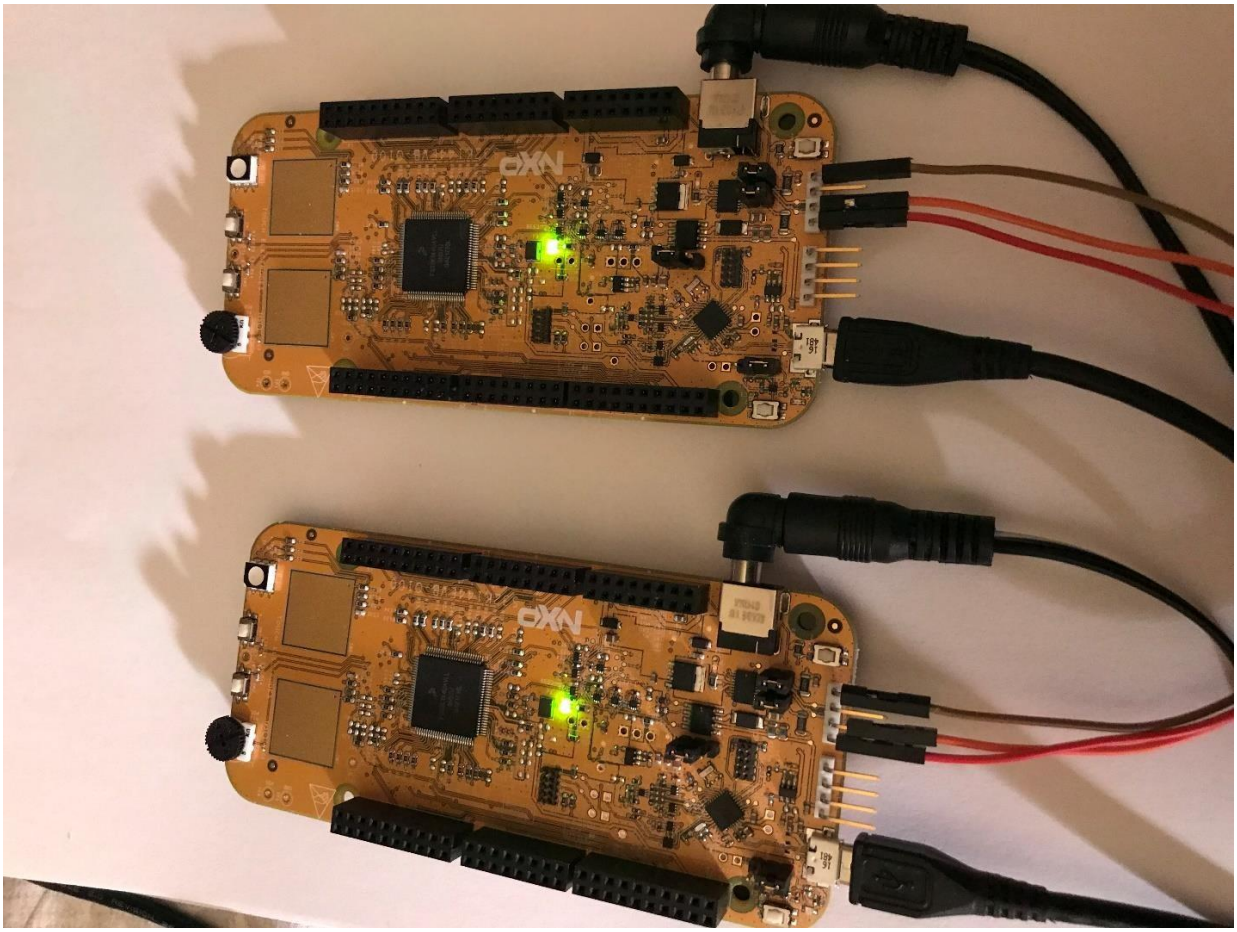
*FlexCAN.h main.c

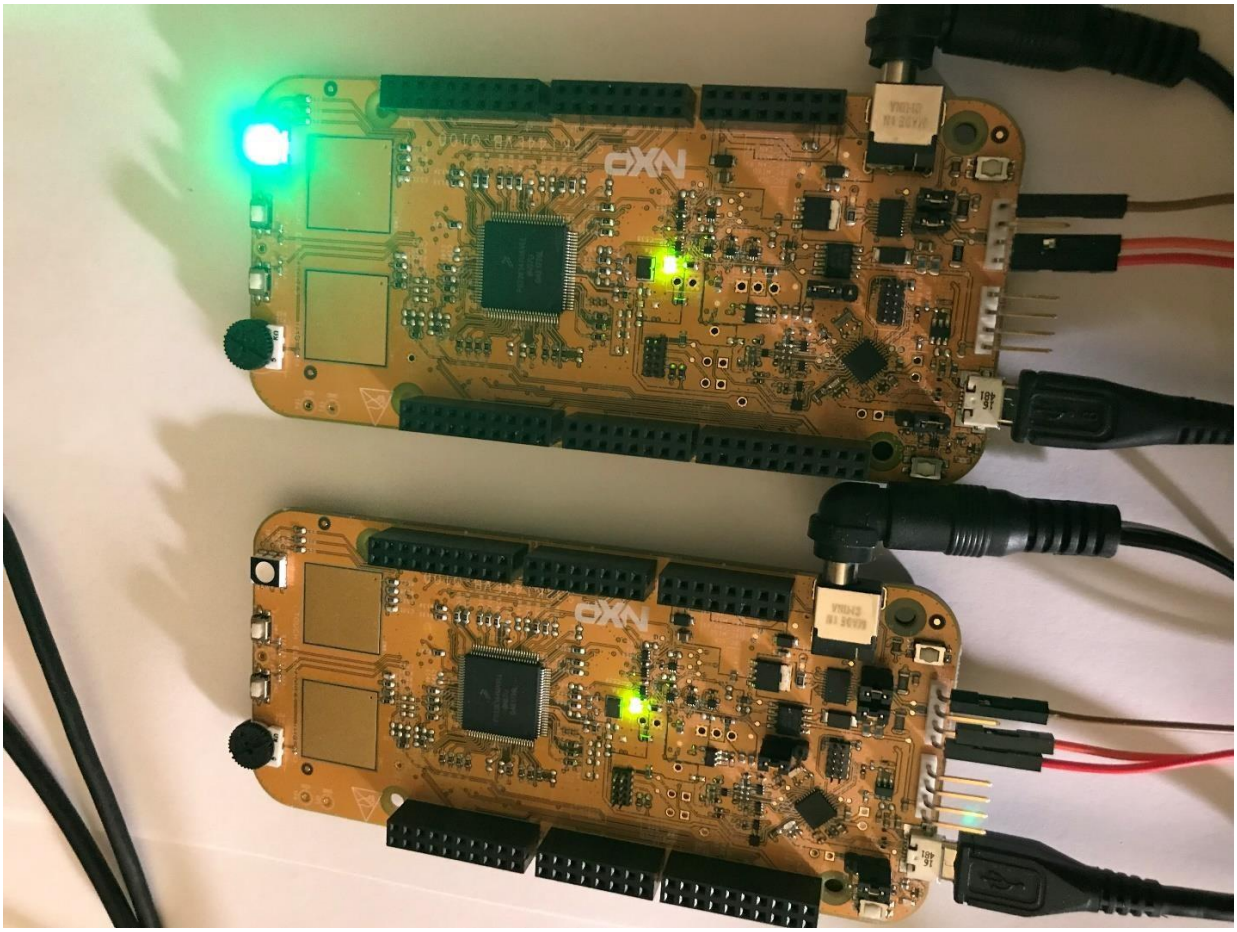
```
/* FlexCAN.h              (c) 2015 Freescale Semiconductor, Inc.

#ifndef FLEXCAN_H_
#define FLEXCAN_H_

#define NODE_A          /* If using 2 boards as 2 nodes, NODE A & B use different CAN IDs */
#define SBC_MC33903     /* SBC requires SPI init + max 1MHz bit rate */

void FLEXCAN0_init (void);
void FLEXCAN0_transmit_msg (void);
void FLEXCAN0_receive_msg (void);

#endif /* FLEXCAN_H_ */
```

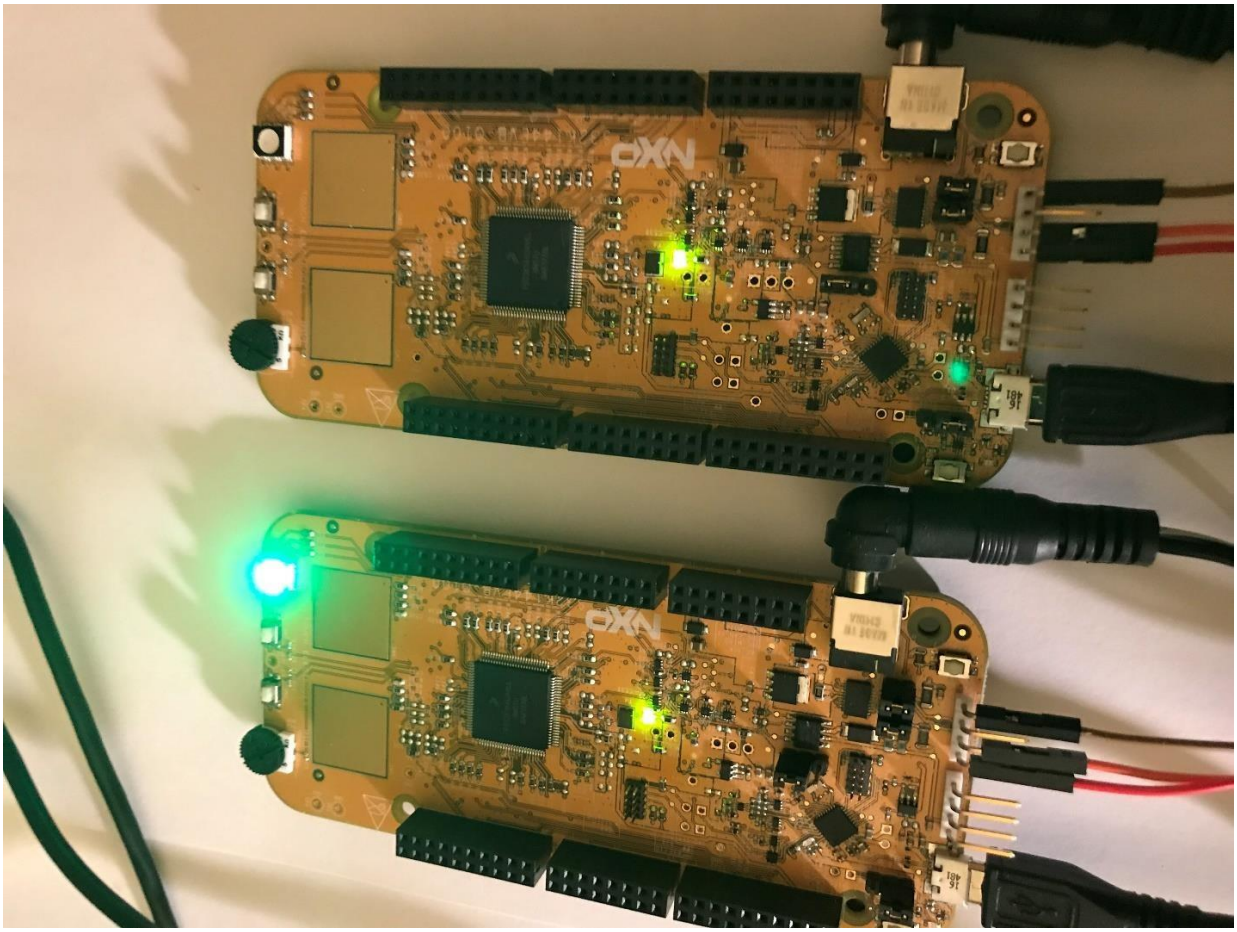
- Same as before, build the program and flash it to the “Node A” EVB.

- 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.



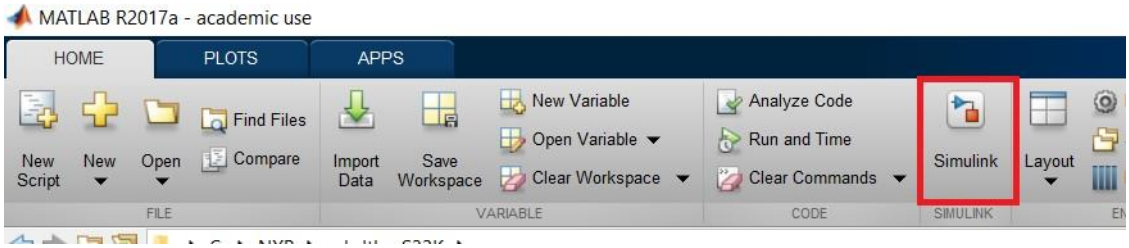




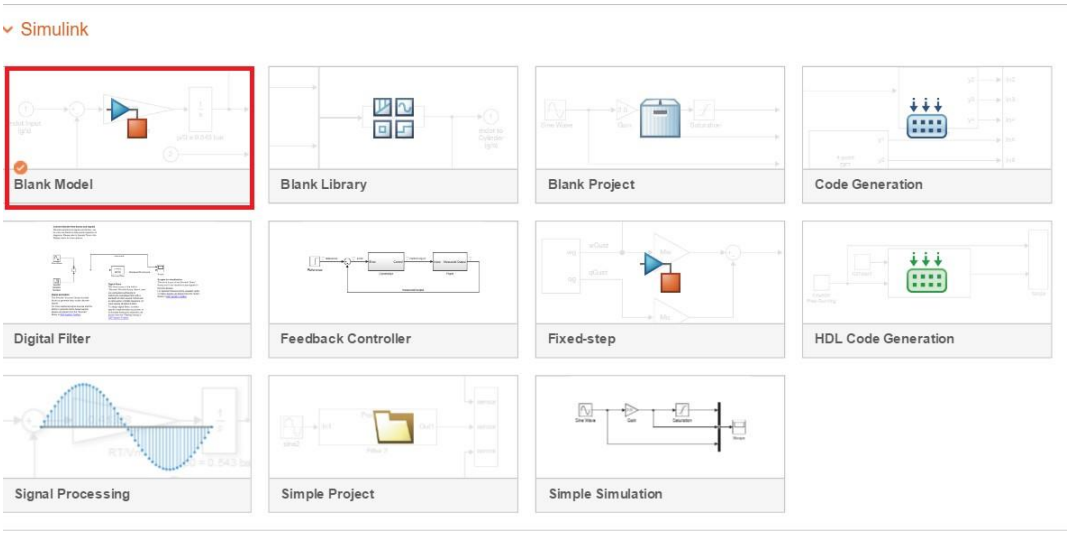
IV. Receiving a Data Pattern over the I2C (Optional part if two boards are available)

- Two S32K144 evaluation boards are needed for this part. The two boards are connected in a Master-Slave configuration using I2C protocol.

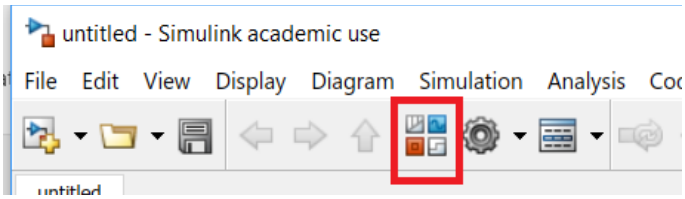
- From MATLAB (R2017a is used here) open Simulink by clicking on the Simulink icon on the top left.



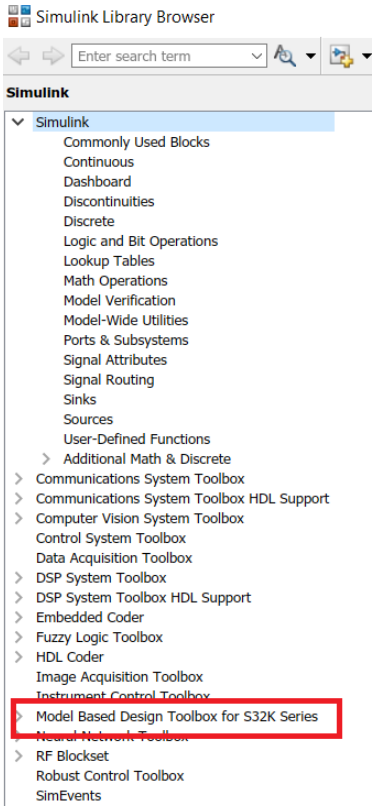
- The Simulink start up page opens. Select the Blank Model option.

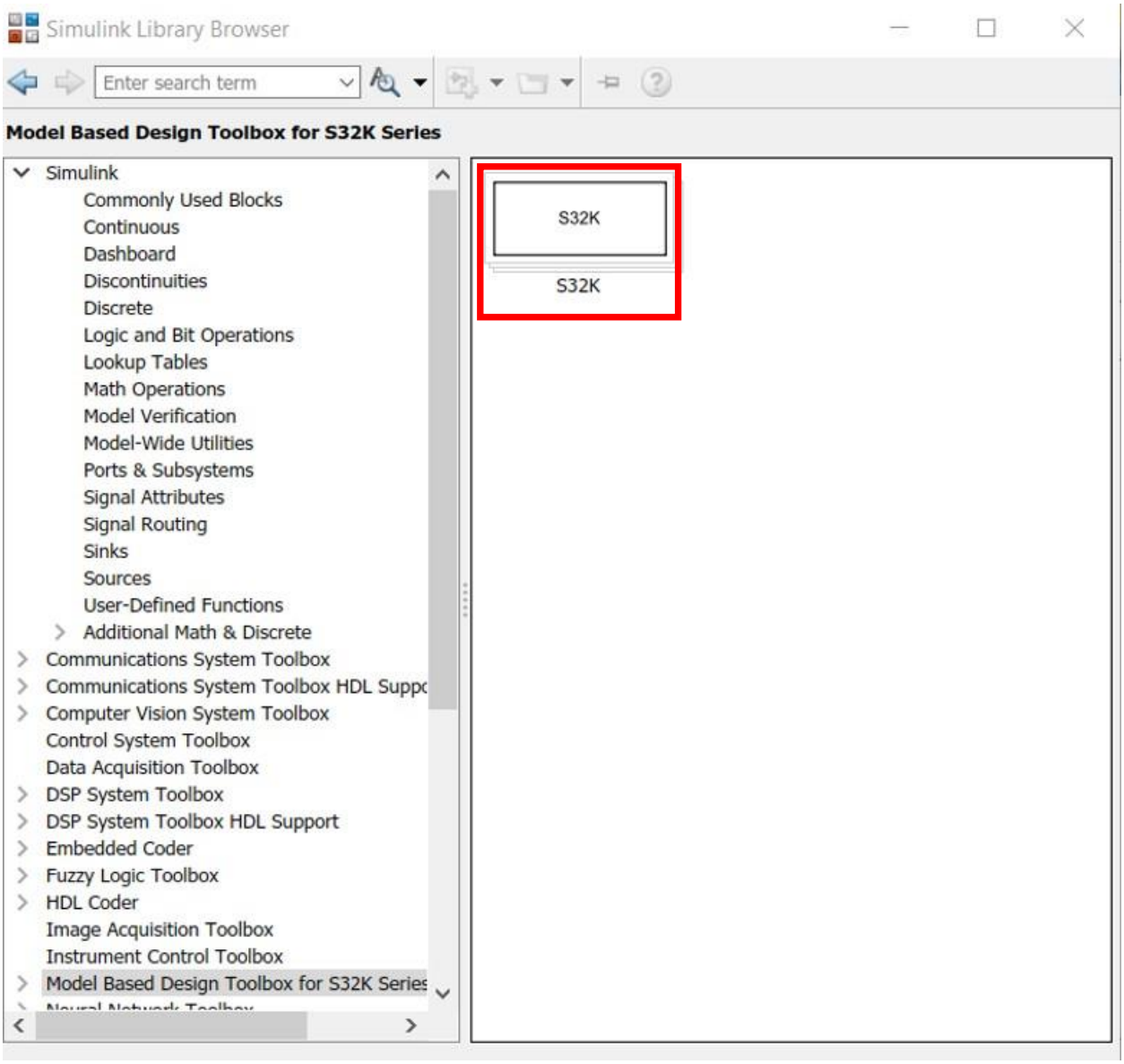


- In the opened blank Model select the Library Browser option.

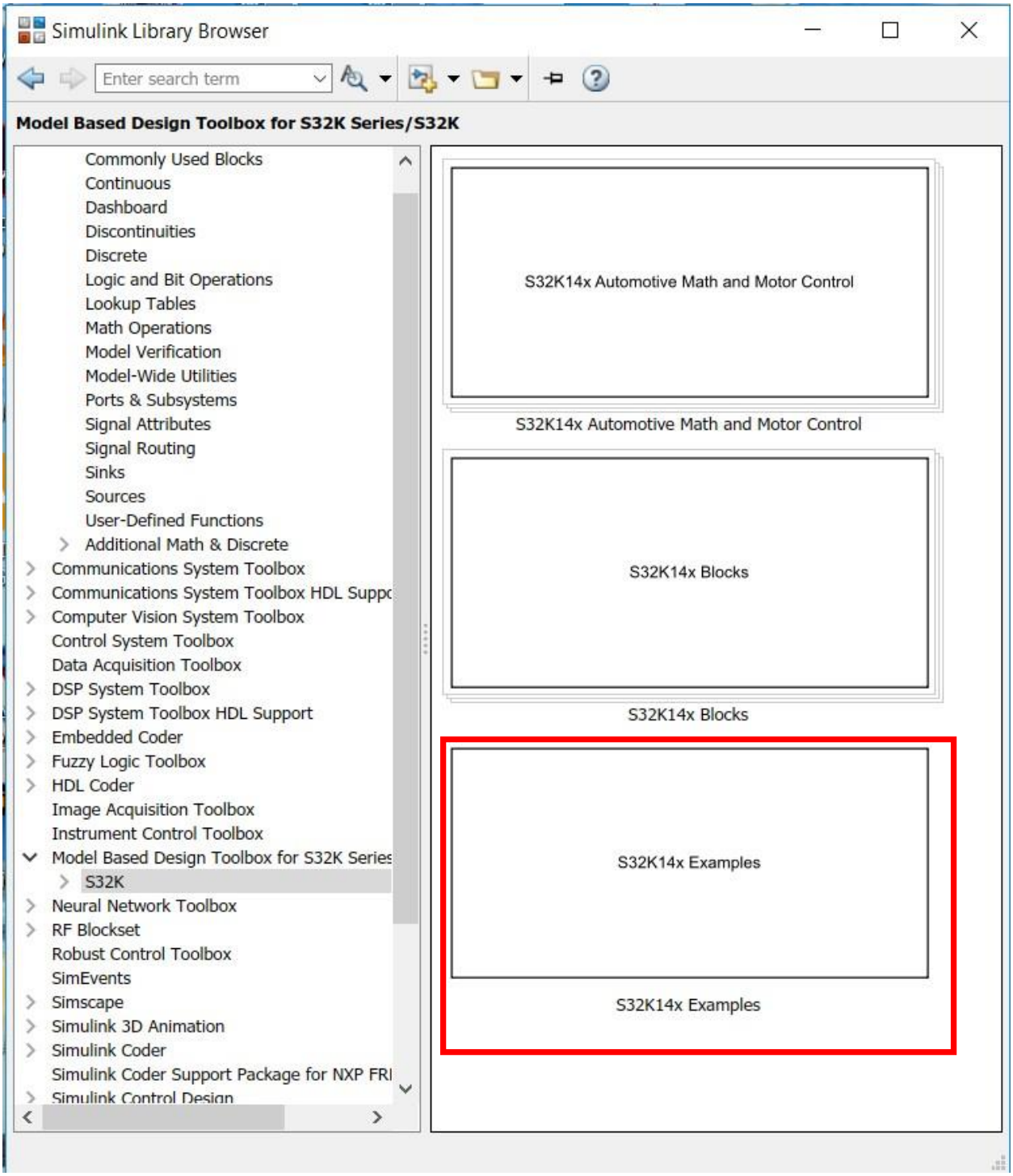


- From the opened list of libraries, select the Model Based Toolbox for S32K Series.



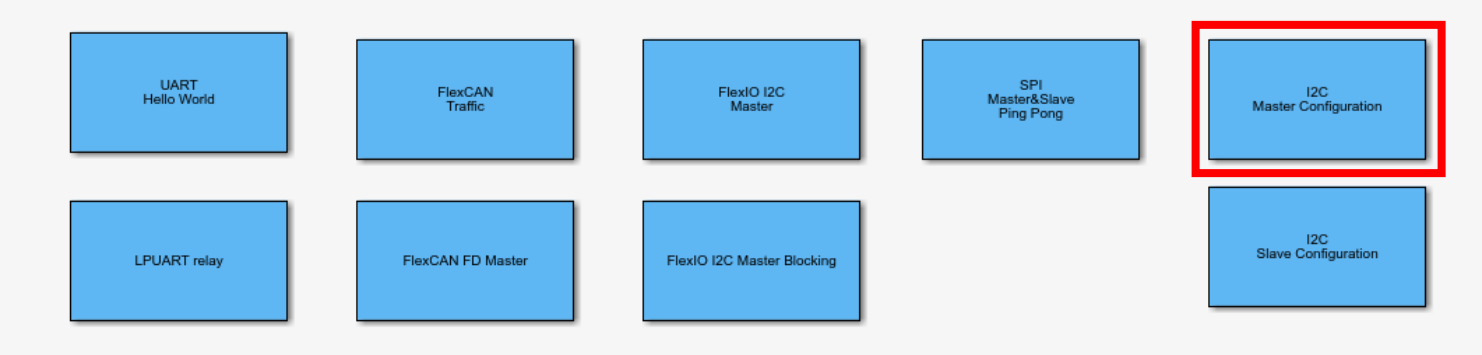


- Expand the Model Based Design Toolbox for S32K series and double click on the Examples option.



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

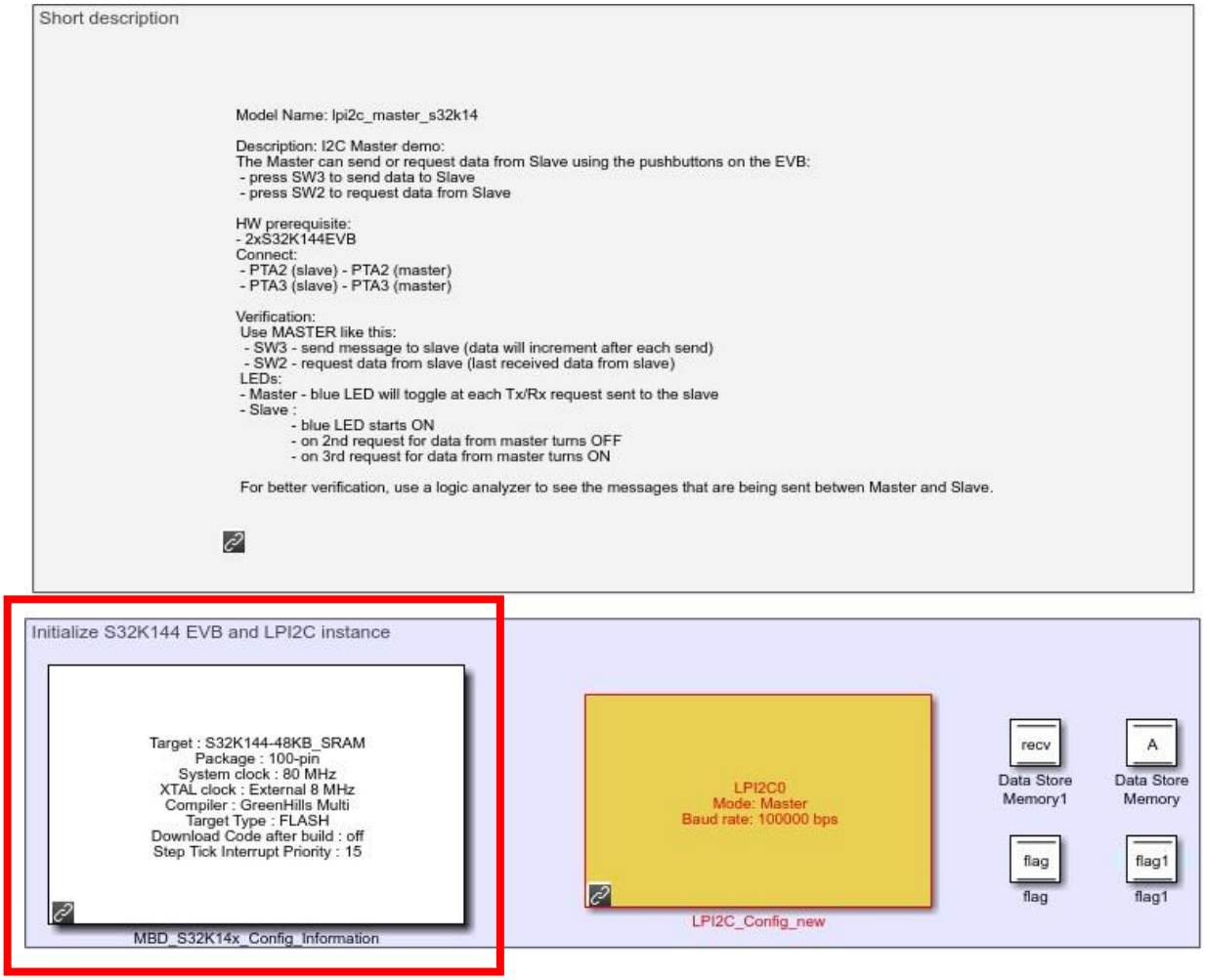
- From the opened Communications window, select the I2C Master Configuration model and build it by selecting the suitable configuration settings.



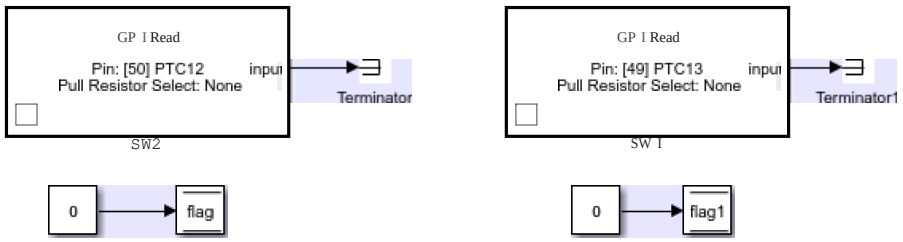
- The Simulink lpi2c_master_s32K14 model will open. Follow the next steps to run the example.

- Be sure that the S32K144 board is connected to the PC using the USB cable.

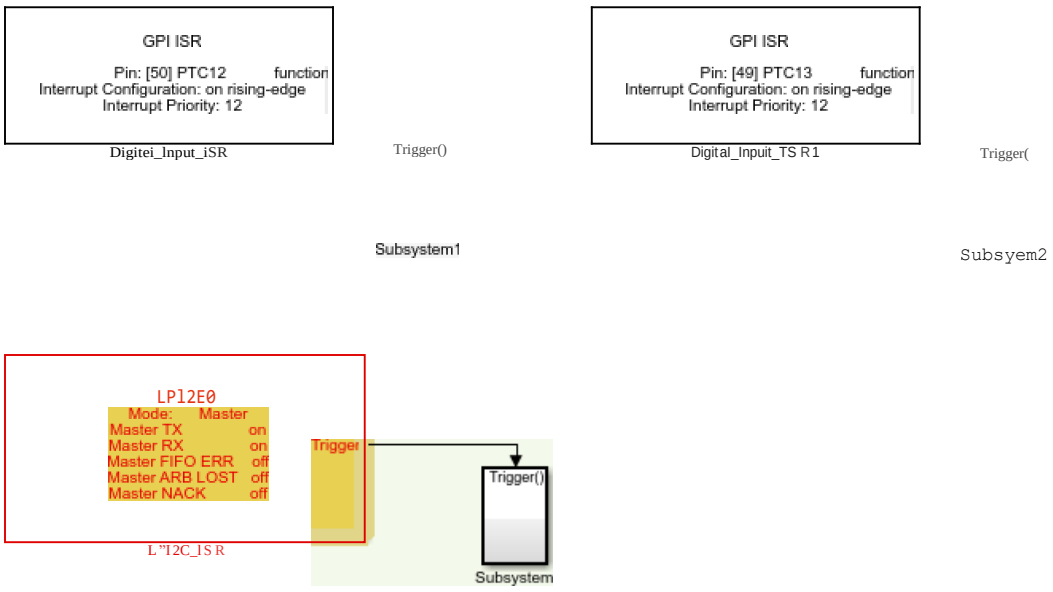
- Double click on MBD_S32K14x_Config_Information block and setup the Download Config parameters according with your PC and HW setup.



Data initialization

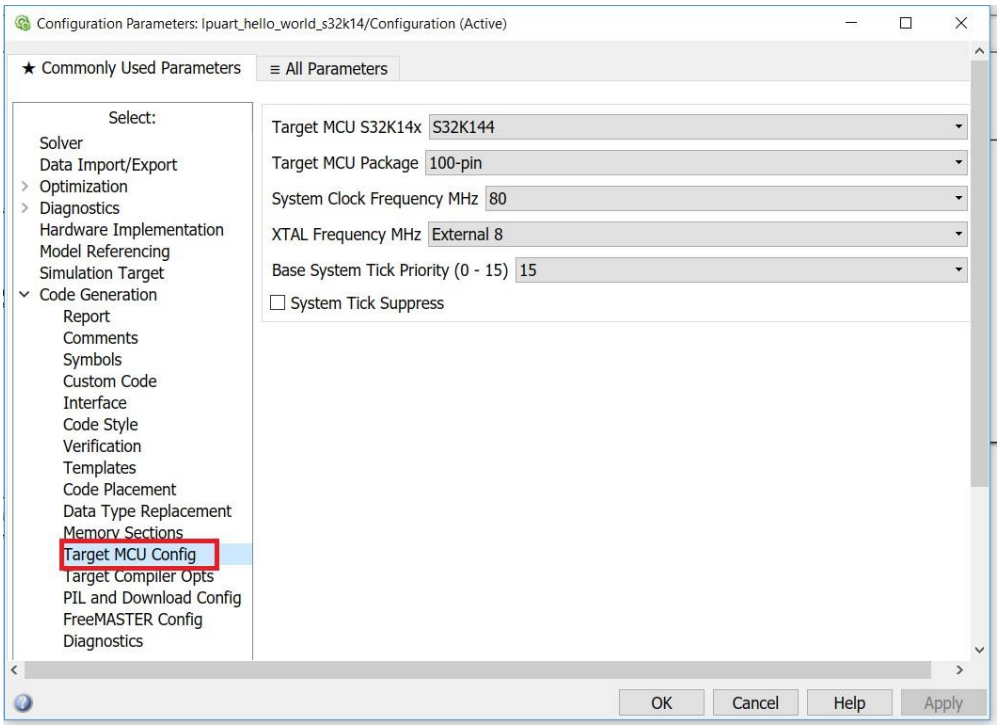


On SW-i press SEND data to Slave
On SW2 press REQUEST DATA from Slave

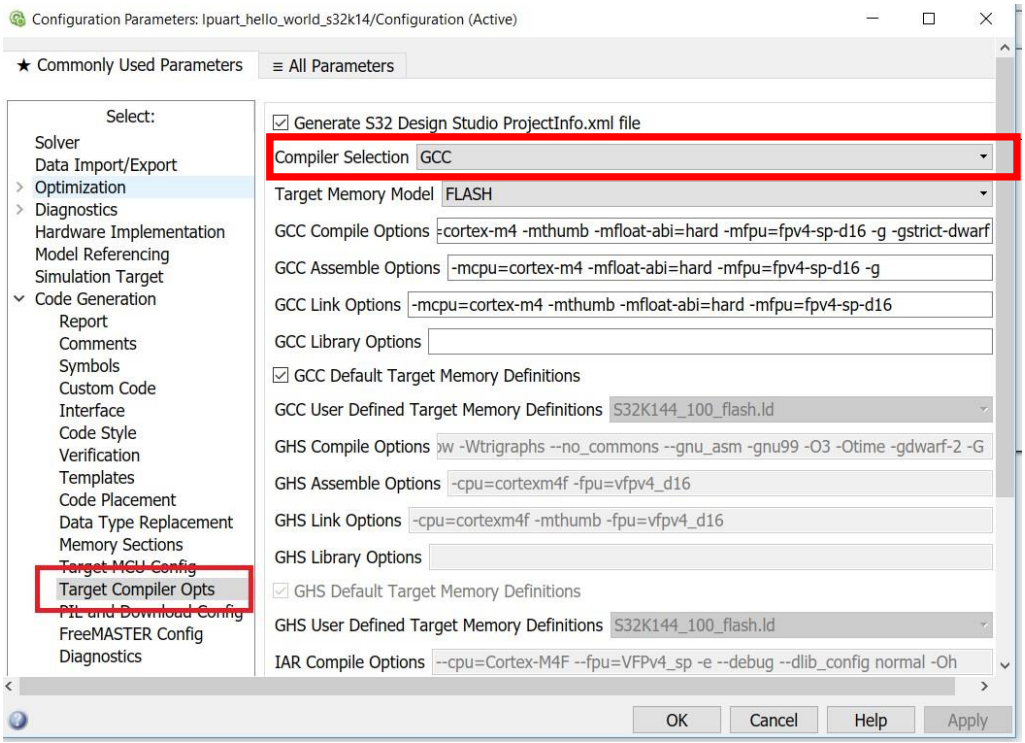


Toggle blue LE D at TX and 'X evant

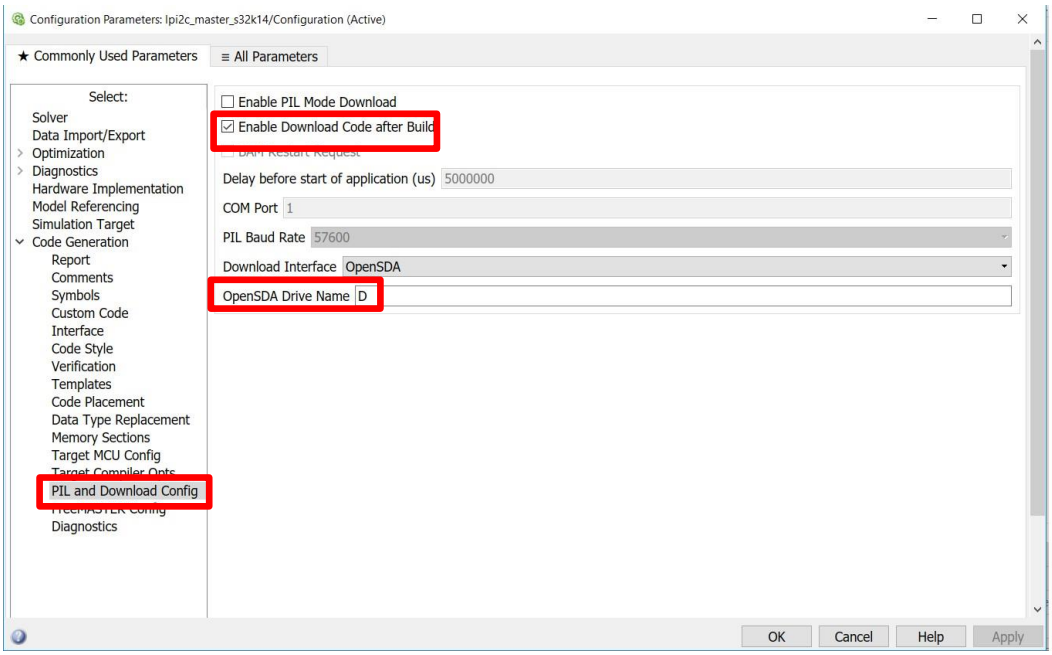
Target MCU Config:



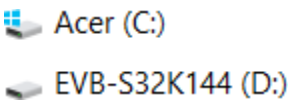
Target Compiler Opts:



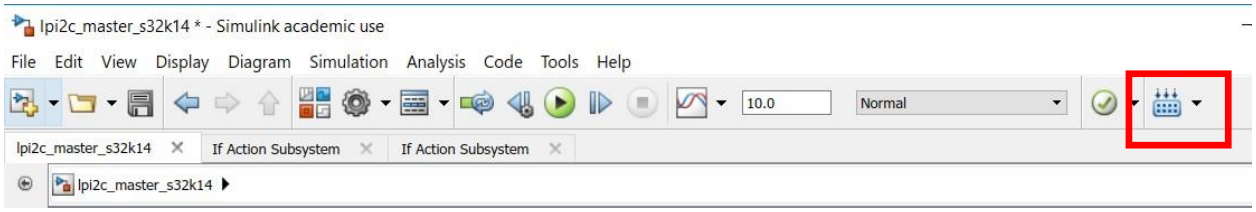
PIL and Download Config:



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



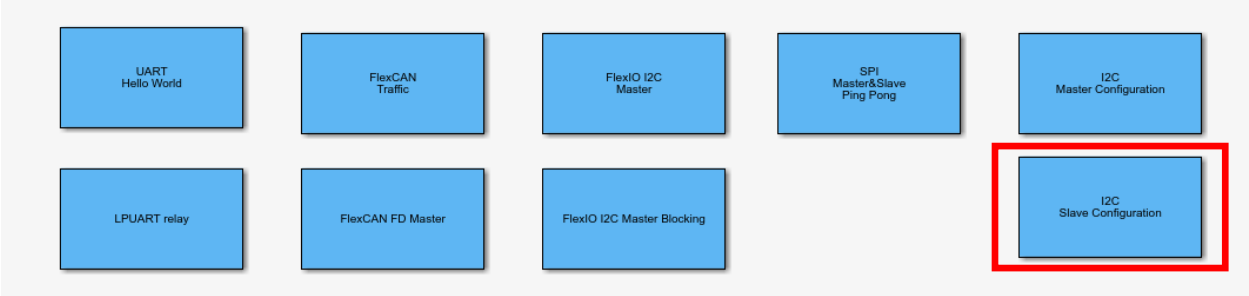
- Apply and close this window. Press Build Model button and wait until the code is generated, compiled and downloaded to the evaluation board.



- 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.

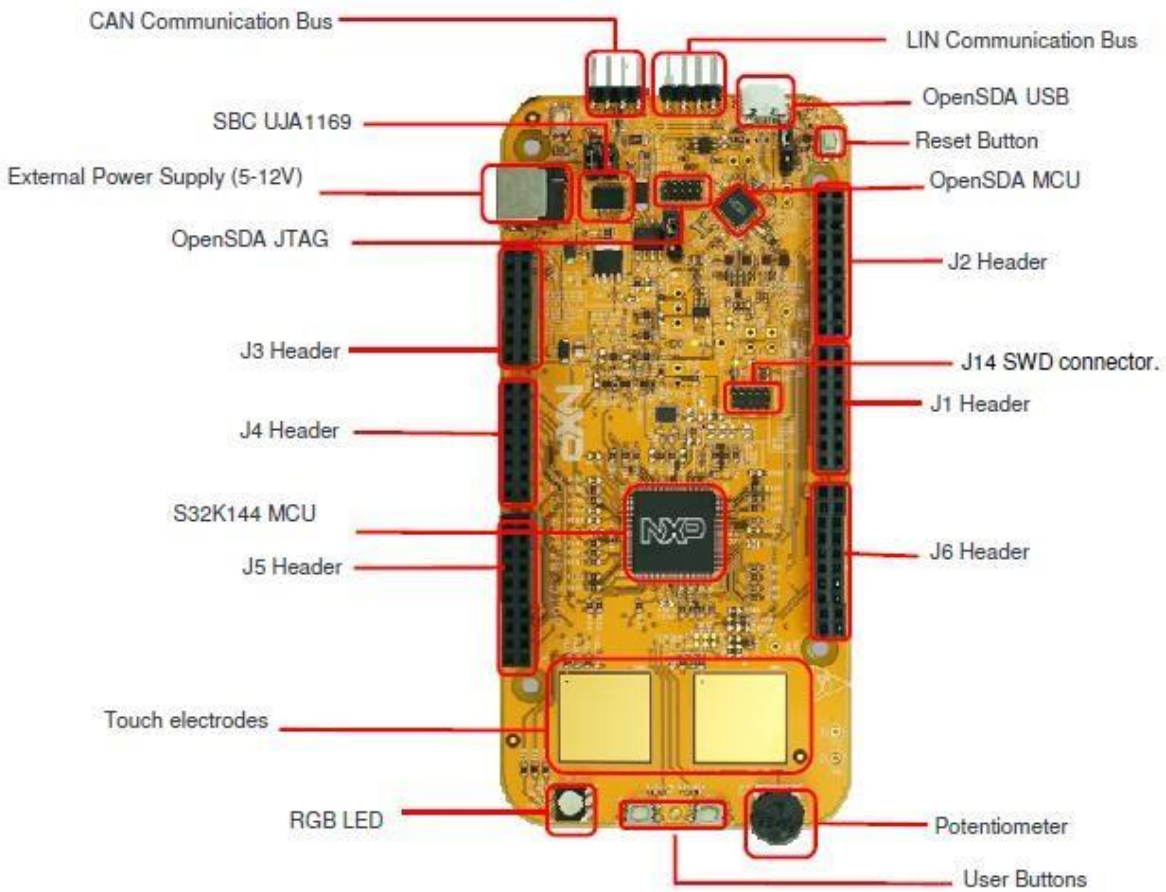
- Follow the same steps as before and select the I2C Slave Configuration model.



- 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.



Header/Pinout Mapping for S32K144

J3

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J3-02	PTB6*	GPIO		J3-01		VIN
J3-04	PTB7*	GPIO		J3-03		IOREF
J3-06	PTE0	GPIO		J3-05	PTA5	RESET
J3-08	PTE9	GPIO		J3-07		3V3
J3-10	PTC5	GPIO		J3-09		5V
J3-12	PTC4	GPIO		J3-11		GND
J3-14	PTA10	GPIO		J3-13		GND
J3-16	PTA4	GPIO		J3-15		VIN

J4

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J4-02	PTC7	GPIO		J4-01	PTD4	ADC0
J4-04	PTC6	GPIO		J4-03	PTB12	ADC1
J4-06	PTB17	GPIO		J4-05	PTB0	ADC2
J4-08	PTB14	GPIO		J4-07	PTB1	ADC3
J4-10	PTB15	GPIO		J4-09	PTA6/PTE11/PTA2	ADC4
J4-12	PTB16	GPIO		J4-11	PTC0/PTE10/PTA3	ADC5
J4-14	PTC14	GPIO		J4-13	PTE2	ADC6
J4-16	PTC3	GPIO		J4-15	PTE6	ADC7

J5

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J5-02	PTE16	GPIO		J5-01	PTA15/PTD11	ADC8
J5-04	PTE15	GPIO		J5-03	PTA16/PTD10	ADC9
J5-06	PTE14	GPIO		J5-05	PTA1	ADC10
J5-08	PTE13	GPIO		J5-07	PTA0	ADC11
J5-10		VDD		J5-09	PTA7	ADC12
J5-12		GND		J5-11	PTB13	ADC13
J5-14	PTE1	GPIO		J5-13	PTC1	ADC14
J5-16	PTD7	GPIO		J5-15	PTC2	ADC15
J5-18	PTD6	GPIO		J5-17	NC	GPIO
J5-20	PTC15	GPIO		J5-19	NC	N/A

J2

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J2-19	PTE10/PTA3	D15/I2C_CLK		J2-20	NC	GPIO
J2-17	PTE11/PTA2	D14/I2C_SDA		J2-18	NC	GPIO
J2-15		ANALOGUE REF		J2-16	PTA14	GPIO
J2-13		GND		J2-14	PTE7	GPIO
J2-11	PTB2	D13/SPI_SCK		J2-12	PTC13	GPIO
J2-09	PTB3	D12/SPI_SIN		J2-10	PTC12	GPIO
J2-07	PTB4	D11/SPI_SOUT		J2-08	PTE8	GPIO
J2-05	PTB5	D10/SPI_CS		J2-06	PTD0	GPIO
J2-03	PTD14	D9/PWM		J2-04	PTD16	GPIO
J2-01	PTD13	D8/PWM		J2-02	PTD15	GPIO

J1

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J1-15	PTC11/PTE8	D7		J1-16	PTE3	GPIO
J1-13	PTC10/PTC3	D6		J1-14	PTD3	GPIO
J1-11	PTB11	D5		J1-12	PTD5	GPIO
J1-09	PTB10	D4		J1-10	PTD12	GPIO
J1-07	PTB9	D3		J1-08	PTD11	GPIO
J1-05	PTB8	D2		J1-06	PTD10	GPIO
J1-03	PTA3	D1		J1-04	PTA17	GPIO
J1-01	PTA2	D0		J1-02	PTA11	GPIO

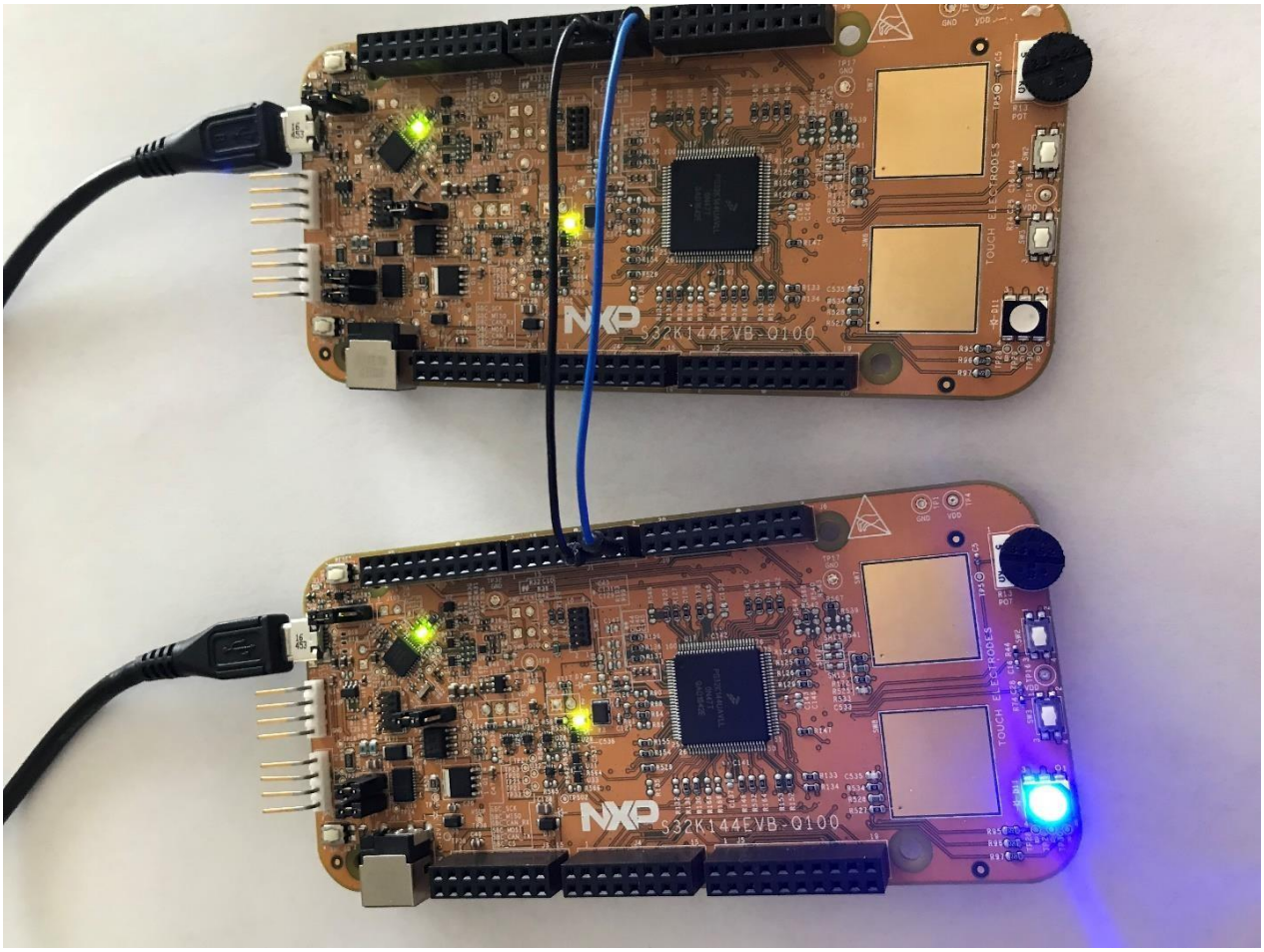
J6

PIN	PORT	FUNCTION		PIN	PORT	FUNCTION
J6-19	PTA9	D14		J6-20	PTE4	GPIO
J6-17	PTA8	D15		J6-18	PTE5	GPIO
J6-15	PTE12	D16		J6-16	PTA12	GPIO
J6-13	PTD17	D17		J6-14	PTA13	GPIO
J6-11	PTC9	D18		J6-12		GND
J6-09	PTC8	D19		J6-10		VDD
J6-07	PTD8	D20		J6-08	PTC16	GPIO
J6-05	PTD9	D21		J6-06	PTC17	GPIO
J6-03	PTD2	GPIO		J6-04	PTD3	GPIO
J6-01	PTD0	GPIO		J6-02	PTD1	GPIO

Arduino compatible pins
NXP pins

*0ohm resistor is not connected

- 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.