Blinking LED Tutorial for the Hello FPGA Kit

Written by: Dan Grise

2020-11-16

Purpose:

To provide a simple tutorial to help the user generate a design targeting the Hello FPGA board.

This tutorial is good for people with little to no experience with the Libero SoC software.

Assumptions:

To run this tutorial the following software must be installed:

Libero SoC (Version 12.0 or later is recommended)

Soft Console (Version 4.0 or later is recommended)

Hello FPGA GUI

The user has access to the Hello FPGA development board

Disclaimer:

The author is not affiliated with Microsemi or Microchip and is not an expert with Libero SoC.

This document was written to help users get over some of the learning curve hurdles associated with trying out new development platforms.

And yes, I acknowledge that the formatting of this document is sparse and bare bones. All the same I hope it is helpful to those that read it.

Setting up Project

Open the Libero SoC software

Click Project → New Project

Enter first_processor as the Project name

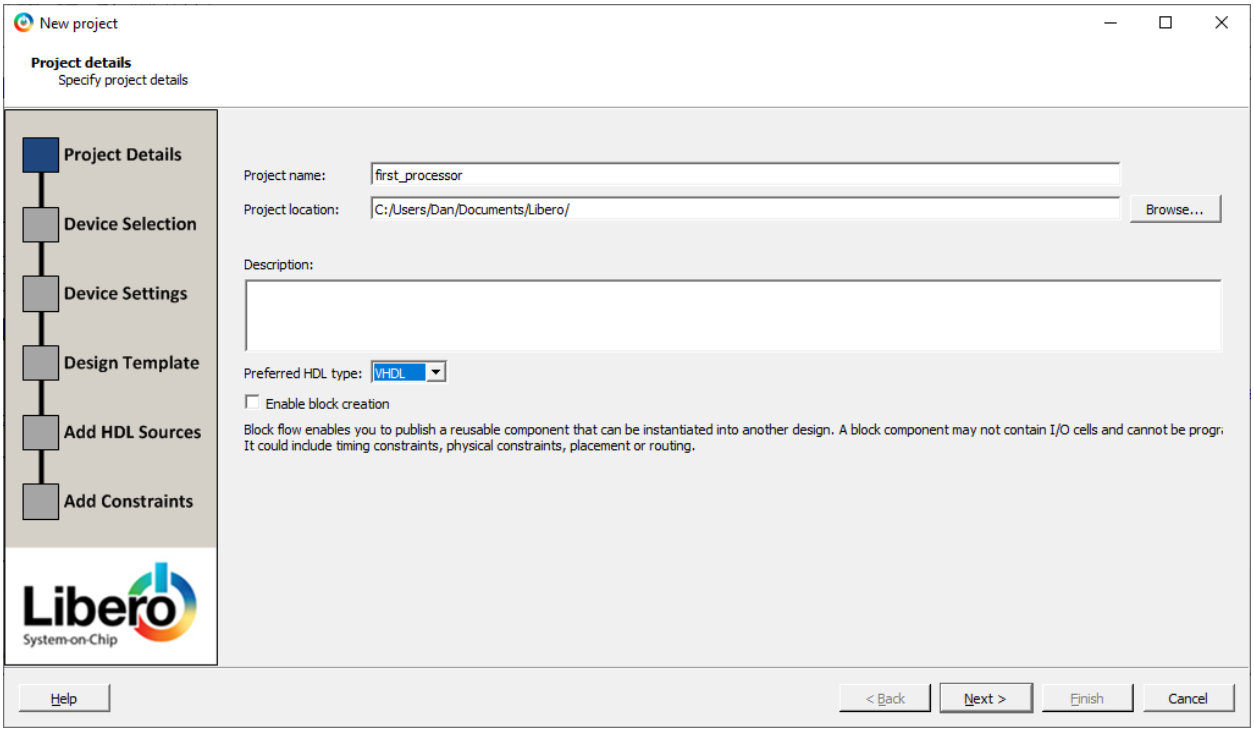
Choose a suitable folder for Project location

You can leave Description blank

Choose your preferred HDL language

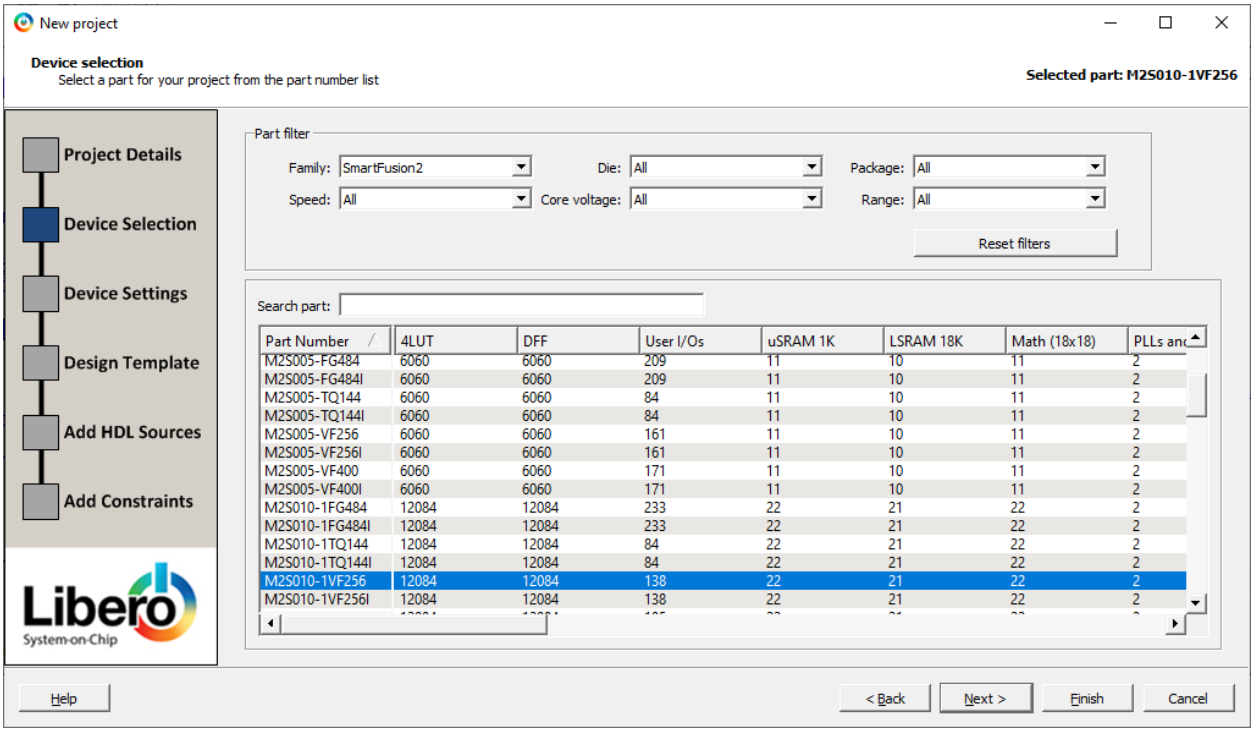
Leave Enable block creation blank

Click Next



Select M2S010-1VF256 as your device

Click Next



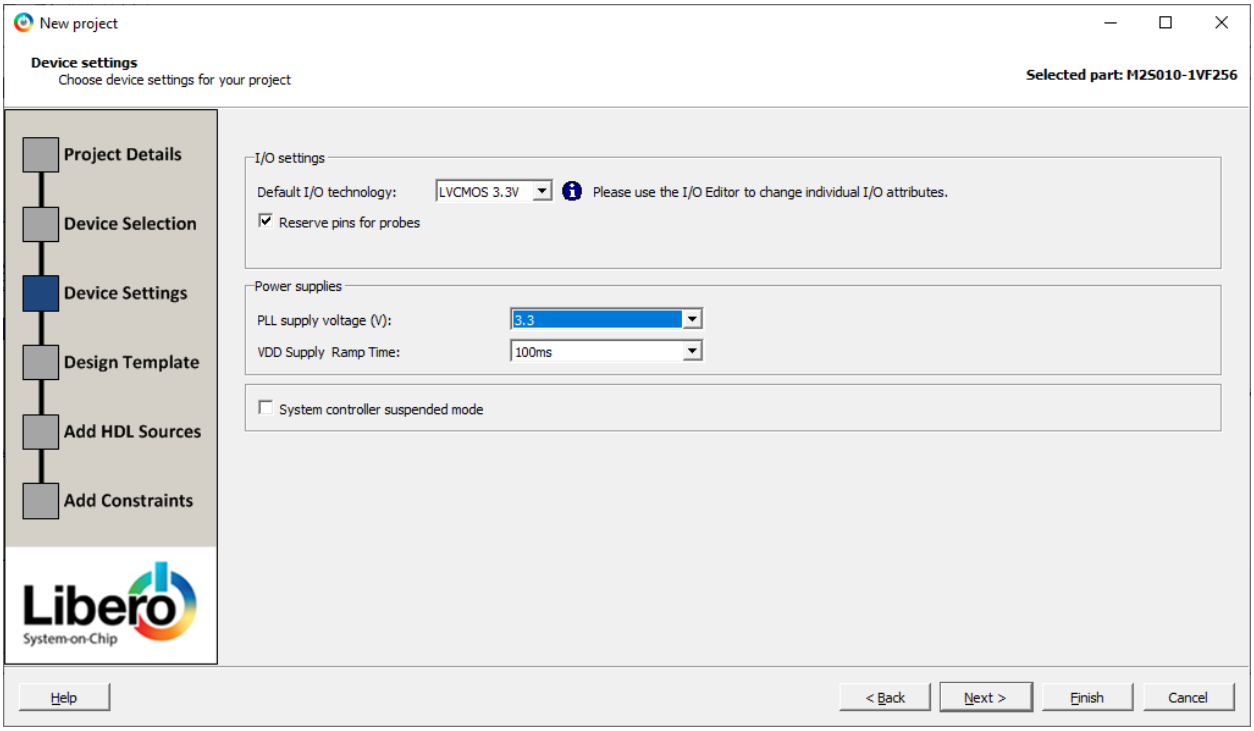
For Default I/O technology select LVMOS 3.3V

Check Reserve pins for probes

Select 3.3 for PLL supply voltage

Leave System controller suspended mode unchecked

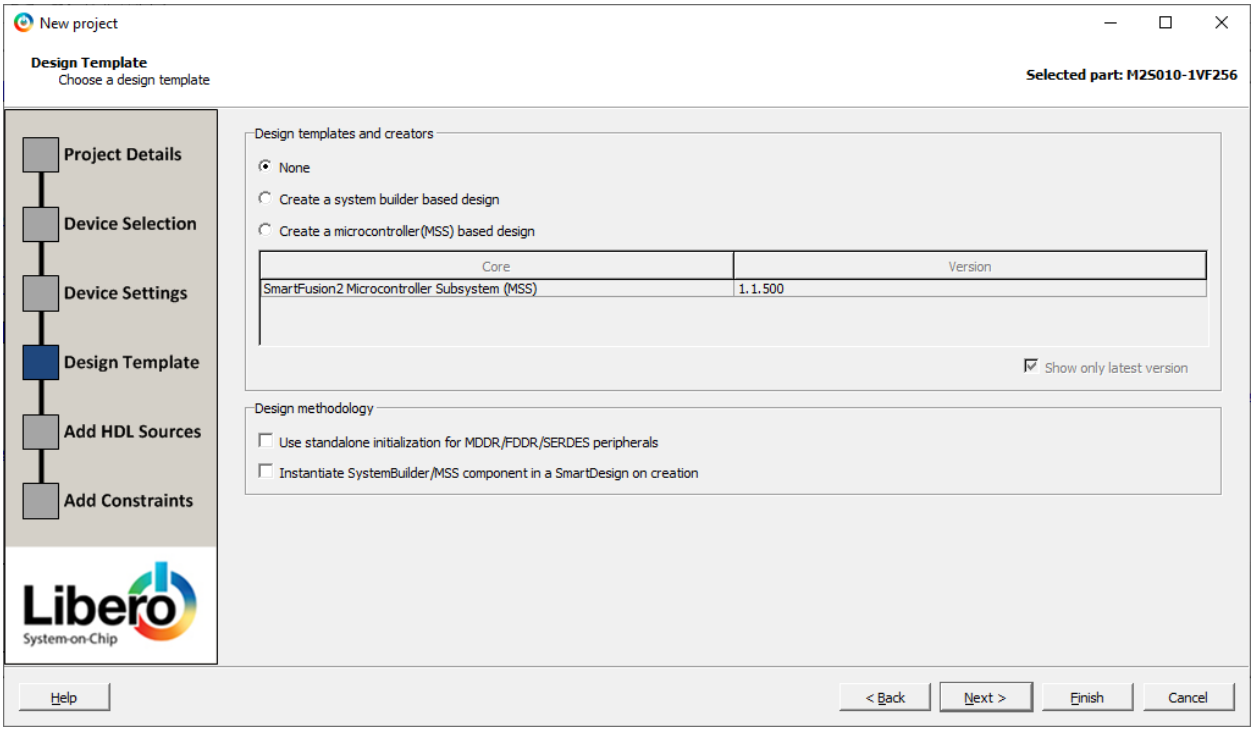
Click Next



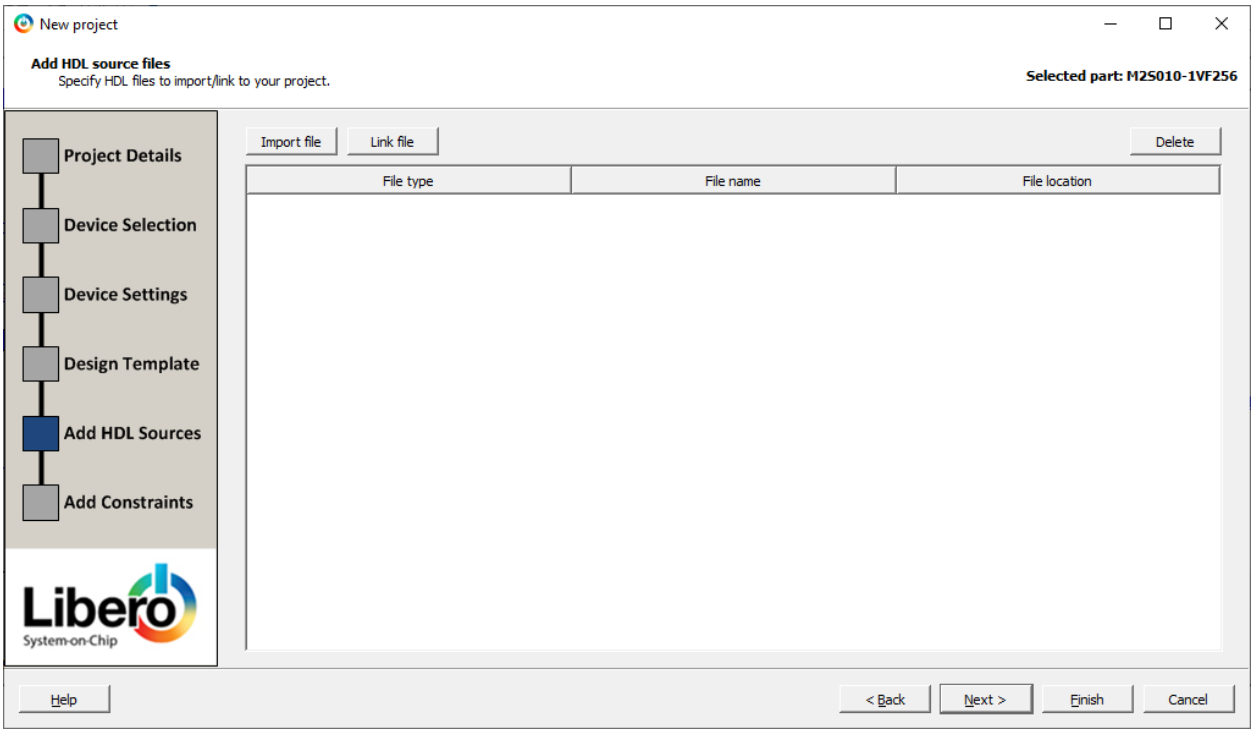
For Design templates and creators select None

Leave other boxes unchecked

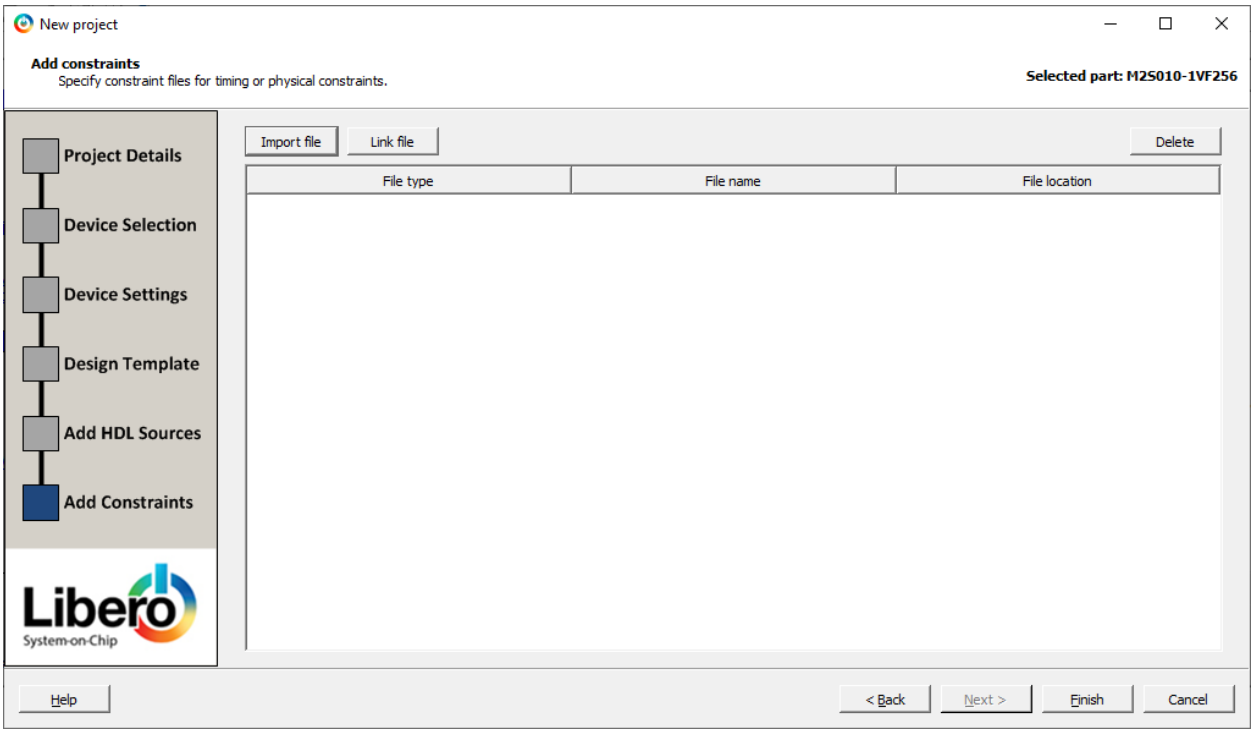
Click Next



When Add HDL screen emerges, click Next



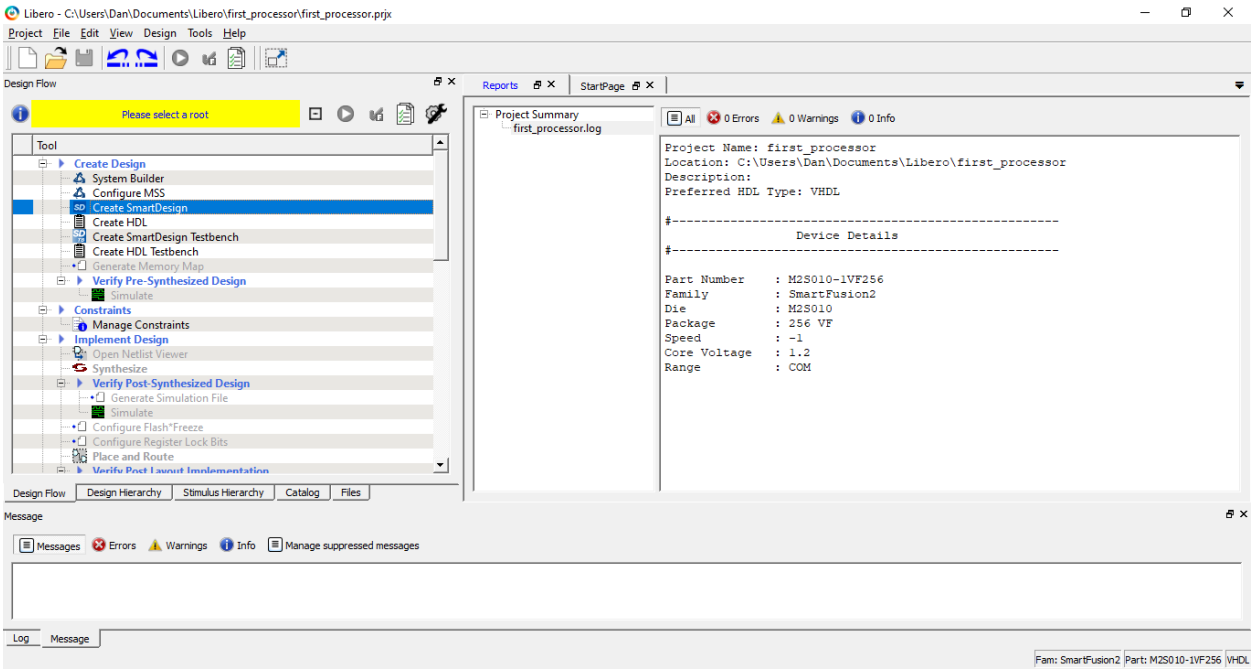
When Add Constraints window emerges, click Finish



Creating Smart Design

In the Design Flow tab double click Create SmartDesign

Enter top as the Design name and click OK



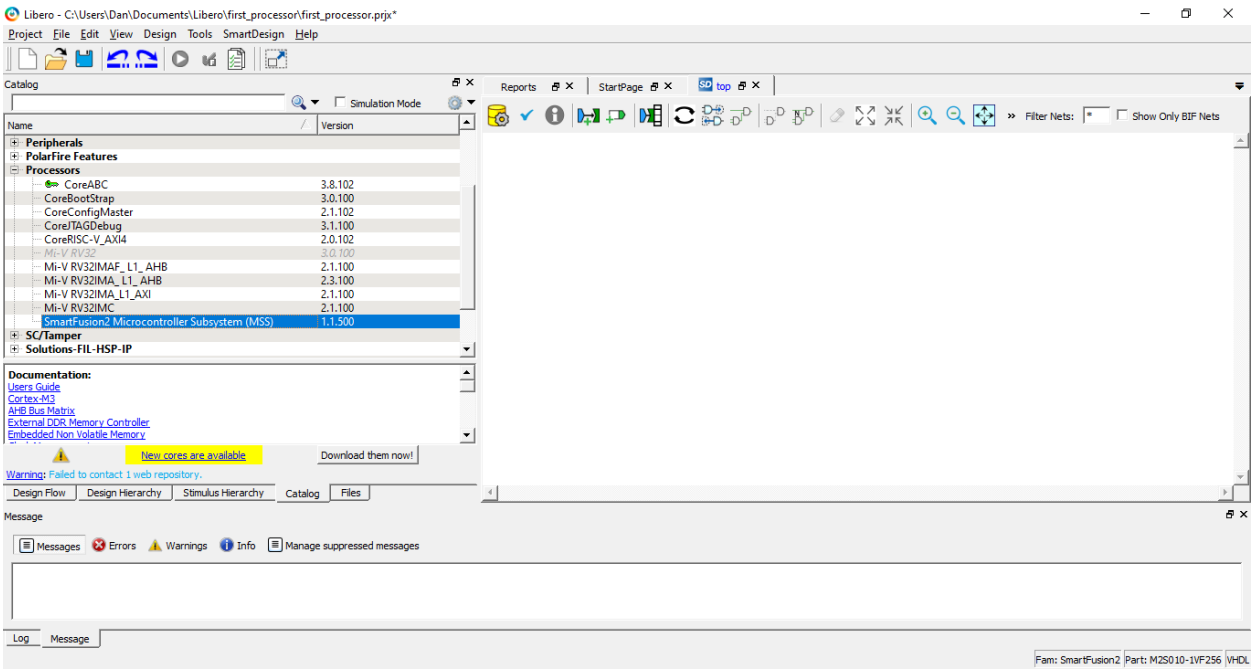
Select Catalog tab

Expand Processors

Select SmartFusion2 Microcontroller Subsystem (MSS)

Note: if SmartFusion2 Microcontroller Subsystem (MSS) is greyed out, select it, right click and click download

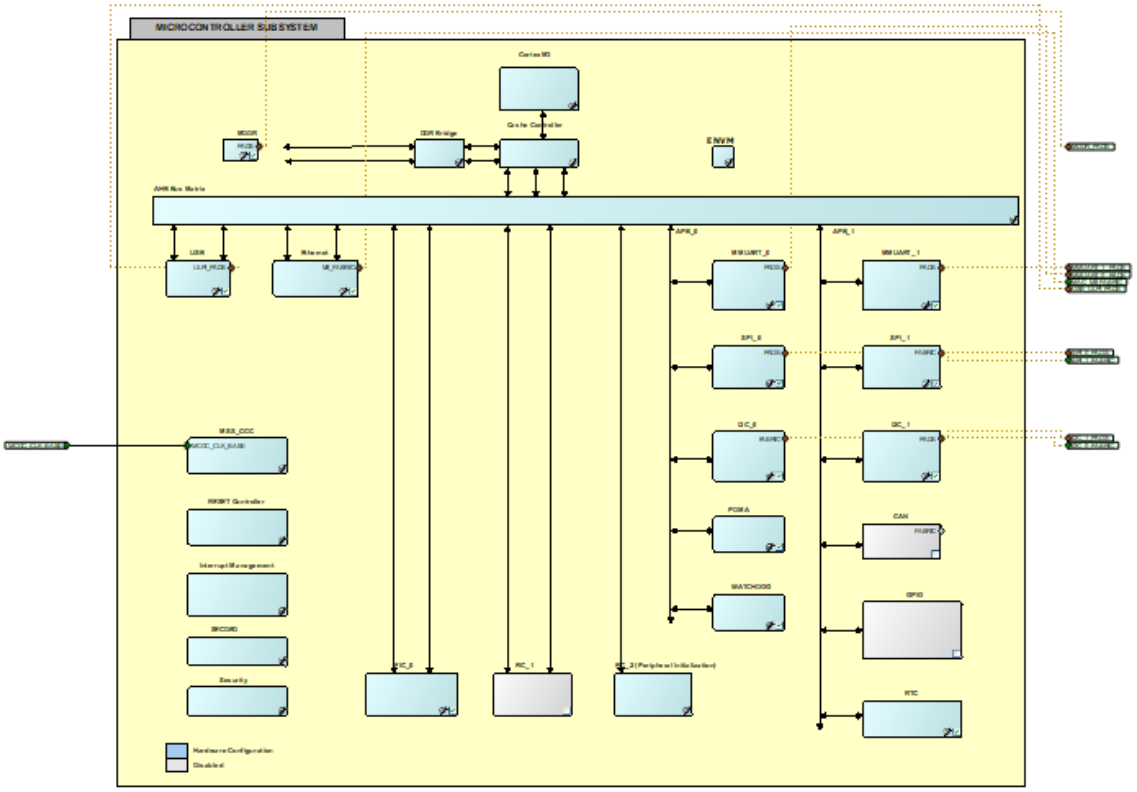
Drag SmartFusion2 Microcontroller Subsytem (MSS) towards the right



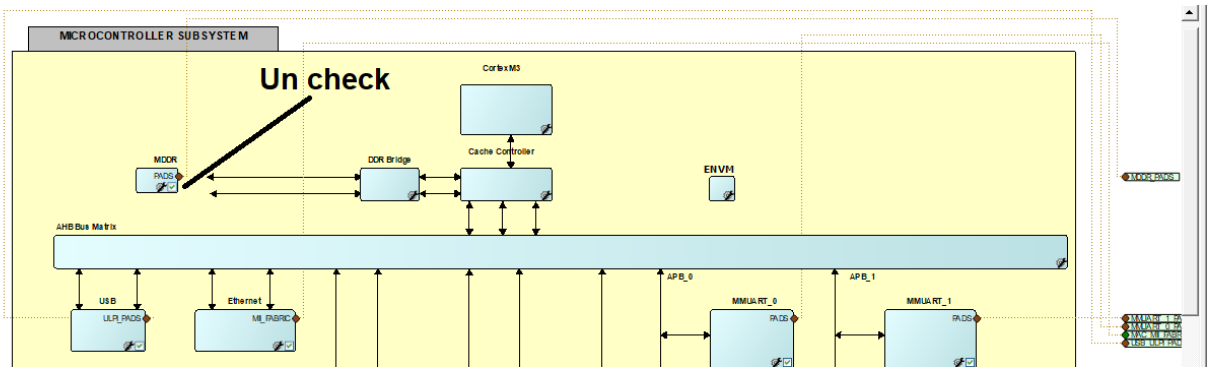
Accept default name and press OK

The MSS_C0 smart design emerges

It should look similar to the image that follows:



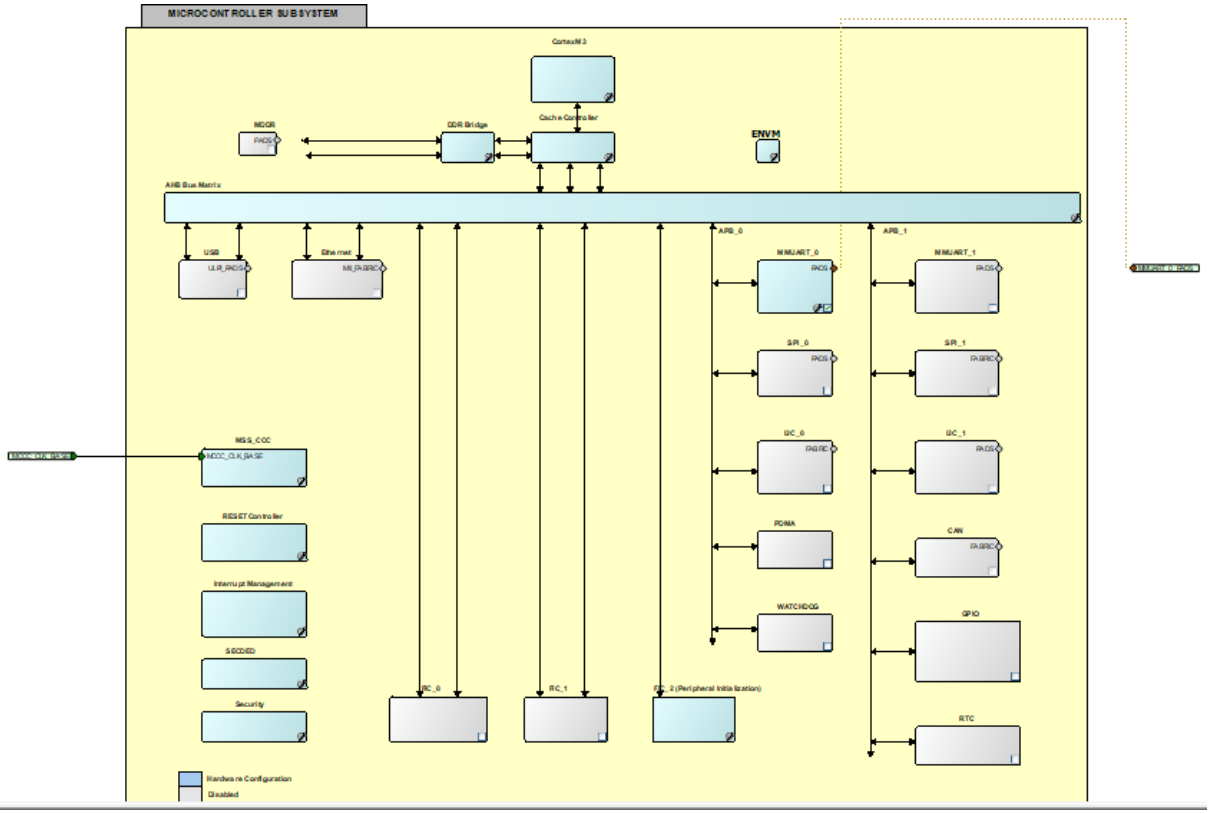
Disable MDDR by unchecking it



In a similar fashion disable the following:

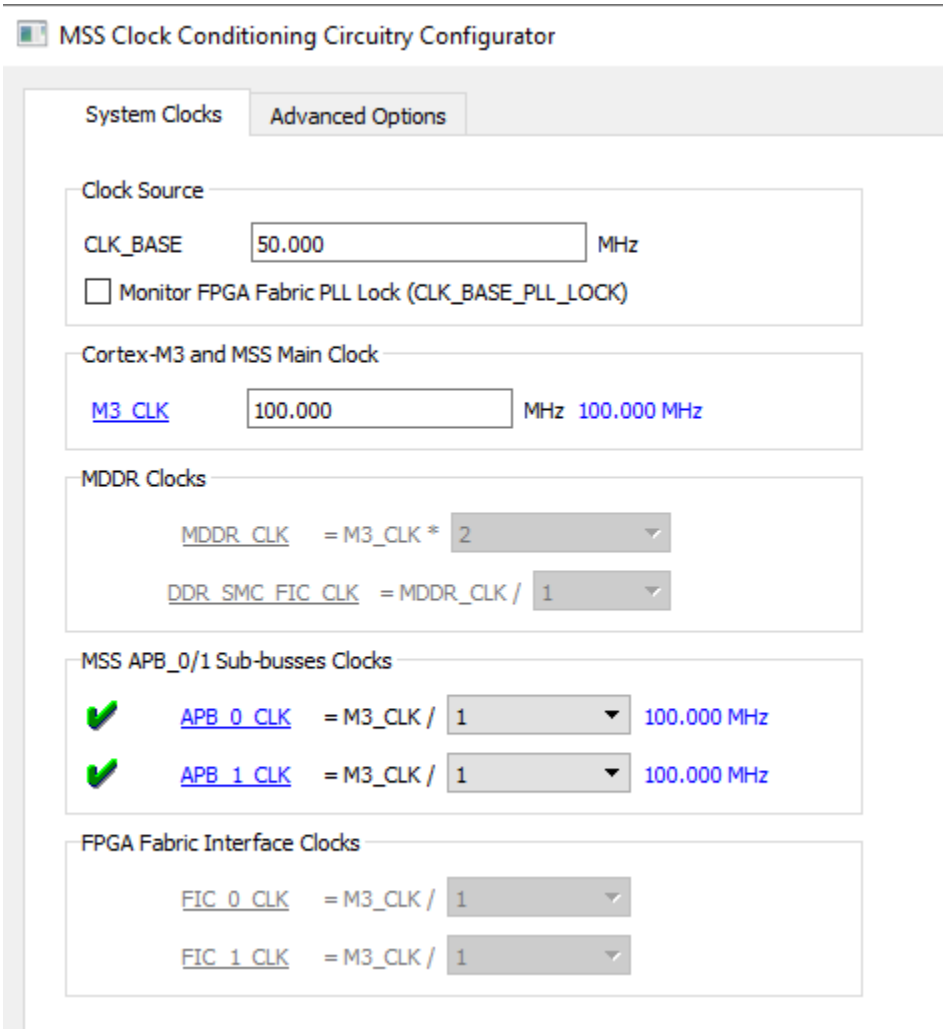
- MMUART_1, SPI_1, I2C_1, CAN, RTC
- SPI_0, I2C_0, PDMA, WATCHDOG
- USB, Ethernet, RC_0, RC_1

When you are finished your MSS should look like the image that follows:



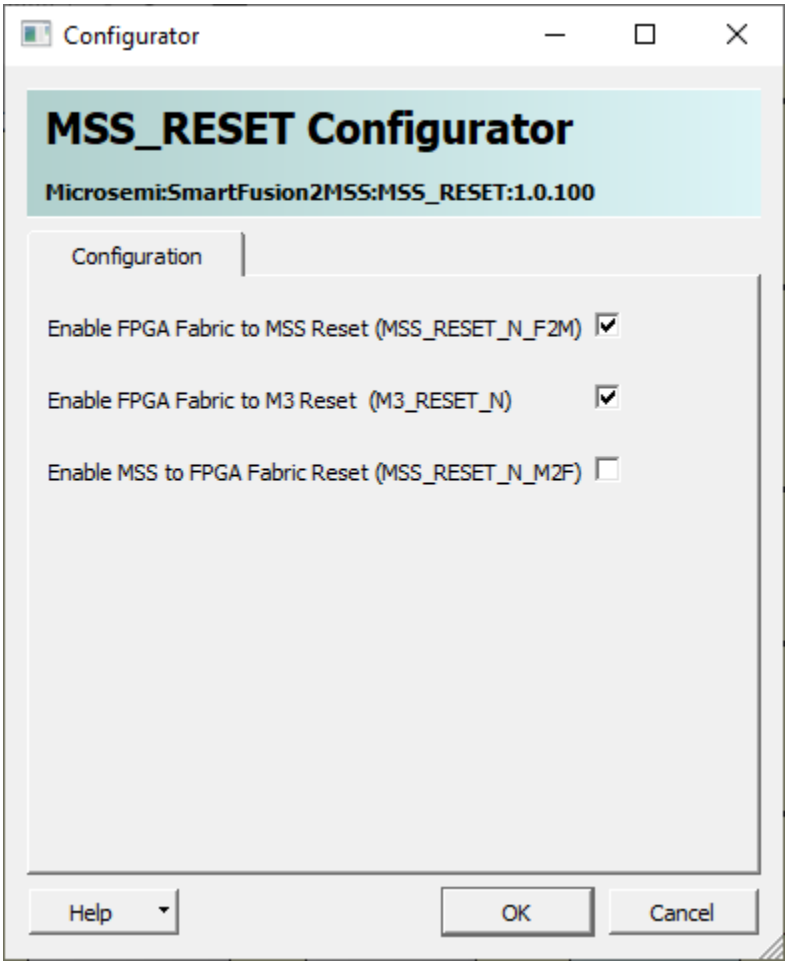
Double click MSS_CCC

Verify that the settings are as follows and press OK to exit



Double click RESET Controller

Apply check box options to match picture below and click OK



Double click ENVM

Double click Data Storage

Enter ENVM as Client Name

Select Intel-Hex firmat

Un-check Use absolute addressing

Select No content (place holder) option

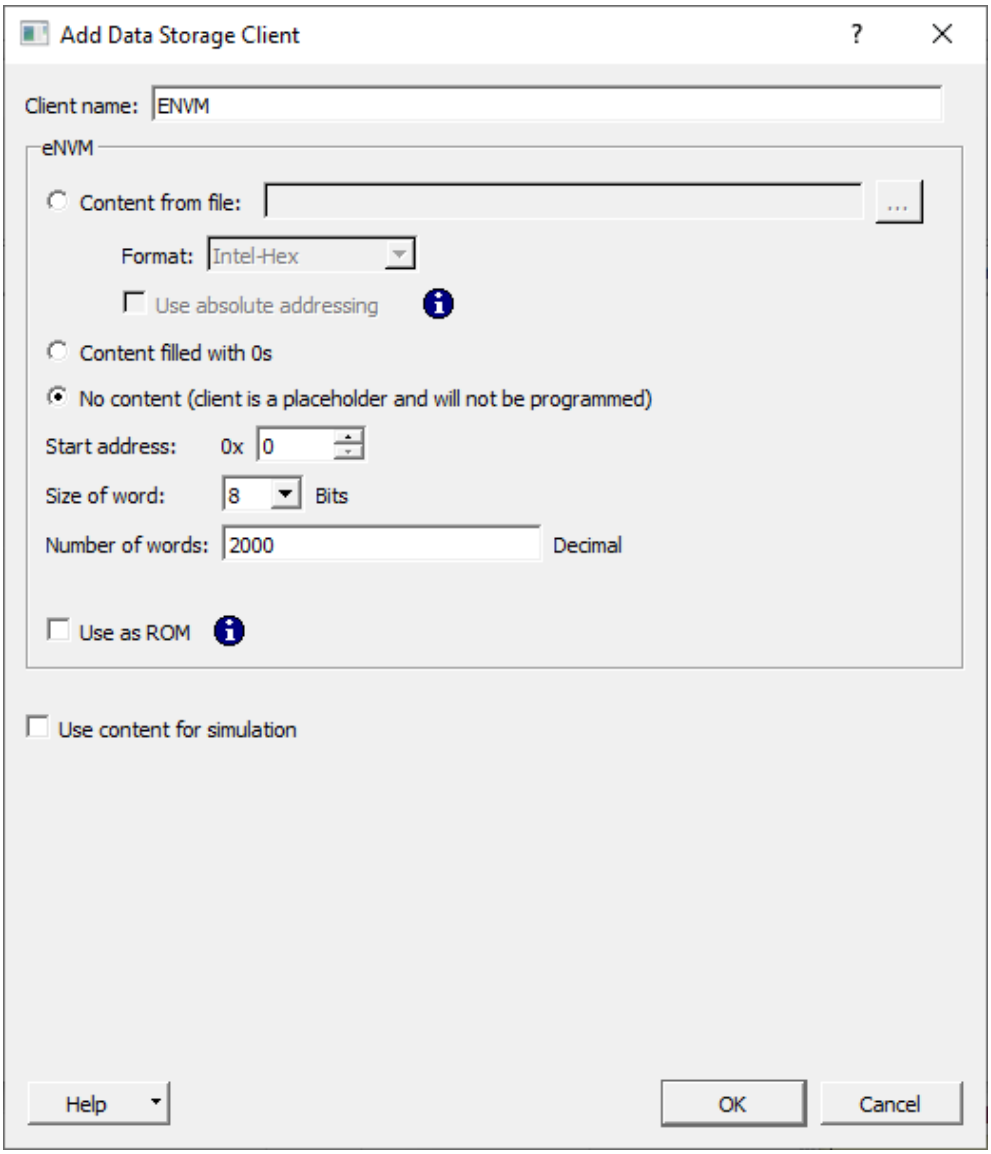
Leave Start Address as 0

Set size of word to 8 Bits

Set Number of words to 2000

Leave other options unchecked

Click OK



Double click MMUART_0

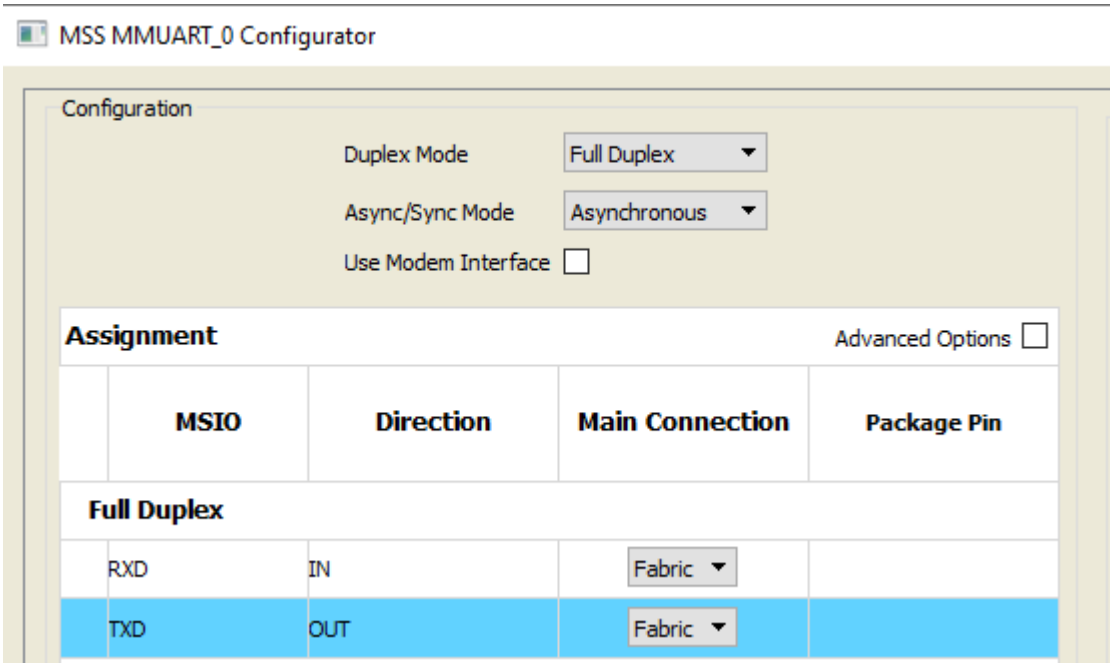
Ensure Duplex Mode is set to Full Duplex

Ensure Async/Sync Mode is set to Asynchronous

For RXD set main connection to Fabric

For TXD set main connection to Fabric

Click OK



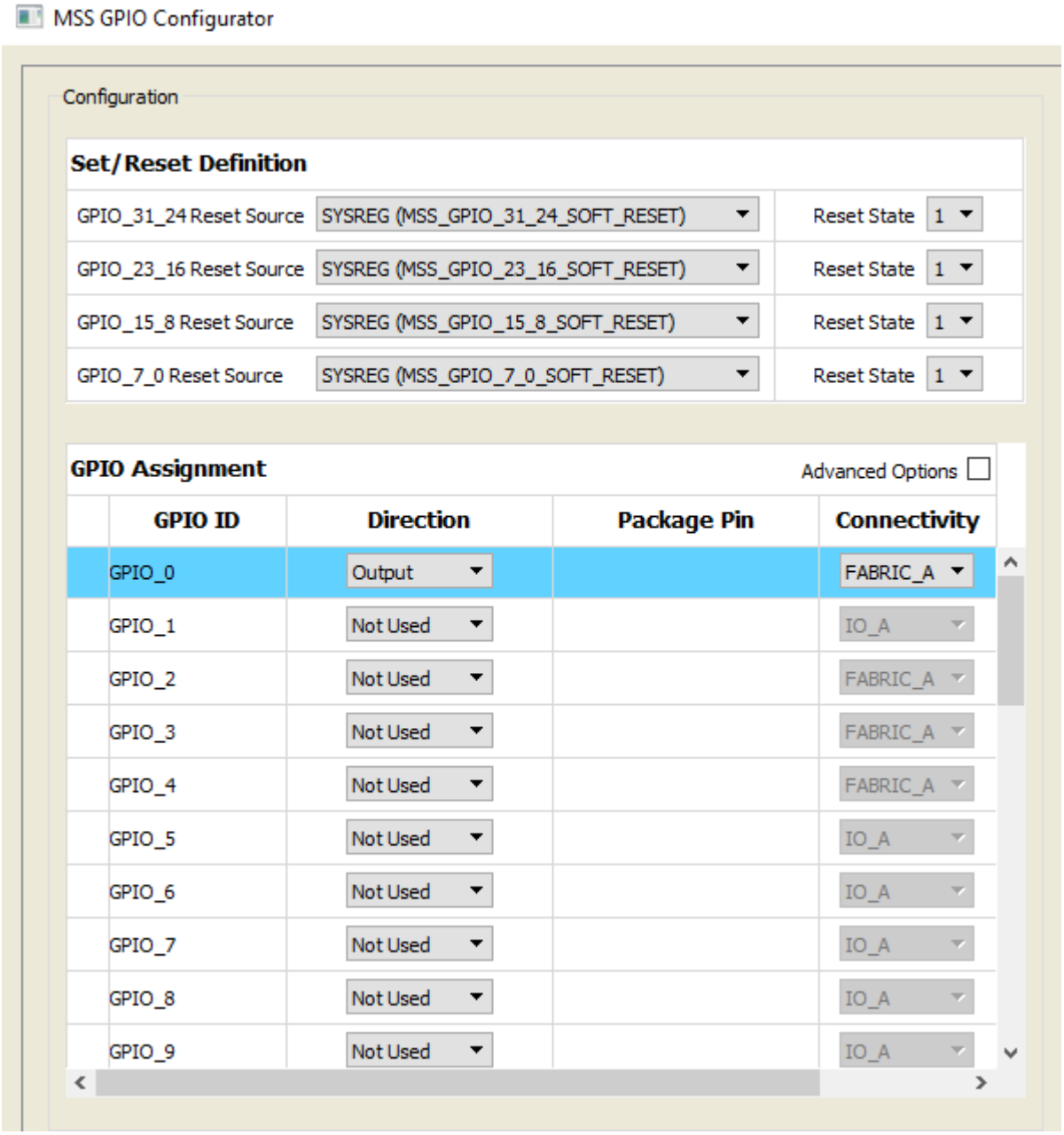
Apply Checkmark to GPIO to enable it

Double click GPIO

Set GPIO_0 Direction to Output

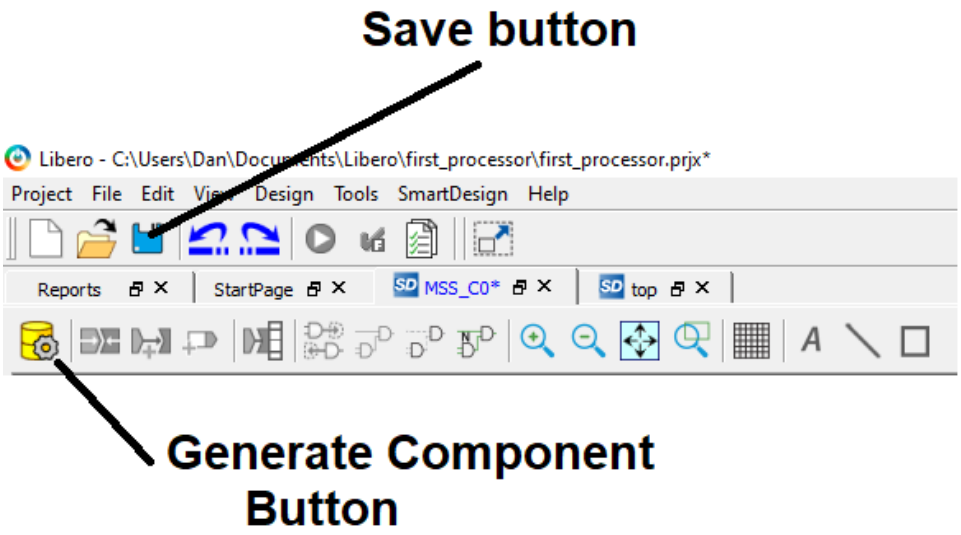
Set GPIO_0 Connectivity to FABRIC_A

Click OK



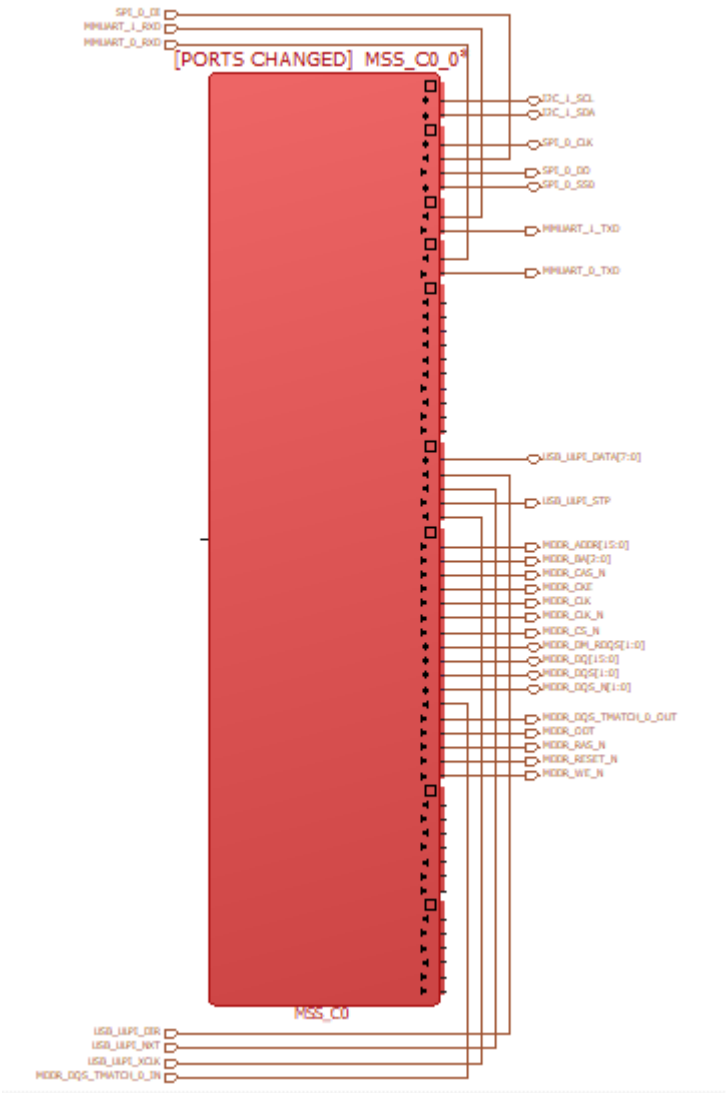
Save MSS_C0

Click Generate Component



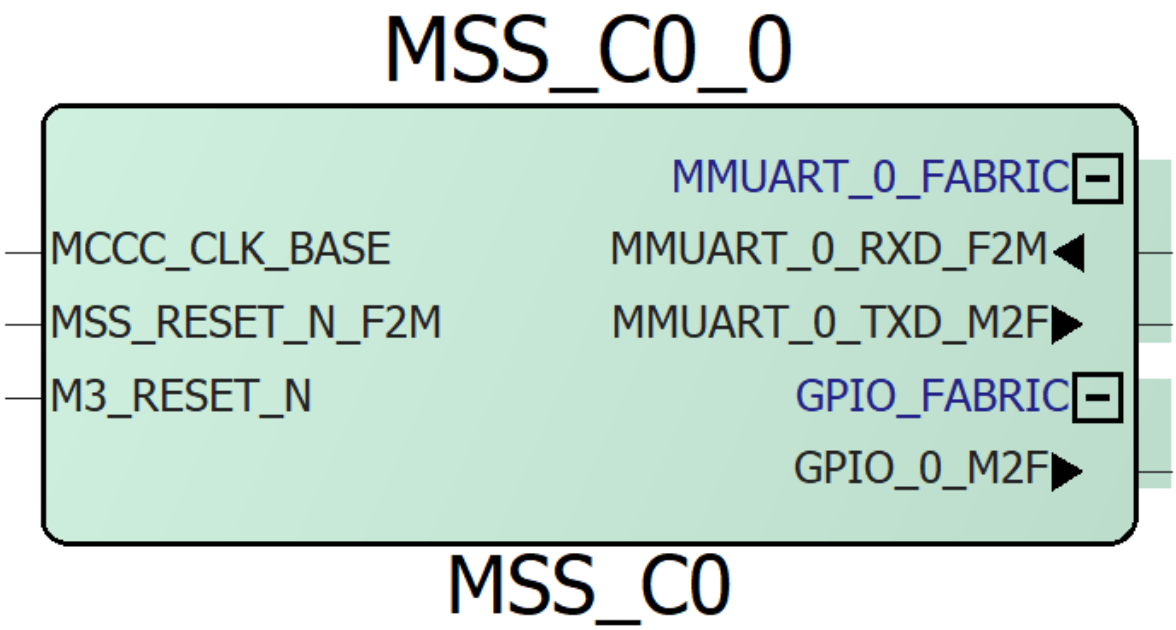
Go to top smart design

The symbol for MSS_C0_0 will be red (because it needs to be updated)



Right click and select Update Component

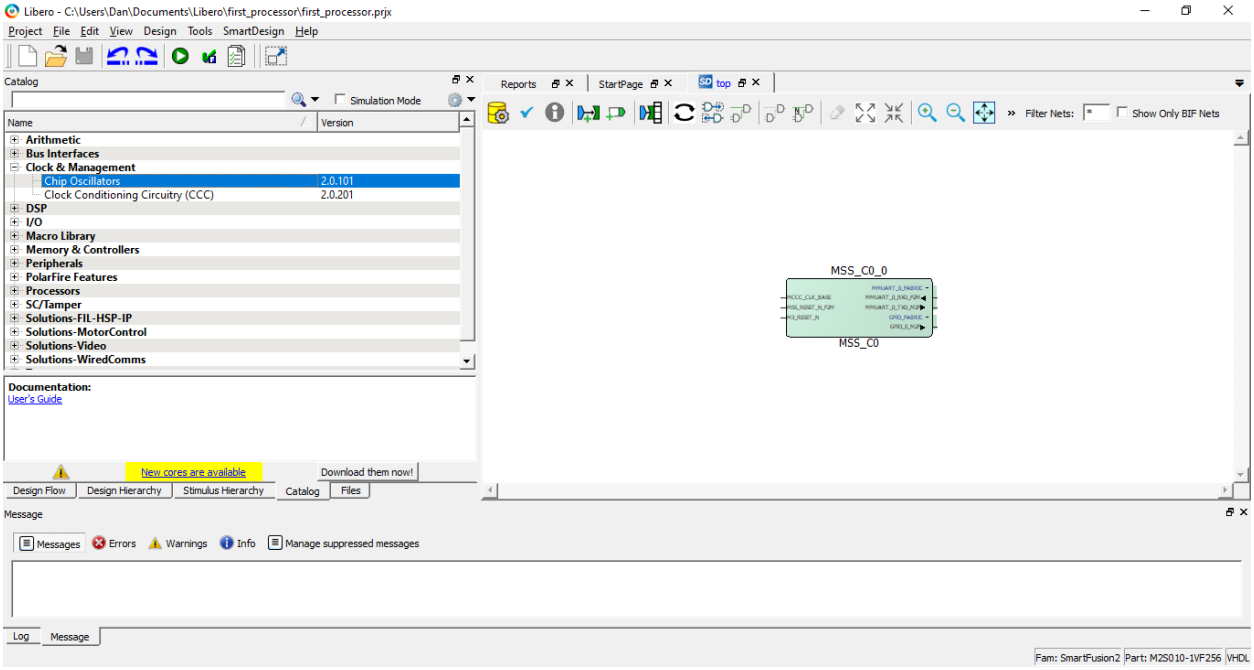
Afterwards the component should look like the image that follows



Select Catalog tab

Expand Clock & Management

Select Chip Oscillators



Drag Chip Oscillators into top

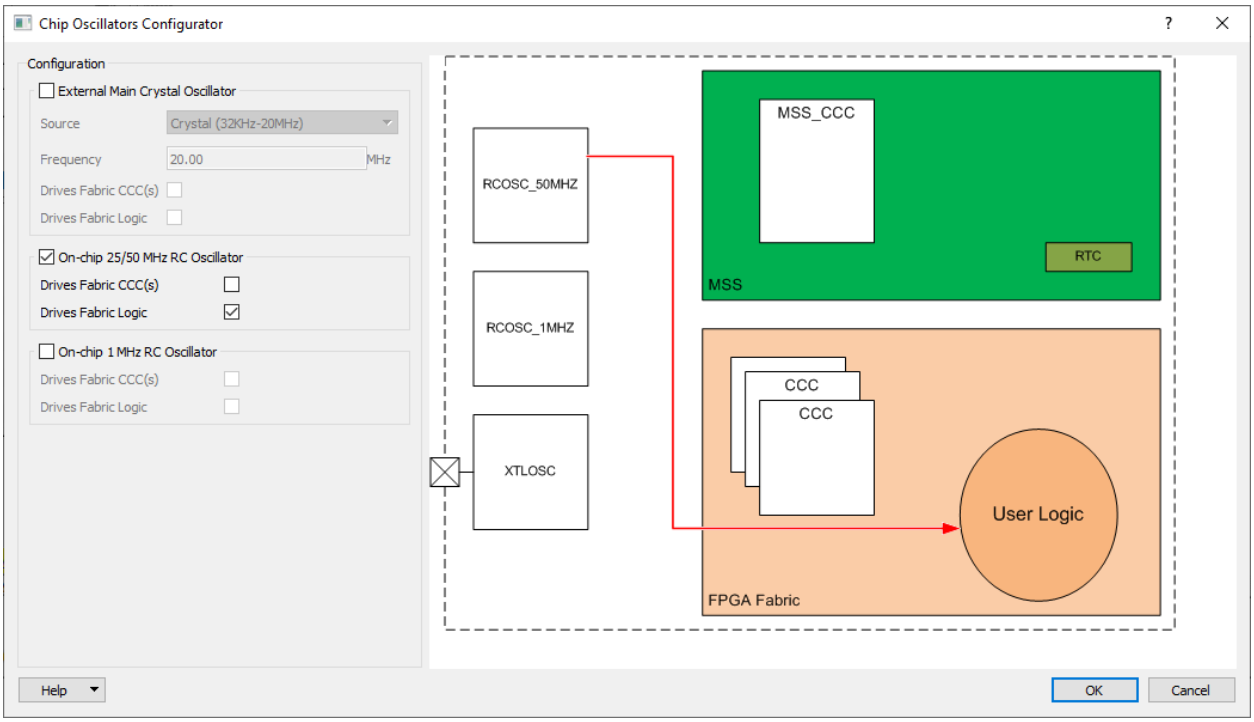
Accept default name (OSC_C0) and click OK

The Chip Oscillators Configurator window will pop up

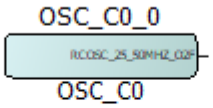
Apply Checkmark to select On-chip 25/50 MHz RC Oscillator

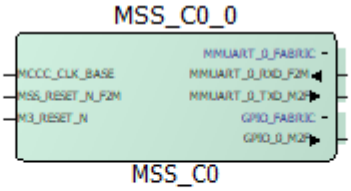
Apply Checkmark to Drives Fabric Logic Option

Press OK



Oscillator OSC_C0_0 will now appear in top

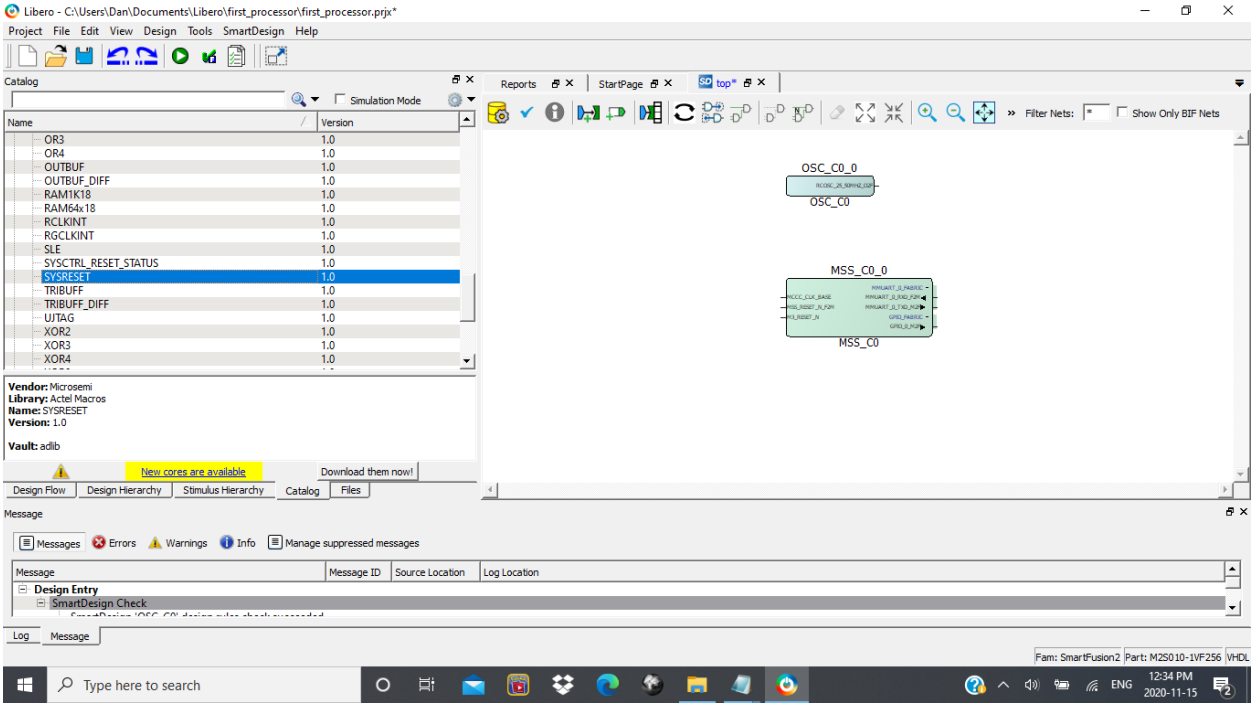




Select Catalog tab

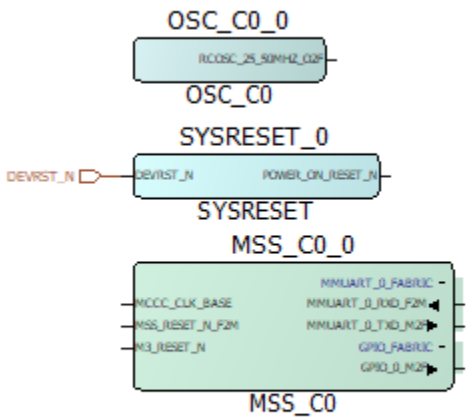
Expand Macro Library

Select SYSRESET

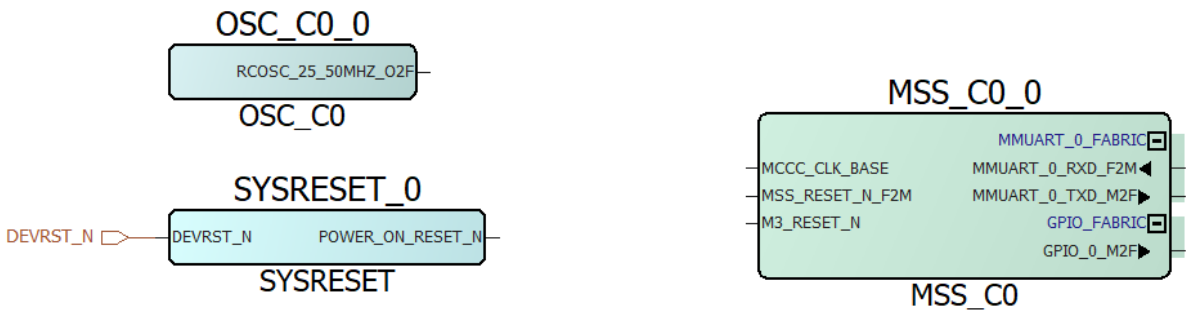


Drag SYSRESET into top

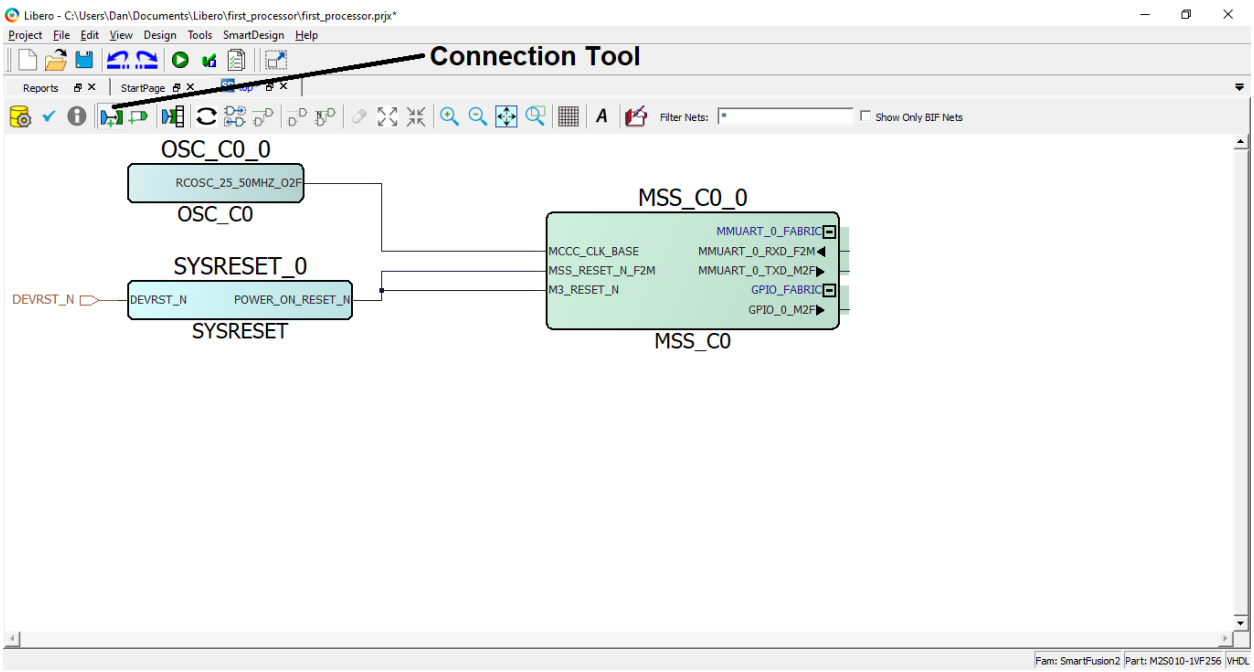
SYS_RESET_0 should now appear in top



Rearrange blocks similar to example pictured below



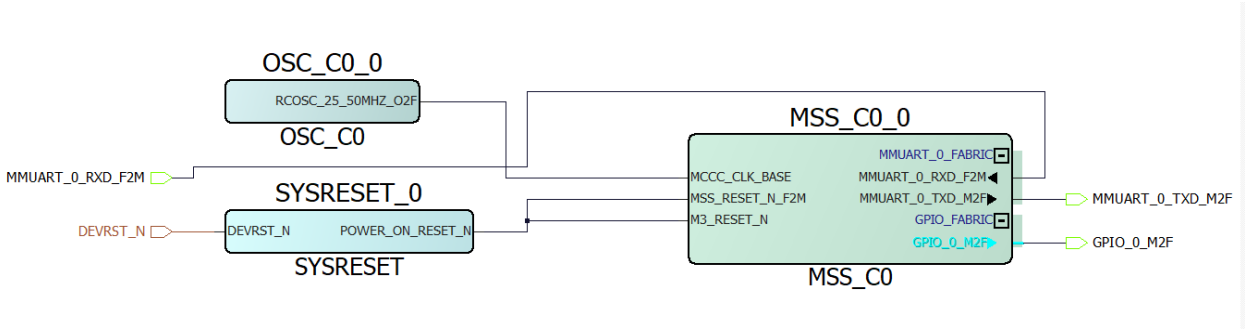
Use connection tool and make connections shown below



Right click the MMUART_0_RXD_F2M pin and select promote to top level

Do the same with the MMUART_0_TXD_M2F and GPIO_0_M2F pins

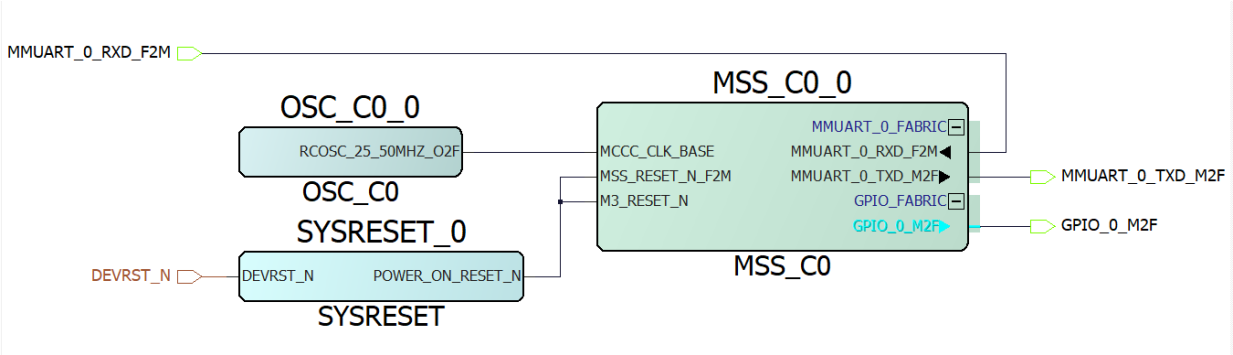
The Smart Design should now resemble the example below



Unselect connection tool

Click auto-arrange

You will notice that things got prettier



Save top

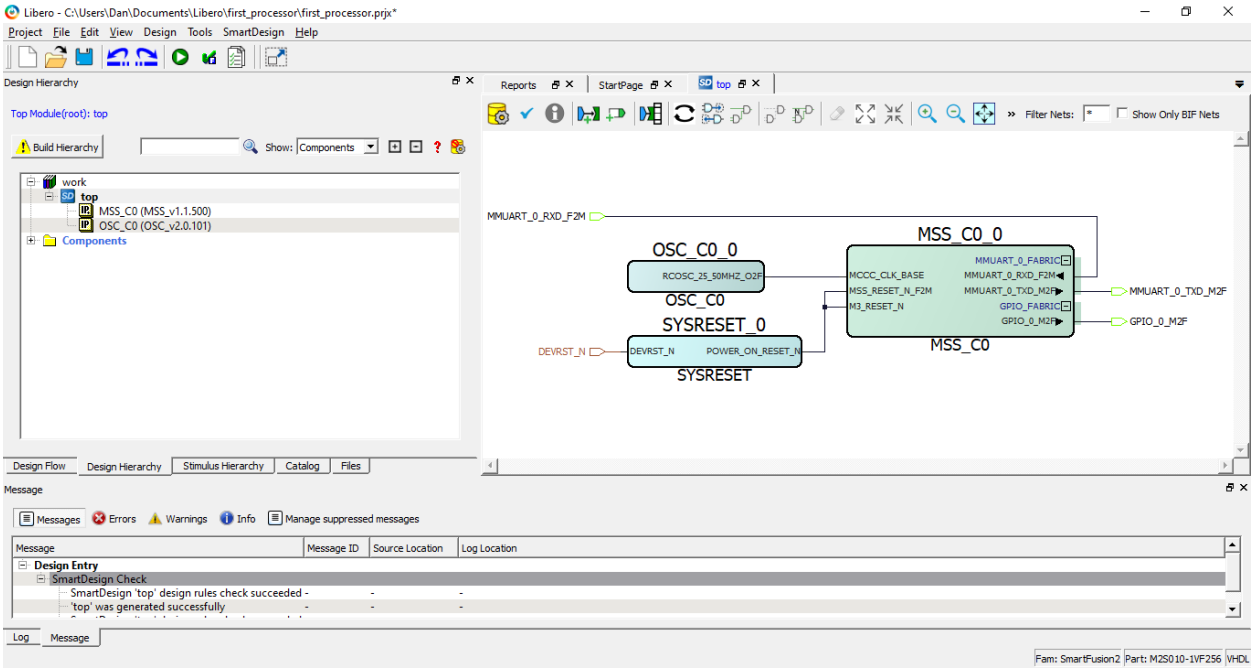
Click generate

Select Design Hierarchy tab

Right click top

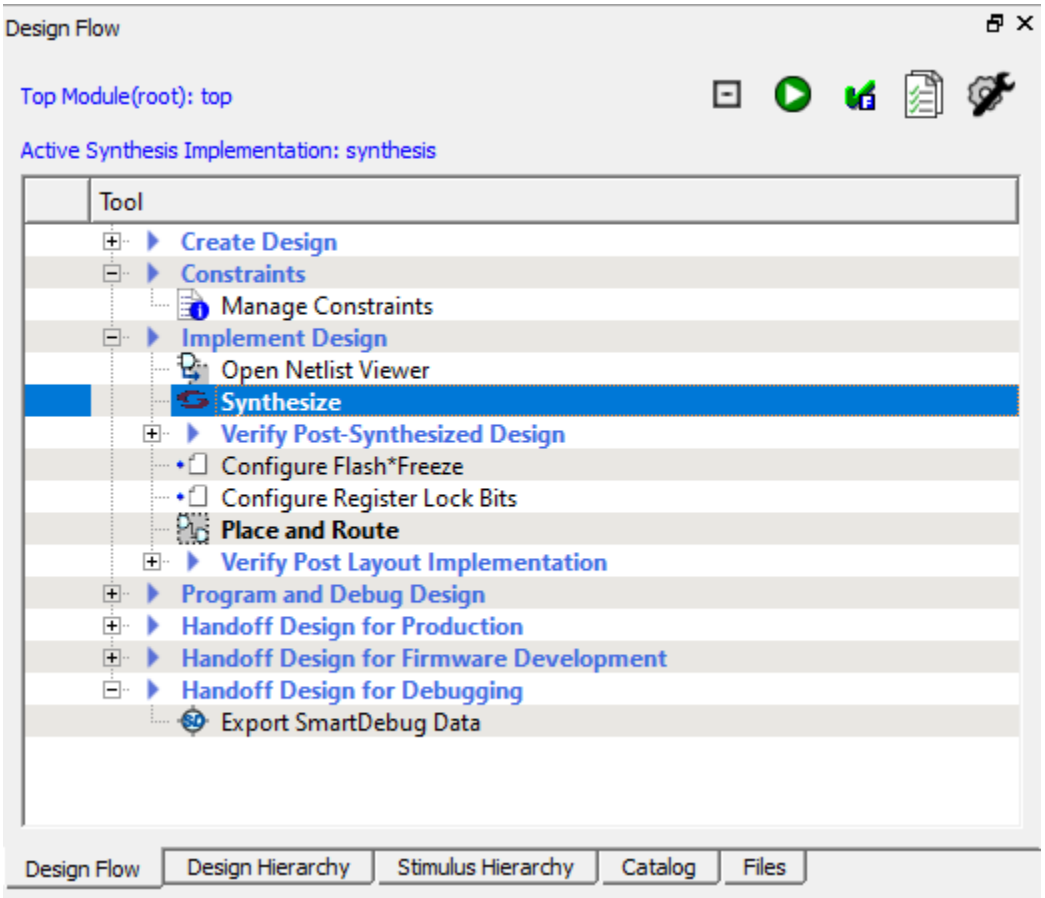
Select Set as Root

Now top should be recognized as the true top level



Double click Synthesize

When complete a check box will appear next to Synthesize



Double click Manage Constraints

In Constraint Manager select I/O Attributes tab

Click Edit

Then click edit with I/O Editor

Set pin number of GPIO_0_M2F to T12

Set pin number of MMUART_0_RXD_F2M to H13

Set pin number of MMUART_0_TXD_M2F to H12

Refer to diagram below to double check pin assignments

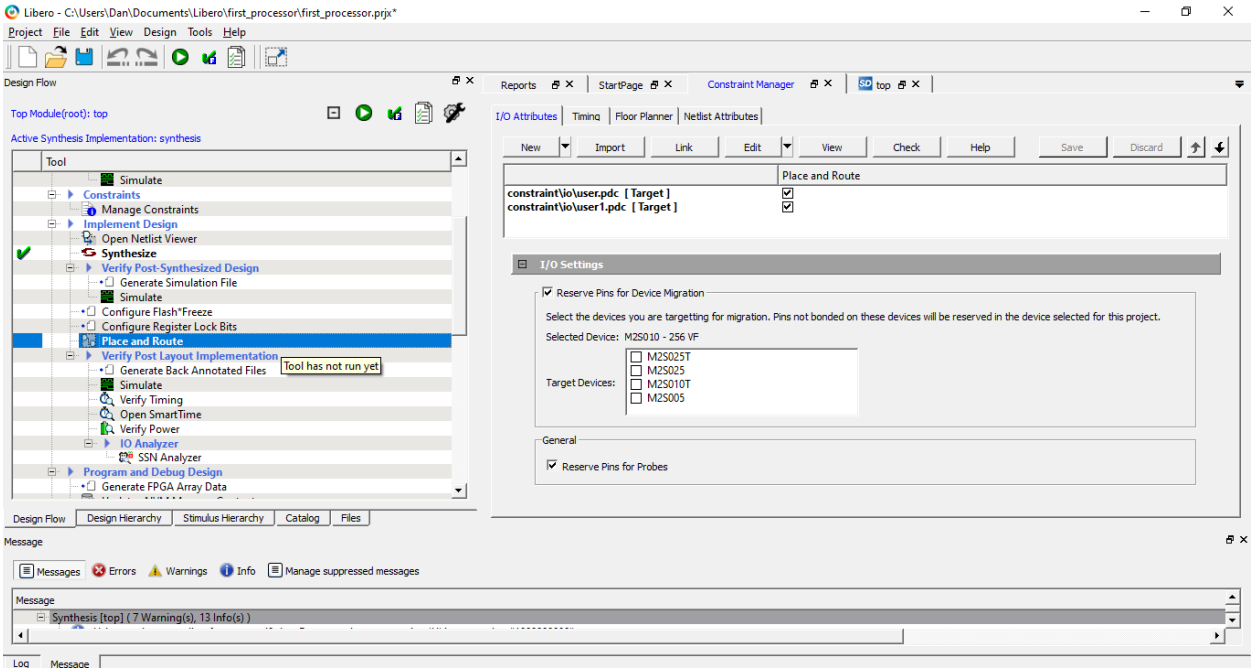
Save I/O Editor

Port View [active] | Pin View | Package View | Floorplanner View | Netlist Viewer - Hier

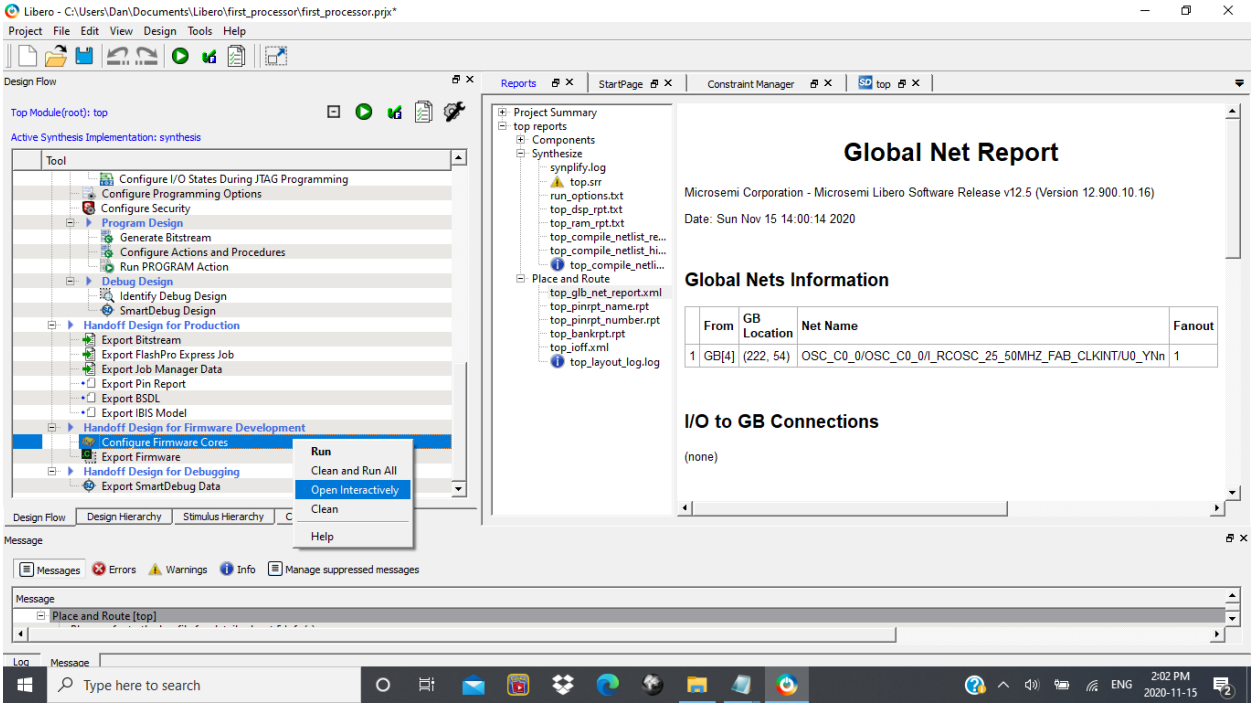
	Port Name	Direction	I/O Standard	Pin Number	Locked
1	DEVRST_N	INPUT	--	M11	☑
2	GPIO_0_M2F	OUTPUT	LVCMOS33	T12	☑
3	MMUART_0_RXD_F2M	INPUT	LVCMOS33	H13	☑
4	MMUART_0_TXD_M2F	OUTPUT	LVCMOS33	H12	☑

Double click Place and Route

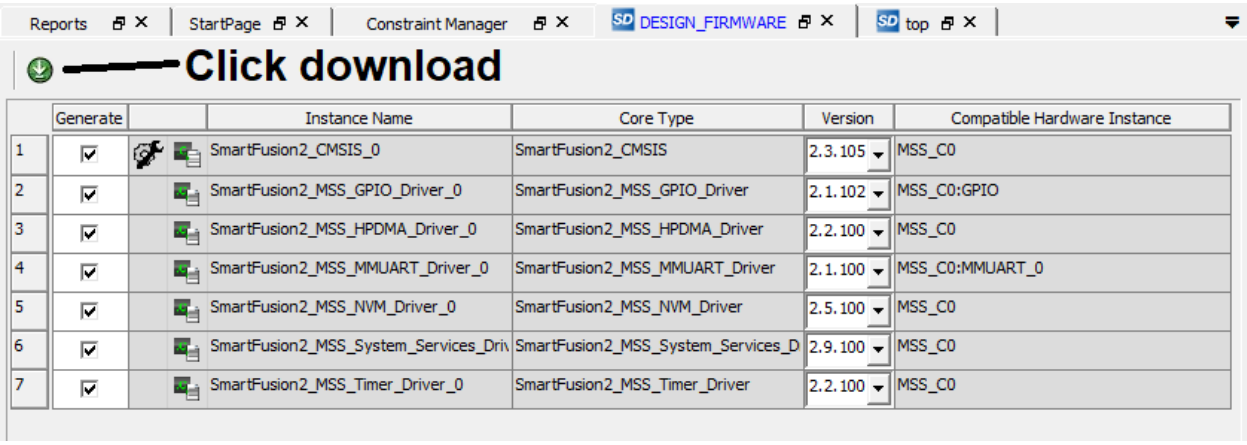
When Complete a check mark will appear next to Place and Route



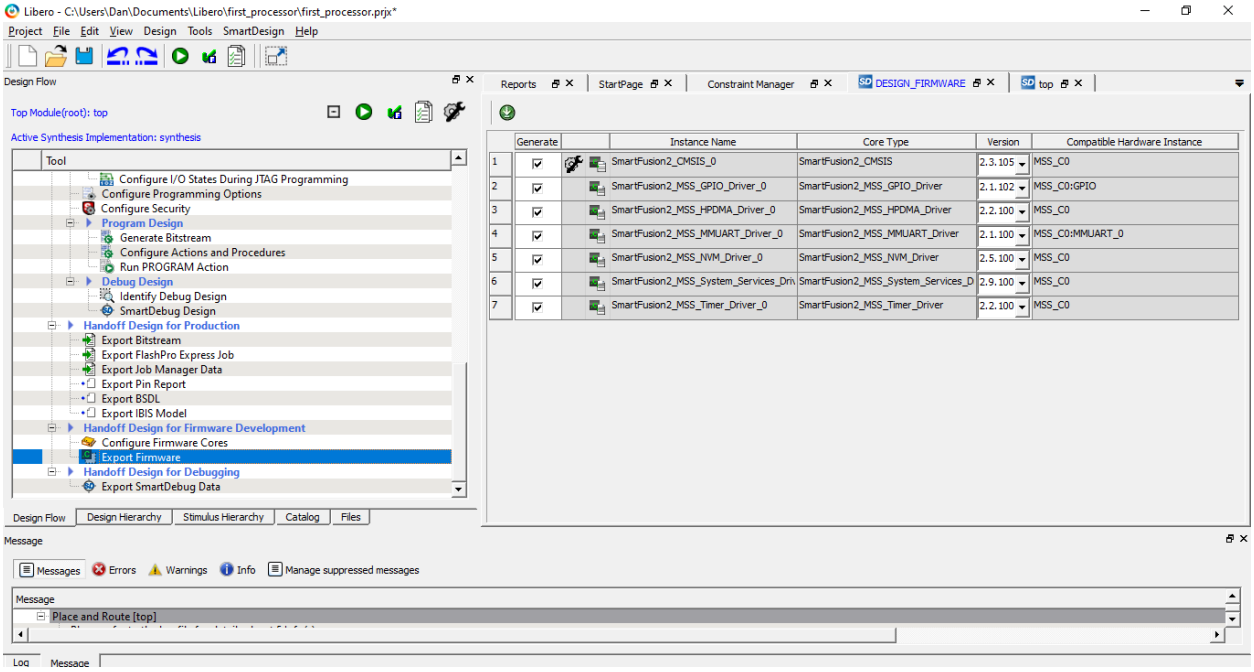
Right click Configure Firmware Cores and select Open Interactively



Click download

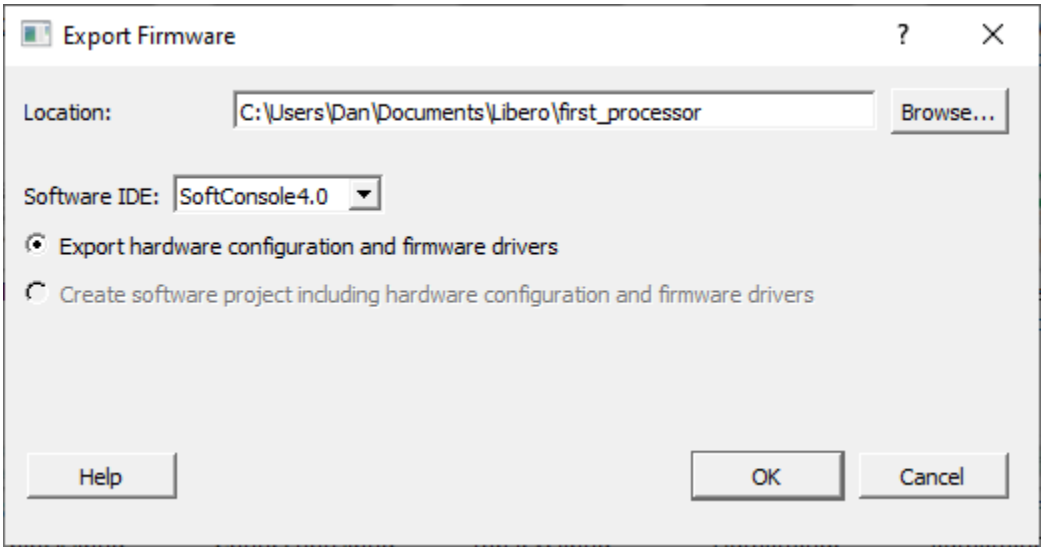


Double click Export Firmware



The location should be set to your project folder by default

Select SoftConsole4.0 as the Software IDE



**** If Export Firmware fails unselect MSS_GPIO_Driver_0 and double click Export Firmware again

When firmware is successfully exported you should see that the project's folder now includes a firmware subfolder

Name	Date modified	Type	Size
component	2020-11-14 10:17 AM	File folder	
constraint	2020-11-15 1:37 PM	File folder	
designer	2020-11-15 1:37 PM	File folder	
firmware	2020-11-15 2:20 PM	File folder	
hdl	2020-11-14 8:40 AM	File folder	
simulation	2020-11-15 1:48 PM	File folder	
smartgen	2020-11-15 2:20 PM	File folder	
stimulus	2020-11-14 8:40 AM	File folder	
synthesis	2020-11-15 1:37 PM	File folder	
tooldata	2020-11-15 2:20 PM	File folder	
first_processor.prjx	2020-11-15 1:48 PM	PRJX File	14 KB

It is critical that the firmware subfolder appear here in this location.

If you were able to successfully export the firmware on your first attempt you can skip ahead to page 40.

For Those That Had Trouble Generating Firmware

If you had to de-select MSS_GPIO_Driver_0 in order to export firmware then follow steps below:

Download the Digital_Signal_Processing_demo sub design from the following webpage:

https://www.microsemi.com/existing-parts/parts/150925#resources

Extract zip folder

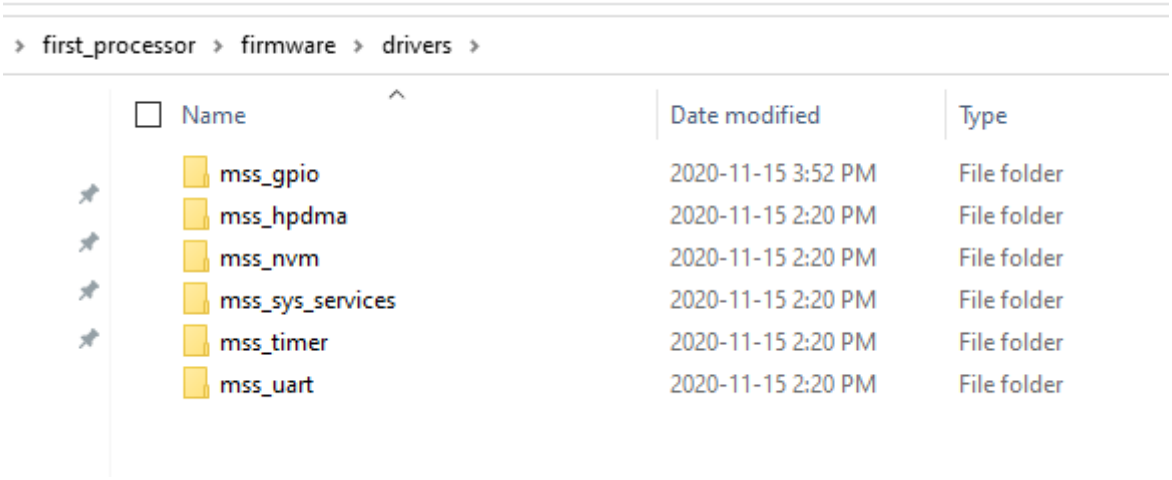
Explore extracted folders and navigate to:
Digital_Signal_Processing_demo\DSP_FIR_FF_LCD_12_2\firmware\drivers

copy mss_gpio folder

Explore your project folder and navigate to the first_processor\firmware\drivers\ folder

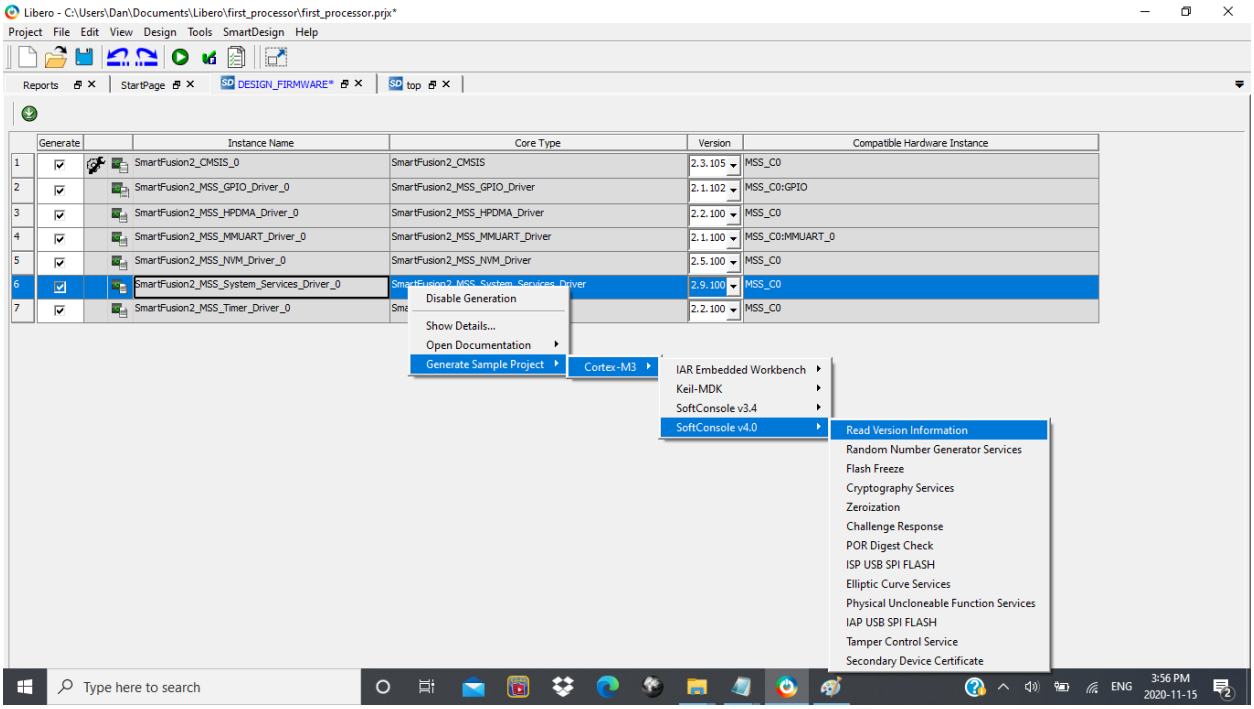
Paste mss_gpio

Your firmware\drivers\ subfolder should now look like:

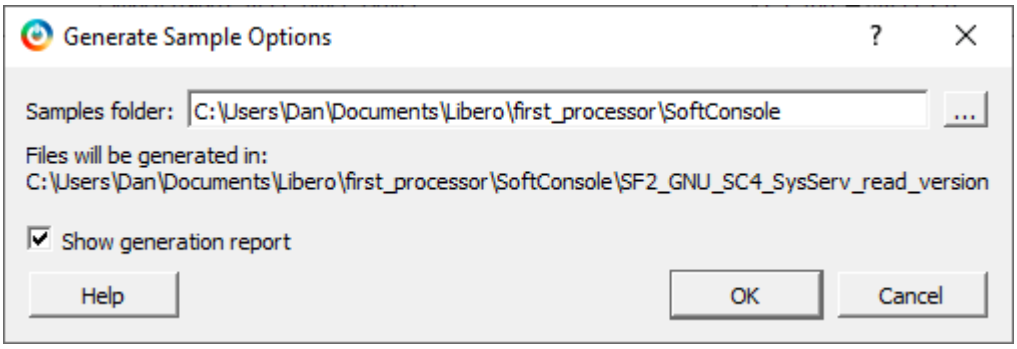


In DESIGN_FIRMWARE window select SmartFusion2_MSS_System_Services_Driver_0

Right click and select Generate Sample Project → Cortex M3 → SoftConsole4.0 → Read Version Information



Accept the default location and click OK



When next popup emerges click close

Explore your project folder

Ensure that the SoftConsole folder was created (as seen below)

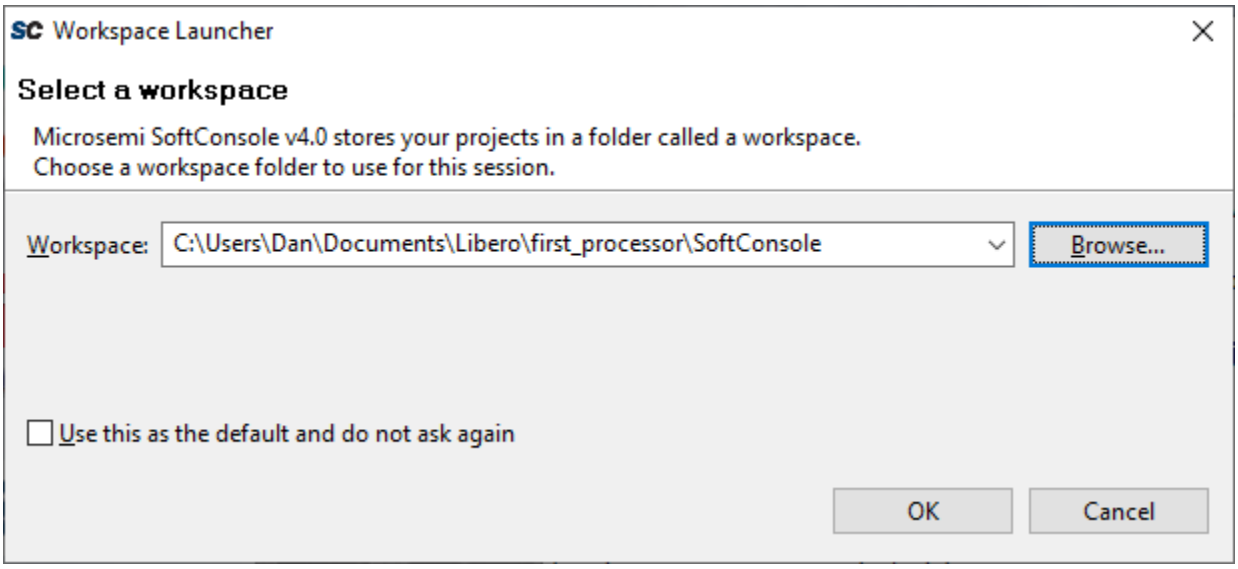
It is critical that this folder was produced in the proper location

Name	Date modified	Type	Size
component	2020-11-14 10:17 AM	File folder	
constraint	2020-11-15 1:37 PM	File folder	
designer	2020-11-15 1:37 PM	File folder	
firmware	2020-11-15 2:20 PM	File folder	
hdl	2020-11-14 8:40 AM	File folder	
simulation	2020-11-15 1:48 PM	File folder	
smartgen	2020-11-15 2:20 PM	File folder	
SoftConsole	2020-11-15 4:04 PM	File folder	
stimulus	2020-11-14 8:40 AM	File folder	
synthesis	2020-11-15 1:37 PM	File folder	
tooldata	2020-11-15 2:20 PM	File folder	
first_processor.prjx	2020-11-15 1:48 PM	PRJX File	14 KB

Using SoftConsole 4.0 IDE

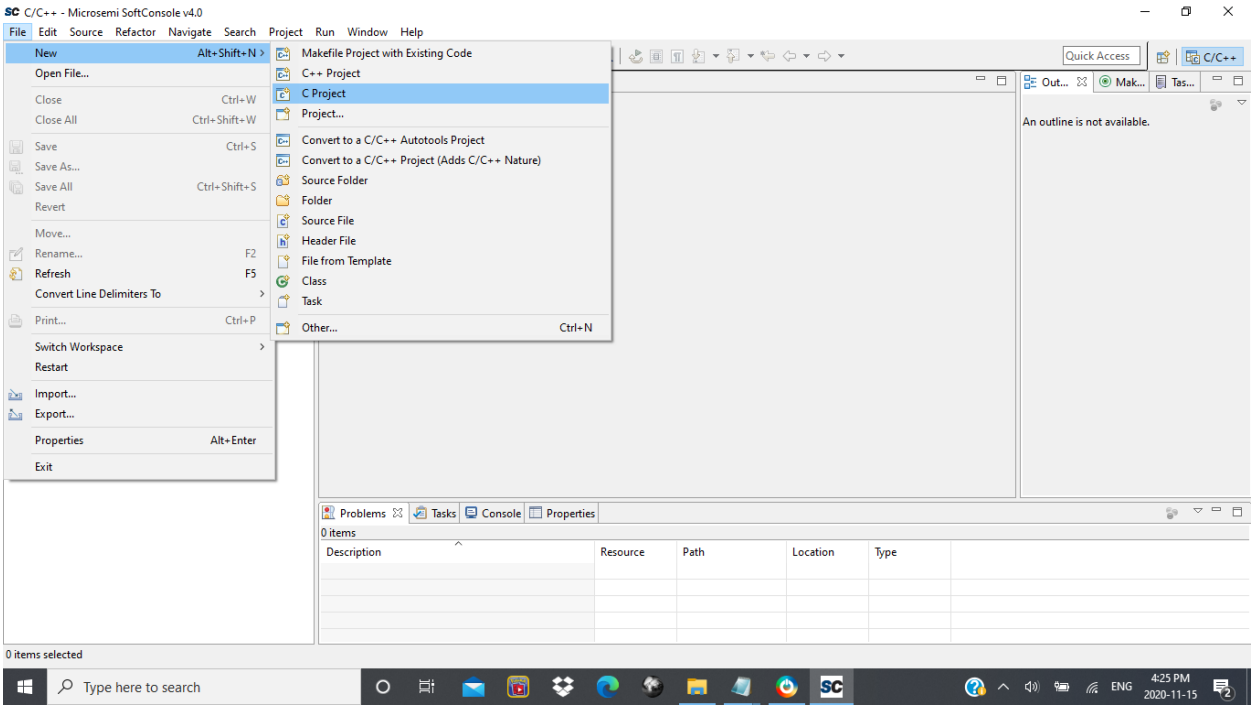
Open Microsemi SoftConsole v4.0 program

For workspace browse to your project's SoftConsole subfolder and click OK



The SoftConsole IDE will open

Click File → New → C Project

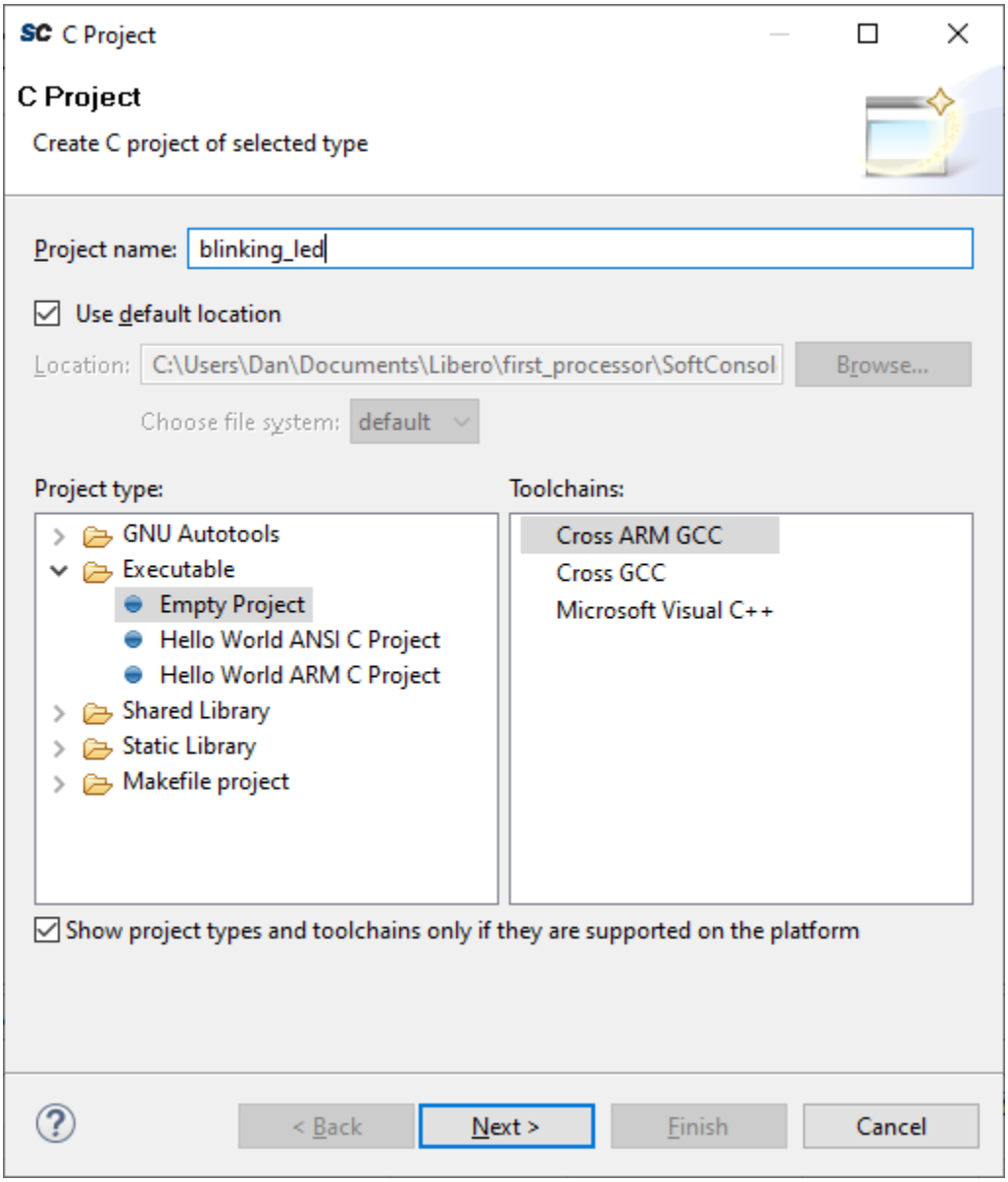


For project name type blinking_led

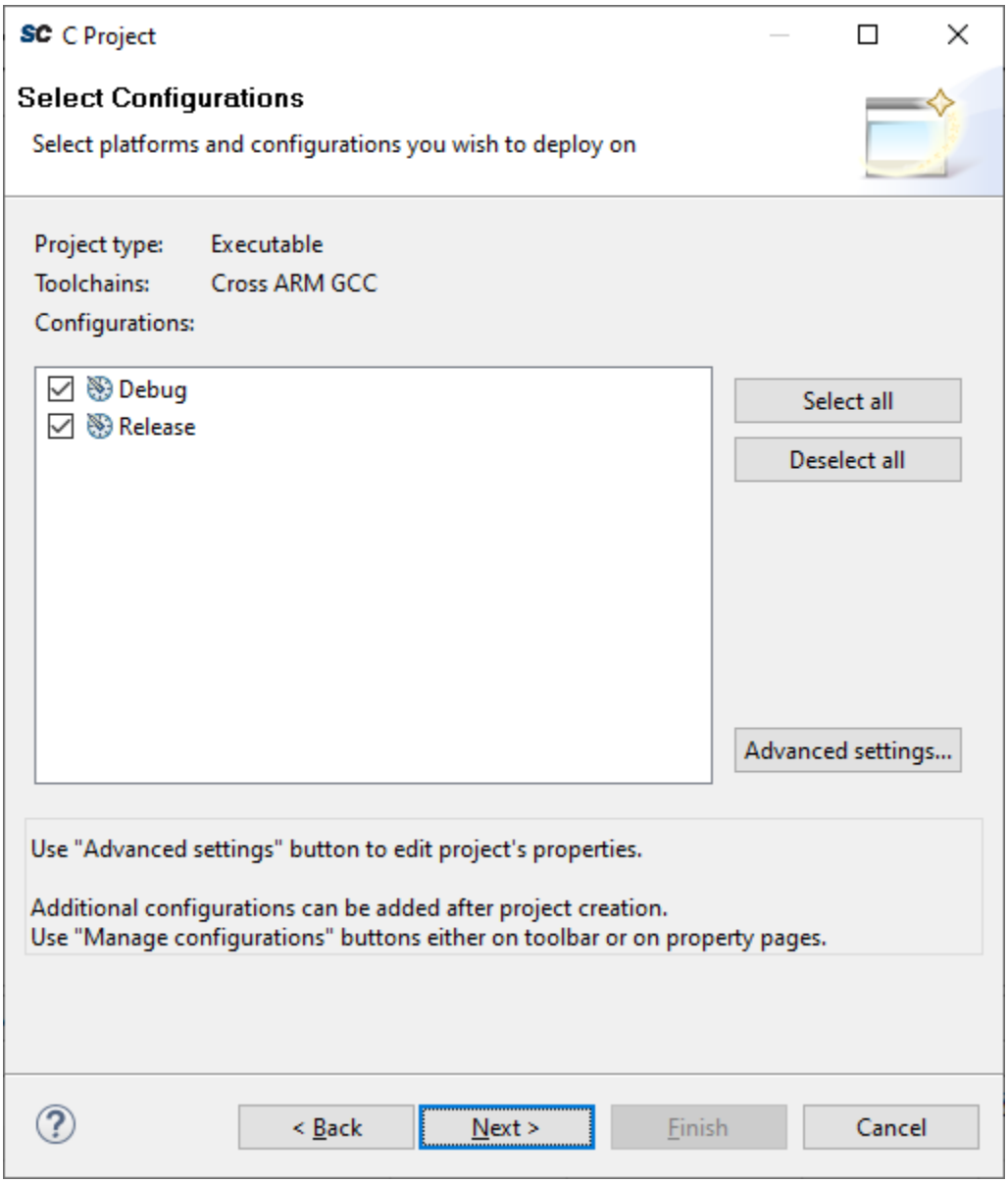
For Project type select Executable -- Empty Project

For tool chain select Cross ARM GCC

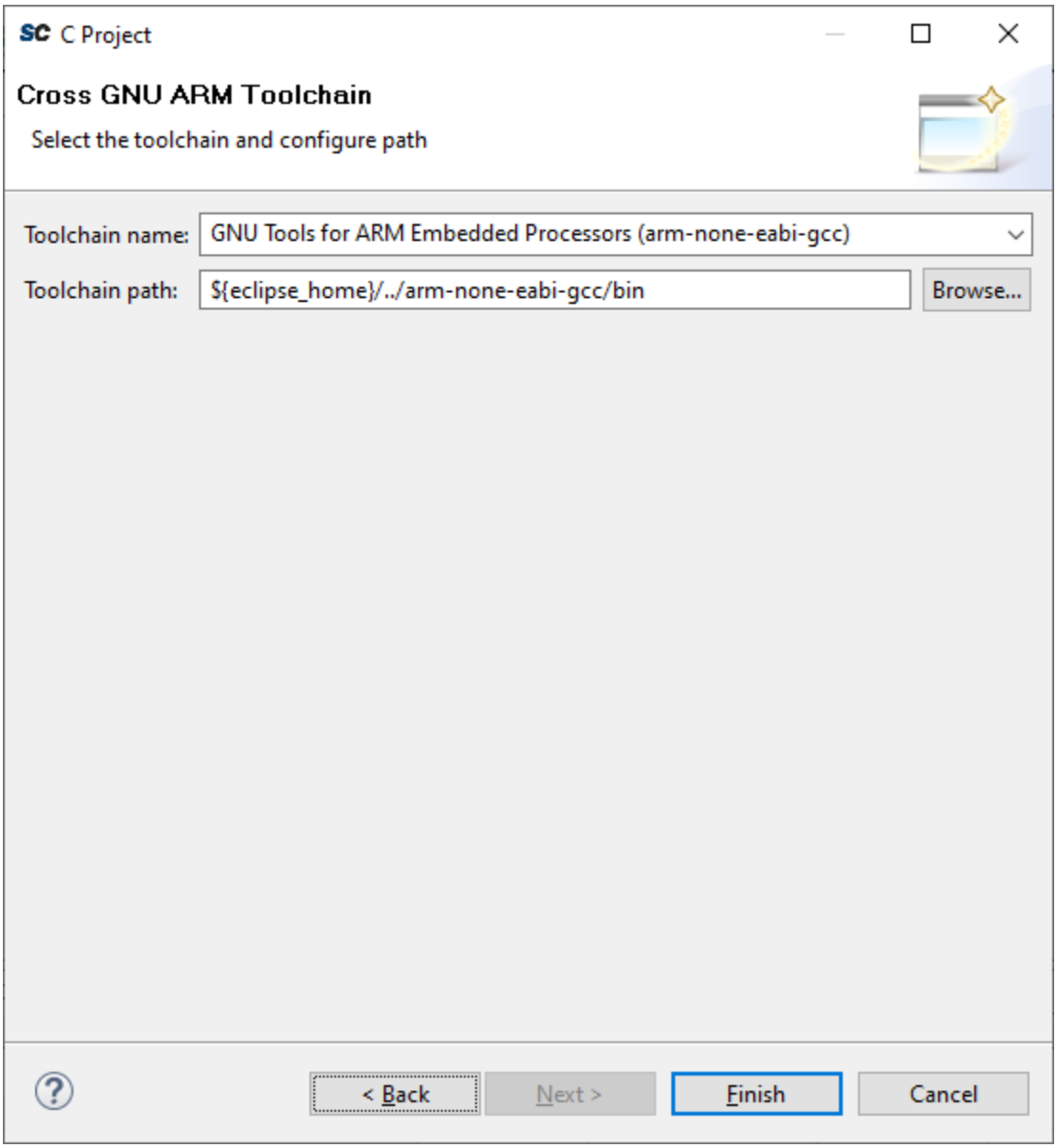
Click Next



Ensure that Debug and Release are checked and click Next



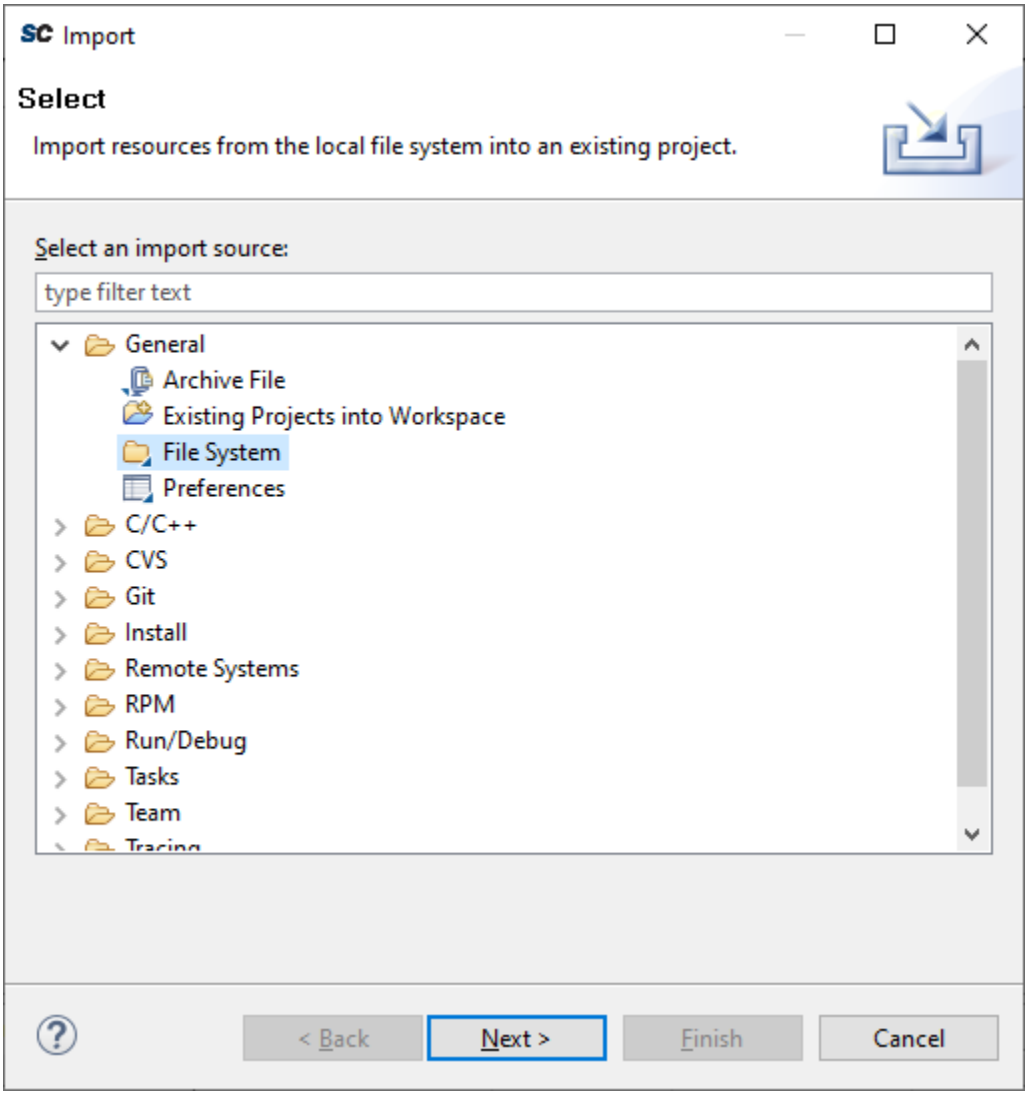
Accept default toolchain name and path and click Finish



Click File → Import

Select General → File System

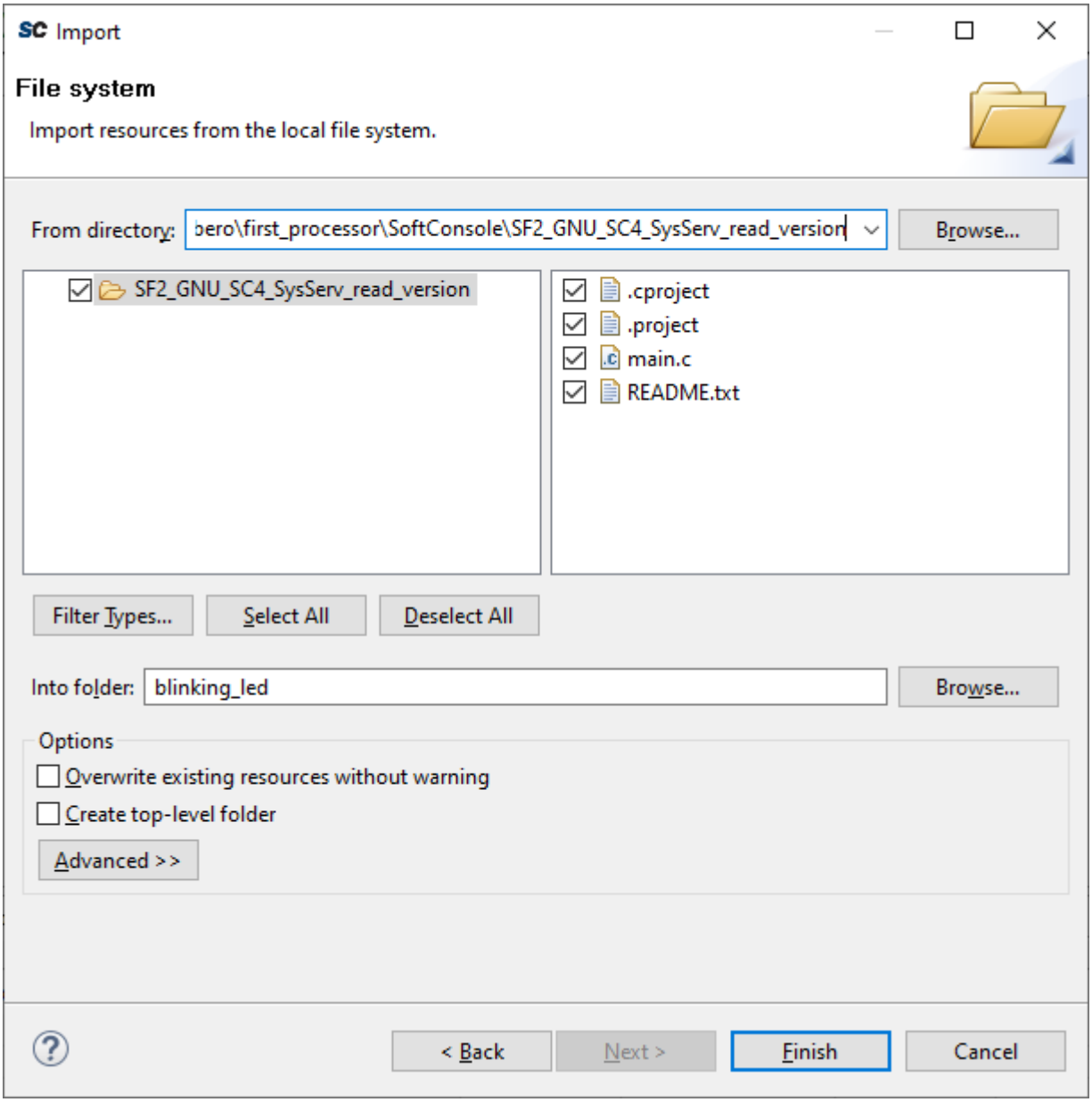
Click Next



Browse to your project's SoftConsole\SF2_GNU_SC4_SysServ_read_version\ subfolder

Apply check boxes like example below

Click Finish



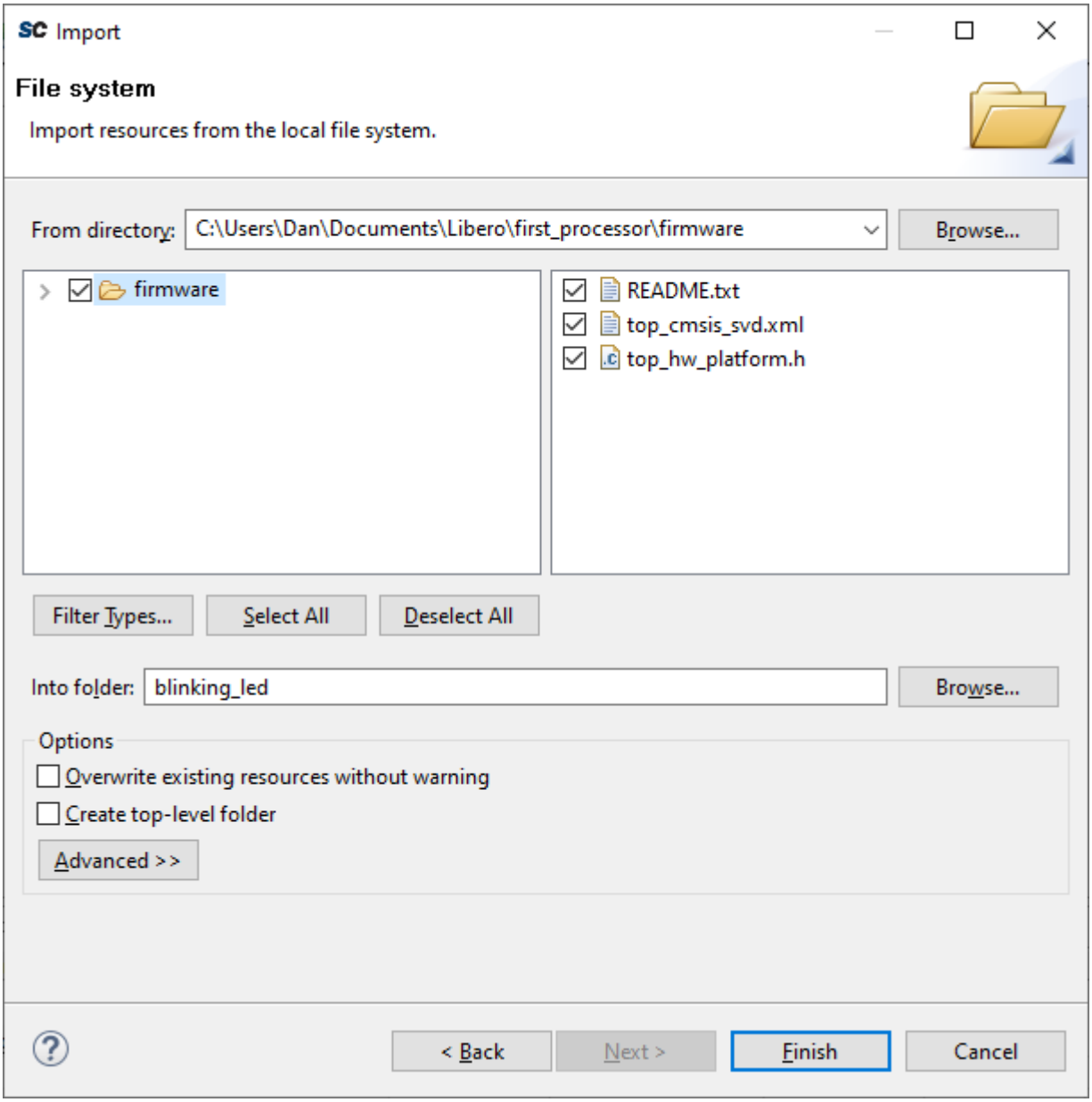
When prompted to overwrite click Yes to All

Click File → Import

Browse to your project's \firmware\ subfolder

Apply check boxes like example below

Click Finish



When prompted to overwrite click Yes to All

Double click main.c to open the file

Overwrite existing code with code on following page

```
#include <stdio.h>

#include <unistd.h>

#include "drivers/mss_sys_services/mss_sys_services.h"

#include "drivers/mss_uart/mss_uart.h"

#include "drivers/mss_gpio/mss_gpio.h"

#include "drivers/mss_timer/mss_timer.h"

//Function Declarations

void timer_delay(uint32_t load_value);

/*==============================================================================

  Main function.

 */

int main(){

        uint32_t load_value= 100000000;

        //initialize GPIO and timer

        MSS_GPIO_init();

        MSS_TIM1_init(MSS_TIMER_ONE_SHOT_MODE);


        while(1){

                //set GPIO high

                MSS_GPIO_set_output(MSS_GPIO_0, MSS_GPIO_DRIVE_LOW);

                //delay

                timer_delay(load_value);

                //set GPIO low

                MSS_GPIO_set_output(MSS_GPIO_0, MSS_GPIO_DRIVE_HIGH);

                timer_delay(load_value);

                //delay

        };

        return 0;

}

void timer_delay(uint32_t load_value){

        MSS_TIM1_load_immediate(load_value);

        MSS_TIM1_start();

        while (MSS_TIM1_get_current_value()); //while counter has not reached 0

}
```

Save the main.c file by clicking file → Save

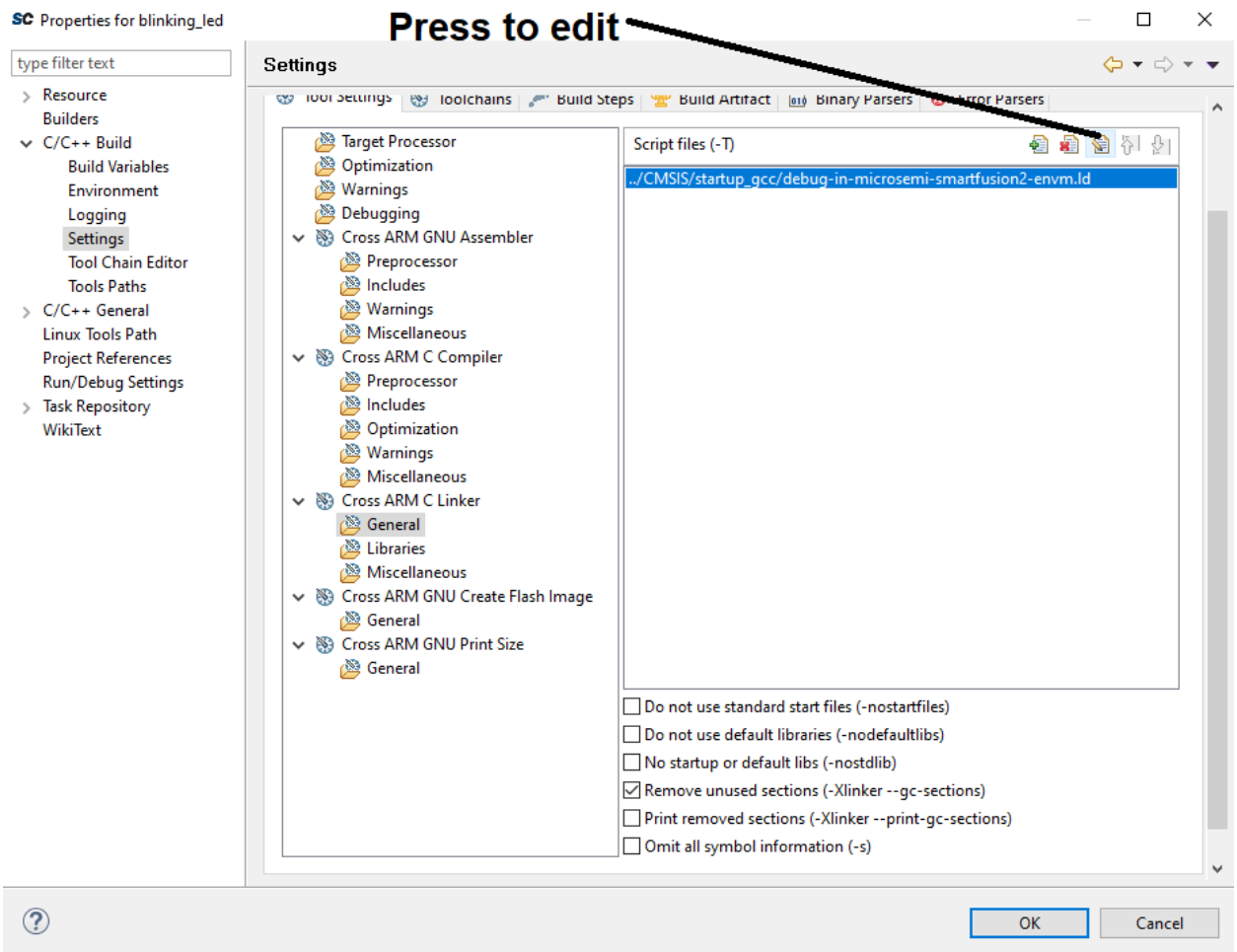
Click Project → Build Configurations → Set Active → Release

Click Project → Properties

Select C/C++ Build → Settings

In Tool Settings tab select Cross Arm C Linker → General

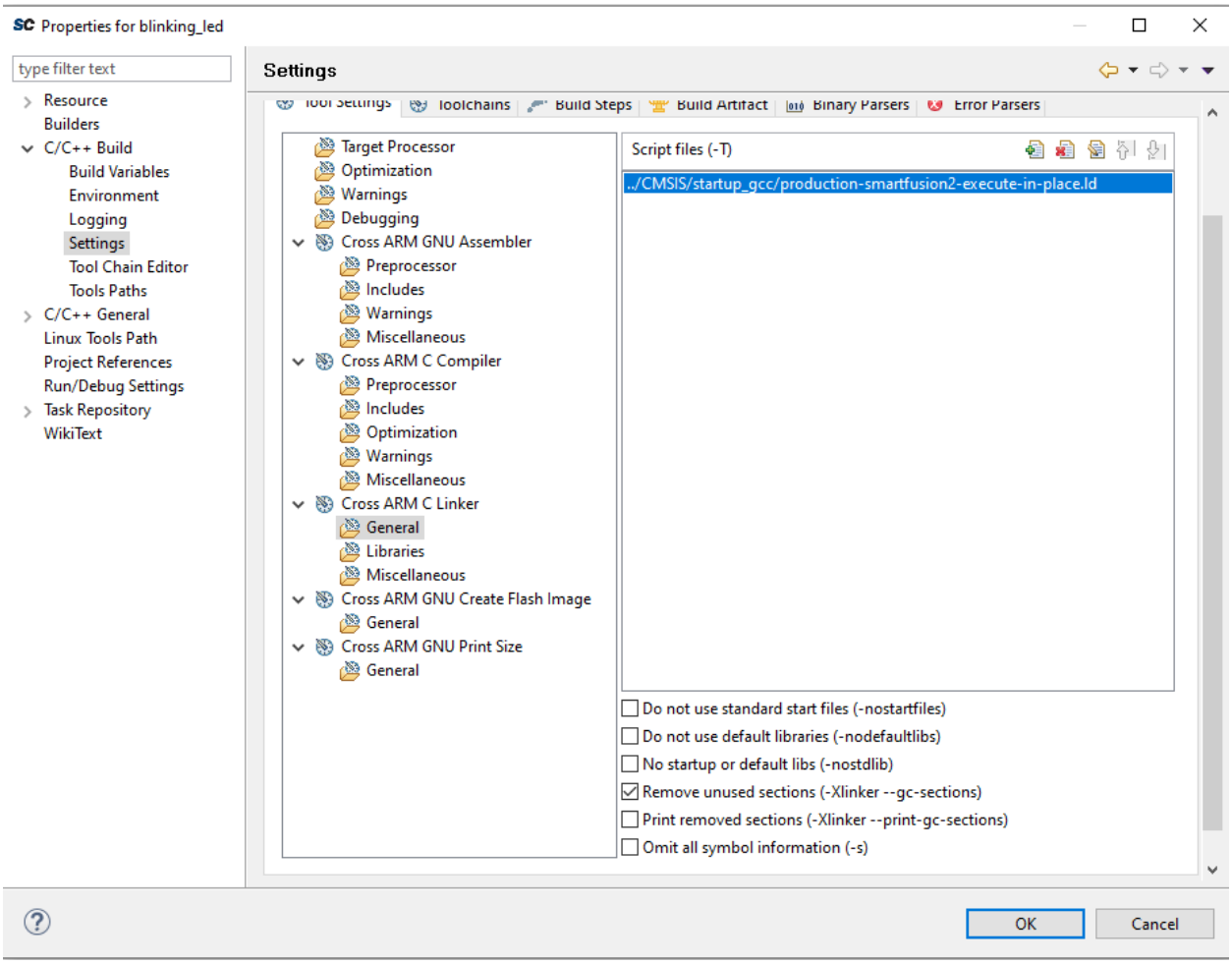
Click edit



Change the existing text to:

../CMSIS/startup_gcc/production-smartfusion2-execute-in-place.ld

Click OK

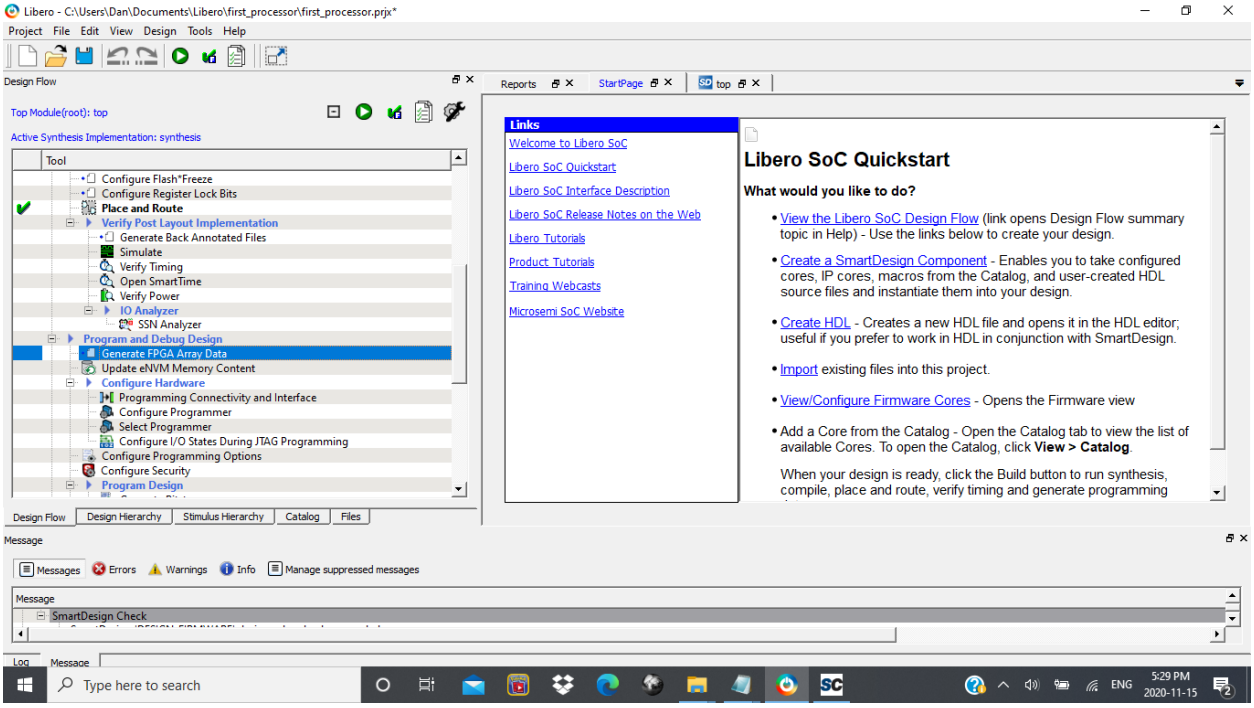


Click Project--> Build All

Go back to Libero SoC

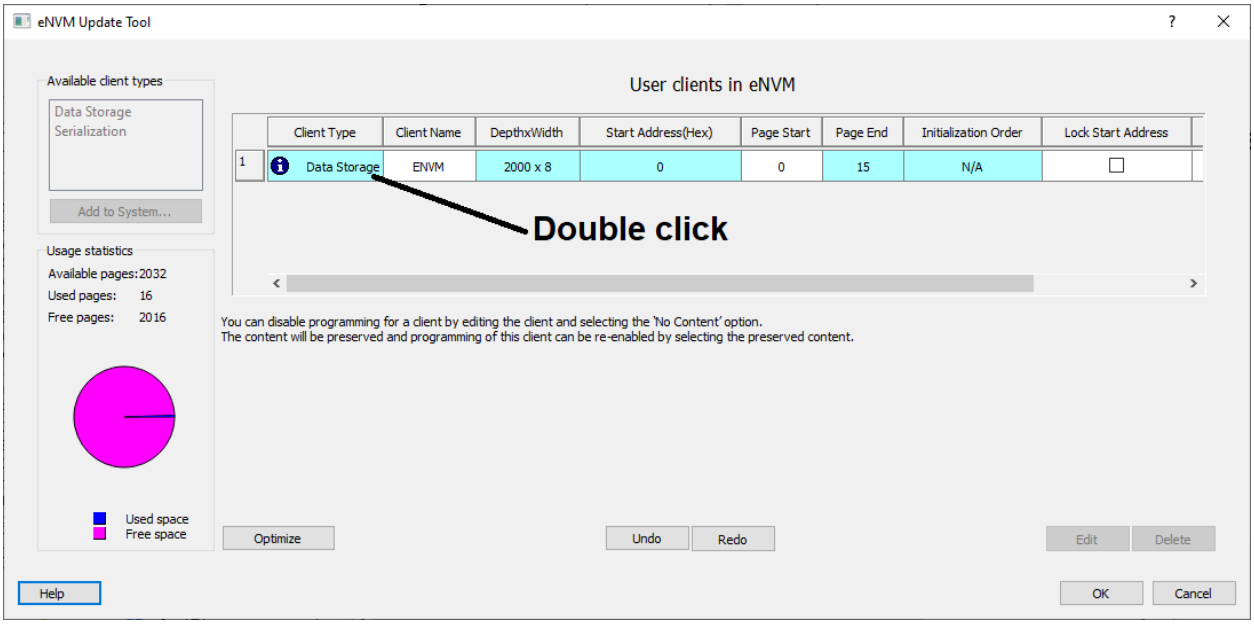
Double click Generate FPGA Array Data

If an information box pops up click OK



Double Click Udate eNVM Memory Content

Double Click Data Storage

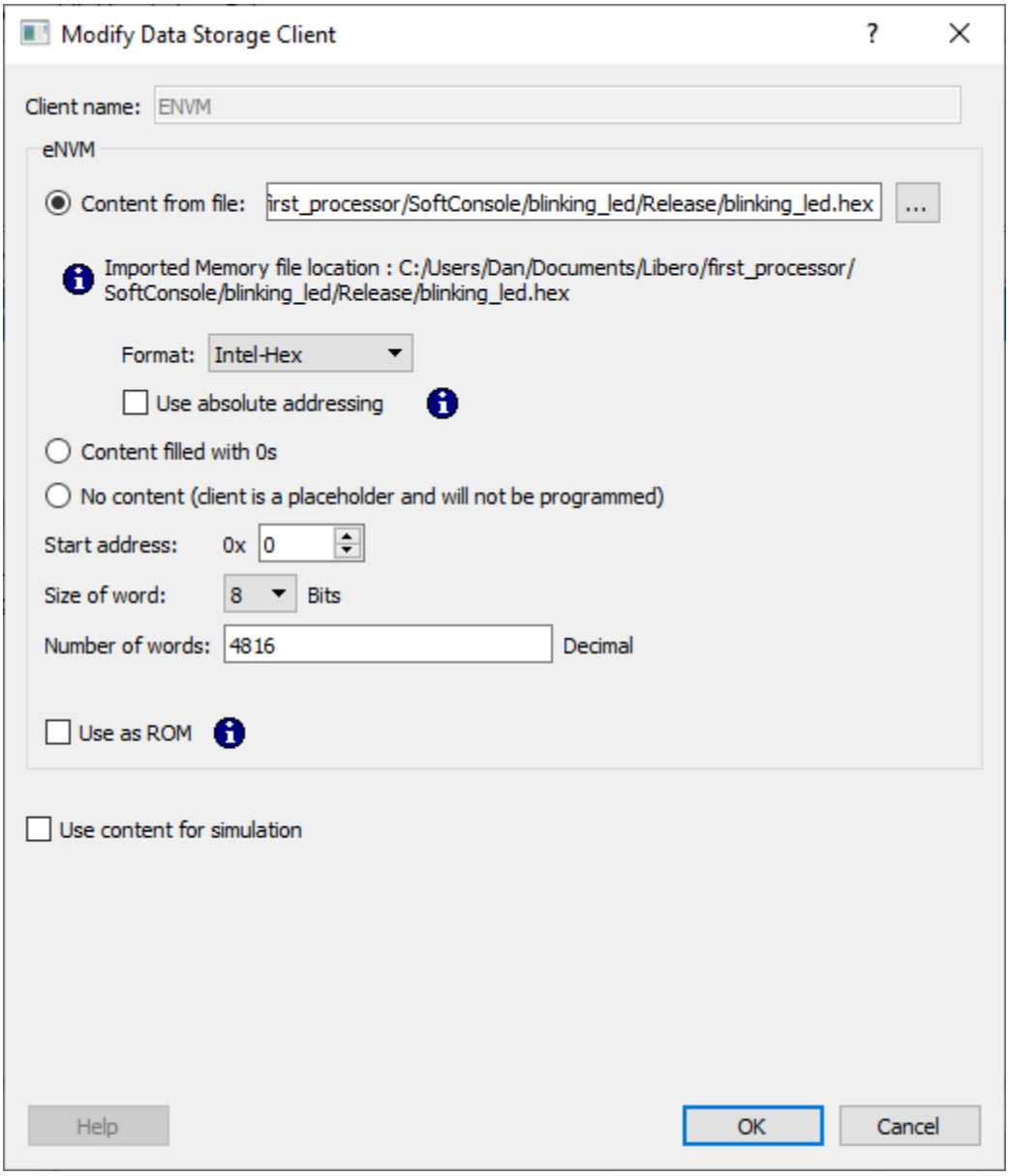


Select Content from file option

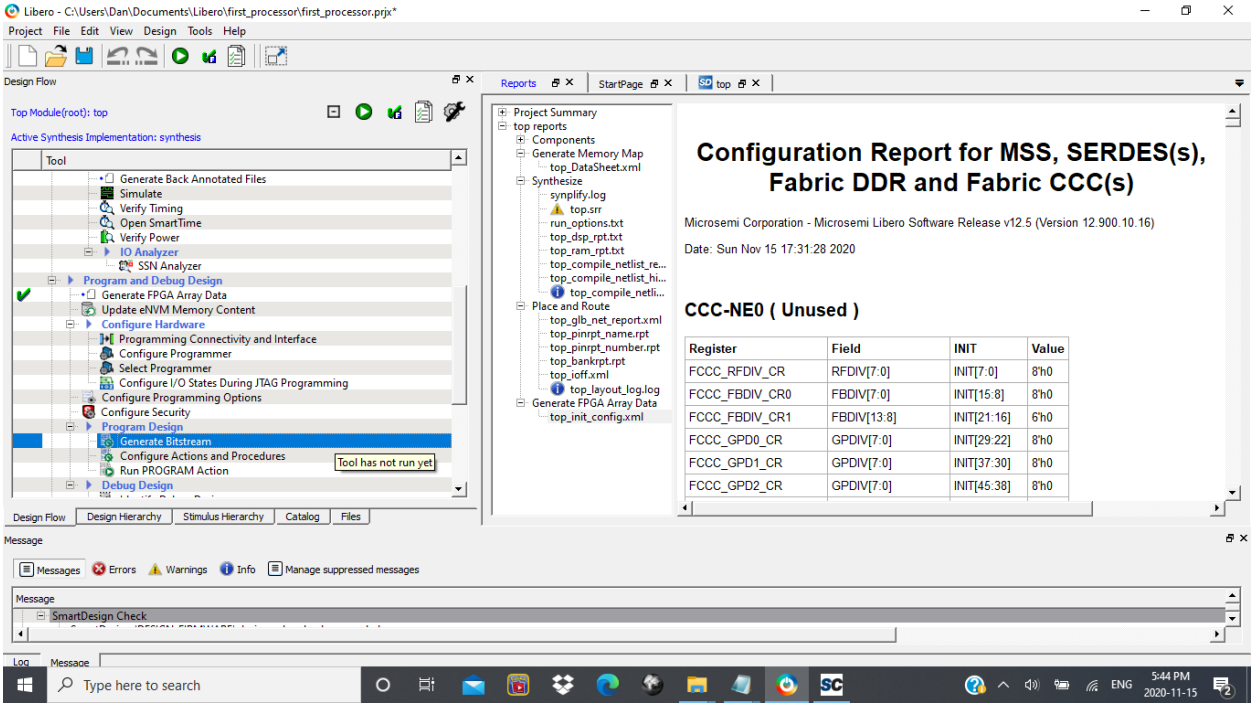
Choose the blinking_led.hex file from your project's \SoftConsole\blinking_led\Release\ subfolder

Refer to following image for other settings

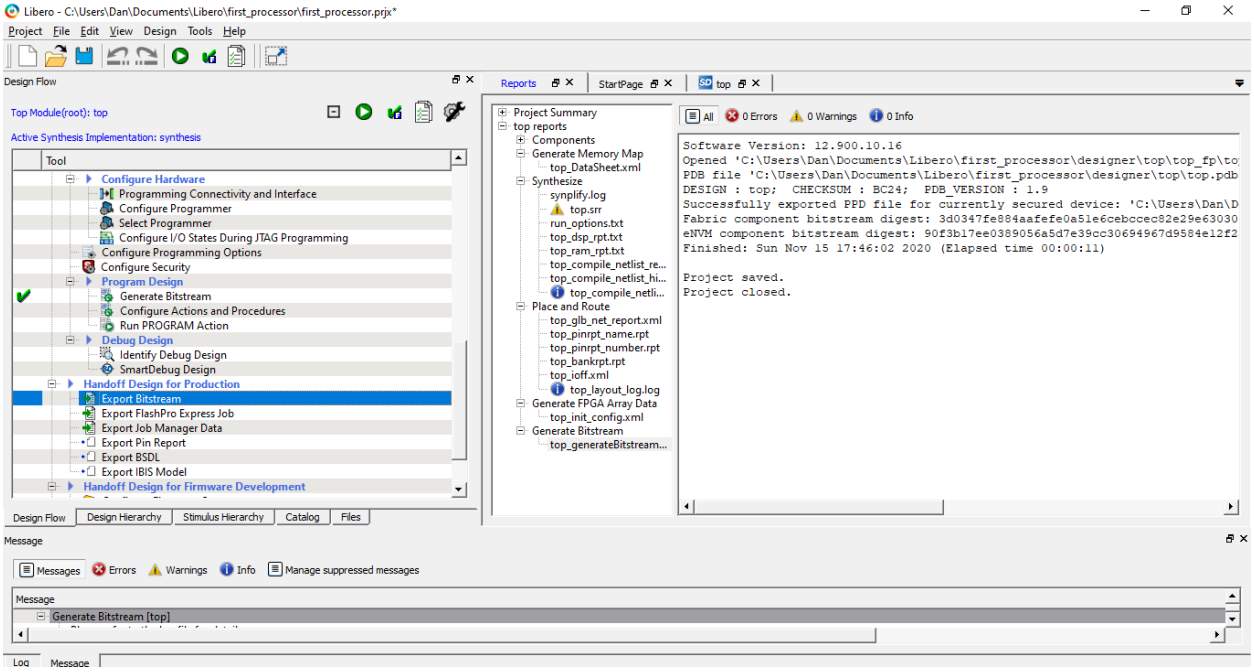
Press OK



Double click Generate Bitstream

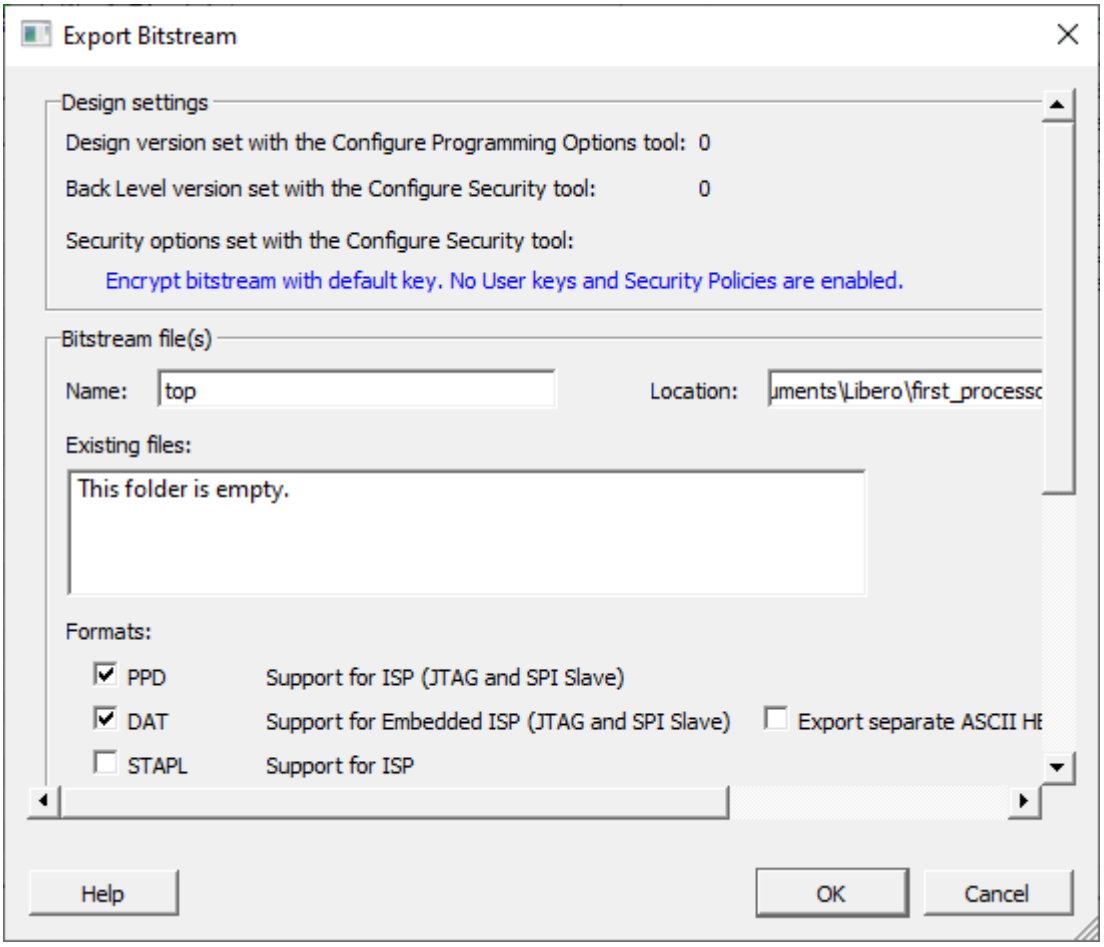


Double click Export Bitstream

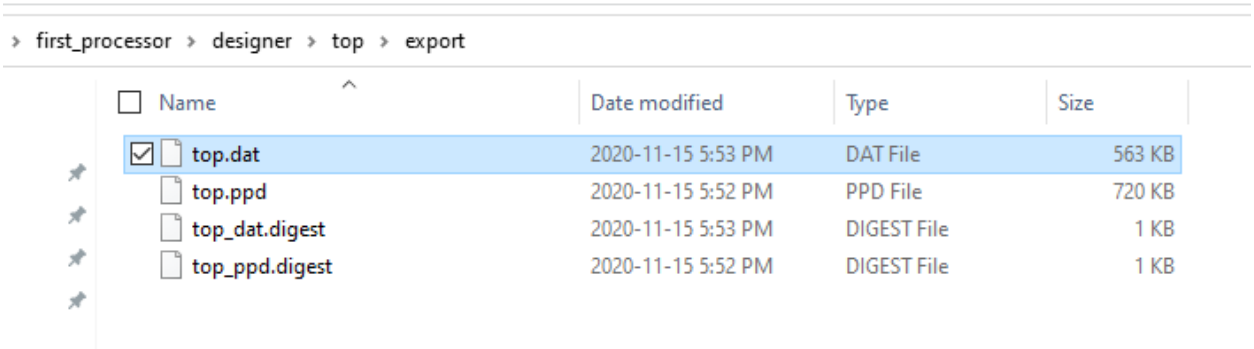


Ensure that PPD and DAT formats are selected

Click OK



The generated .DAT file can be found in your project's \designer\top\export\ subfolder

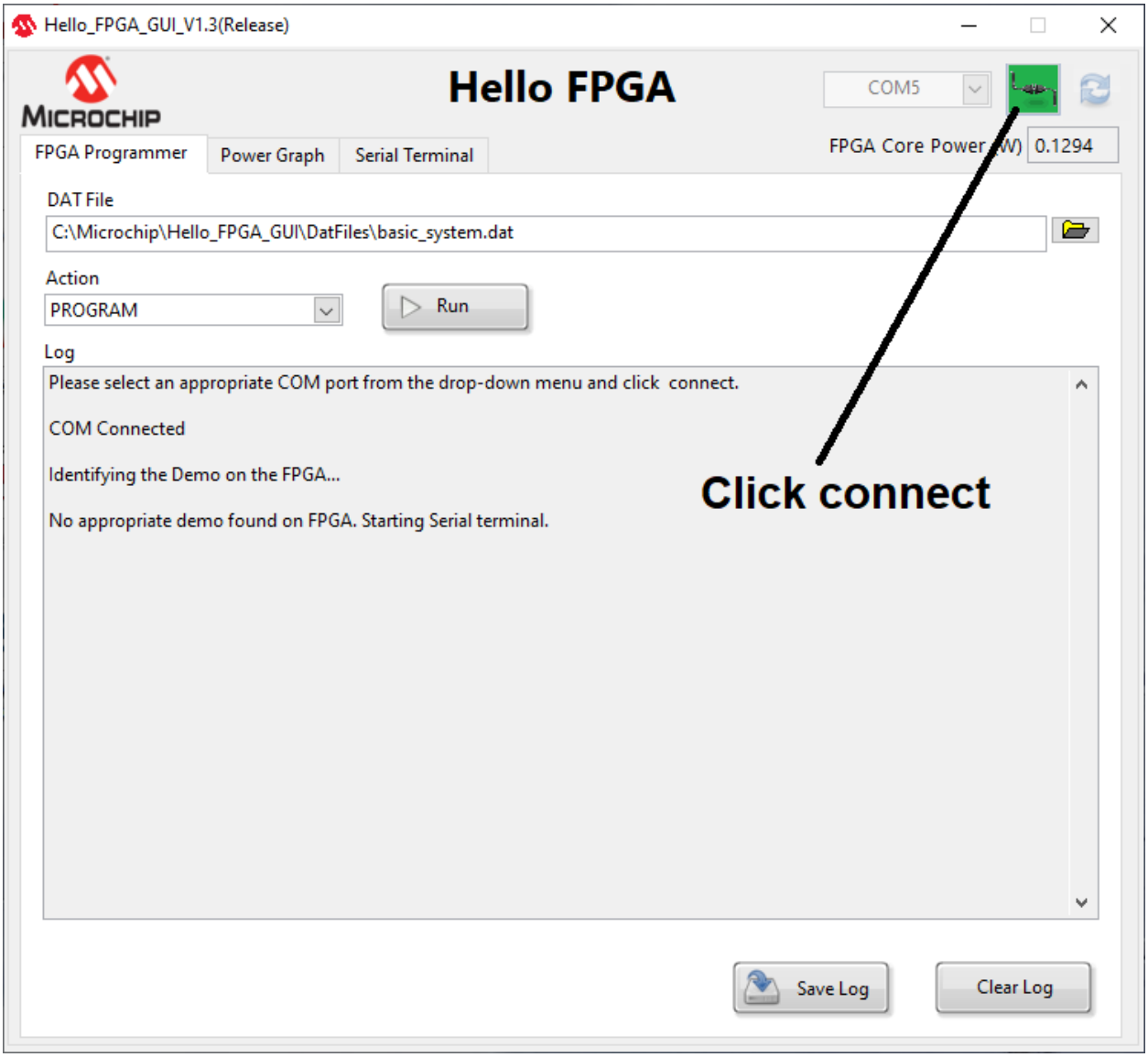


(Assuming that you installed the Hello FPGA GUI in its default location) I recommend that you copy top.DAT to C:\Microchip\Hello_FPGA_GUI\DatFiles

Connect the Hello FPGA Development board to your computer using the USB cable included with the kit

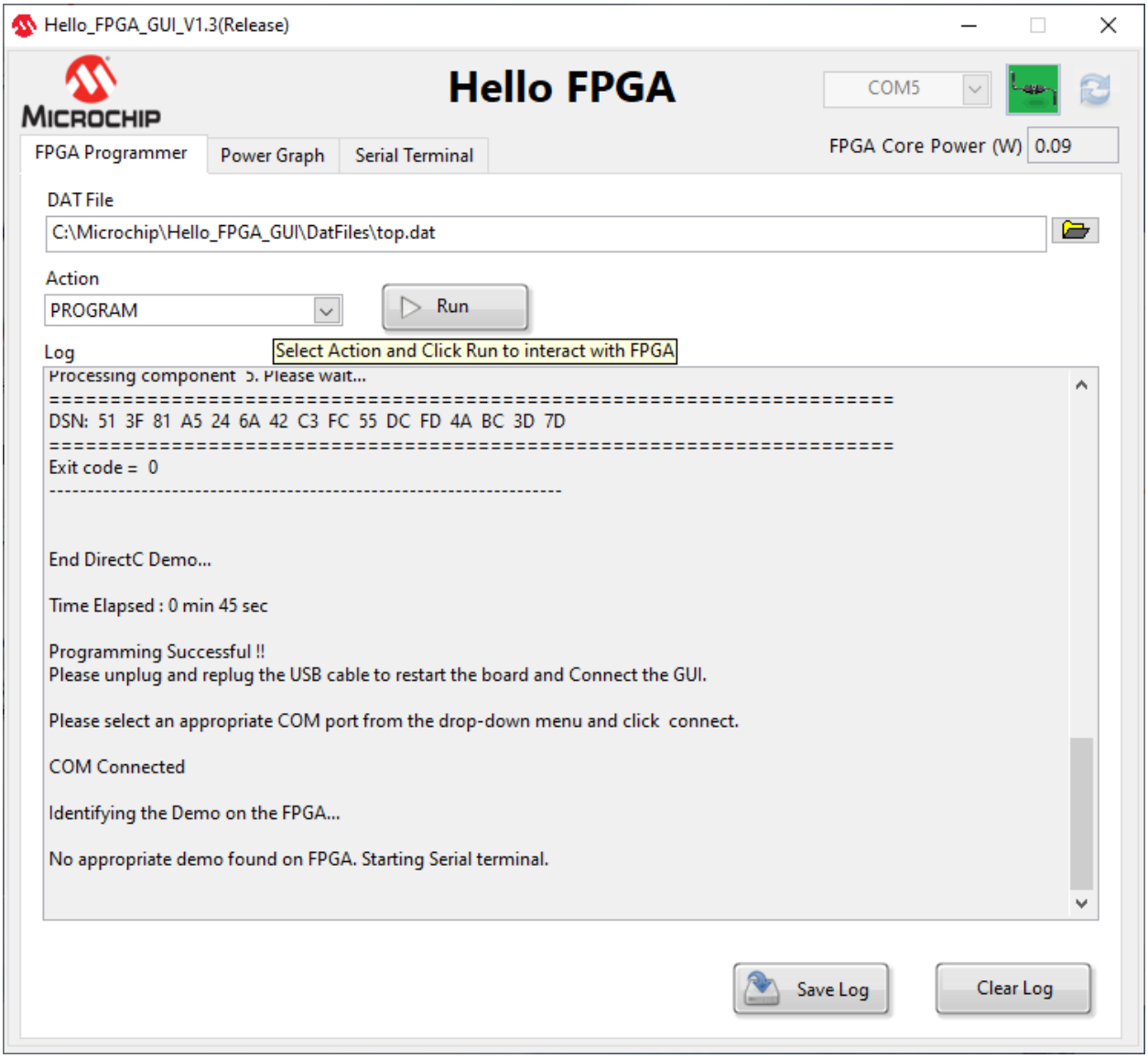
Open the Hello FPGA GUI Software

Click the connect graphic



Choose the top.DAT file

Click Run



Unplug and then reconnect the USB cable

LED1 should now be flashing (toggling between on and off every second)

Congratulations on making your first design targeting the Hello FPGA Design Kit