Lab 1 - Implement ACTIVE, IDLE, STANDBY, OFF power mode

Purpose:

After completing Lab 1, you will understand how to use low power modes in SAML10 device which can be used to optimize the power consumption of any application. We will also analyze the SAM L10 Xplained Pro board's power consumption using Data Visualizer tool in MPLAB X IDE.

Overview:

The goal of this hands on is to describe and illustrate the different low power modes available in the SAML10 device and it will also demonstrate various low power techniques which can be used to optimize the power consumption of any application:

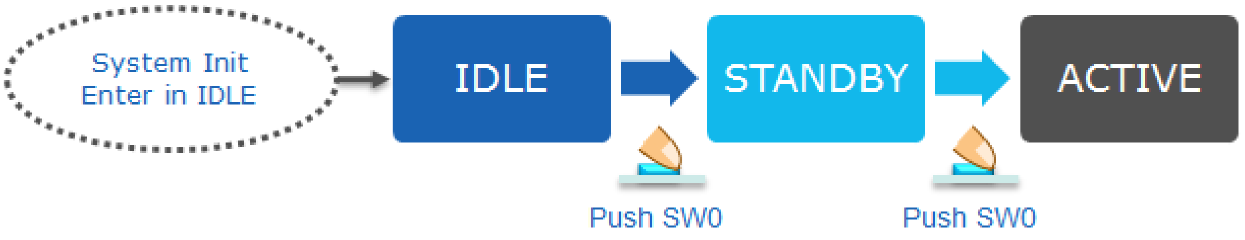
- Flexible clock architecture
- Regulators architecture with embedded BUCK converter
- Dynamic voltage scaling using SAM L10 Performance Levels
- Clock and Power domain gating

The application uses:

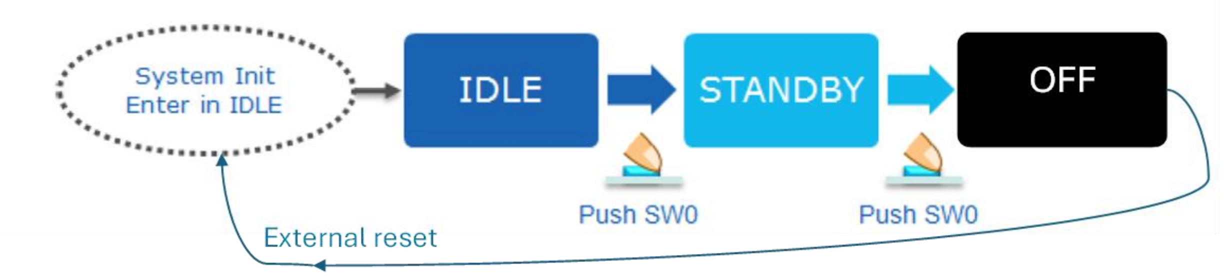
- EIC to control the user button SW0
- PM and Supply Controller (SUPC) peripheral libraries to configure Low Power modes.

Flowchart

Lab 1.a



Lab 1.b



Procedure:

This lab is completed in the following steps:

Lab 1.a

Part 1: Create project and configure SAML10

STEP 1: Install the MPLAB® Code Configurator (MCC) Plugin in the MPLAB X IDE

STEP 2: Create MCC Harmony Project using MPLAB X IDE

Part 2: Configure Oscillators and the clocks, Add and configure External interrupt controller peripheral

STEP 3: Configure clock peripheral

STEP 4: Configure EIC and select pins

Part 3: Implement active idle and standby mode with PM, SUPC and NVMCTRL configurations.

STEP 5: Configure PM, SUPC and NVMCTRL as required

STEP 6: Generate code

Part 4: Add code and observe output.

STEP 7: Add code to your application

STEP 8: Build and Run the Application

Lab 1.b

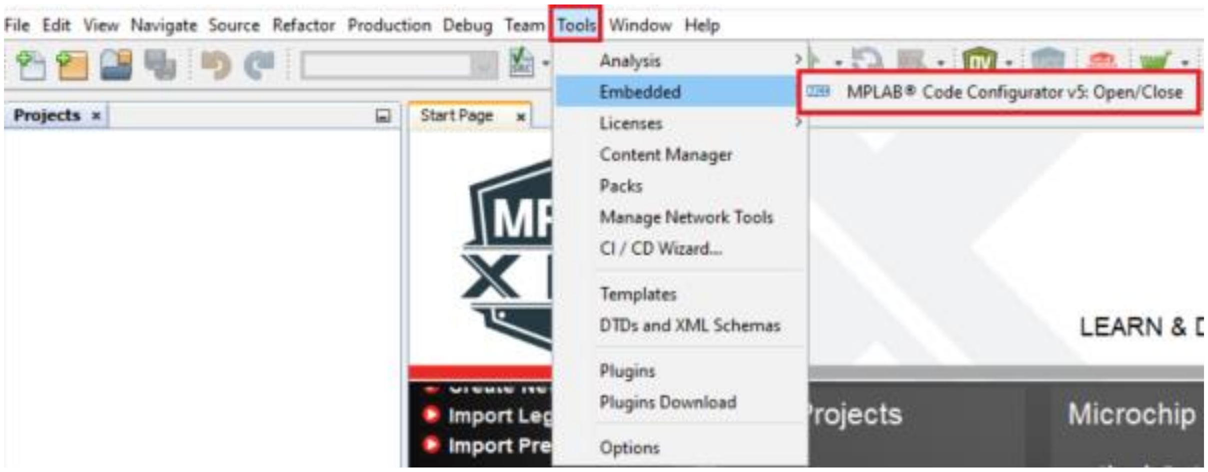
Part 5: Add code for OFF mode and observe output.

STEP 9: Add code to your application, build and run

Procedure:

STEP 1: Install the MPLAB® Code Configurator (MCC) Plugin in the MPLAB X IDE

- Launch MPLAB X IDE from the Windows® Start Menu. Close any projects and files that are currently open
- Go to **Tools > Embedded**.
- You will see MPLAB® Code Configurator in the menu. If you don't see it in the menu, please **Install MCC**



Note:

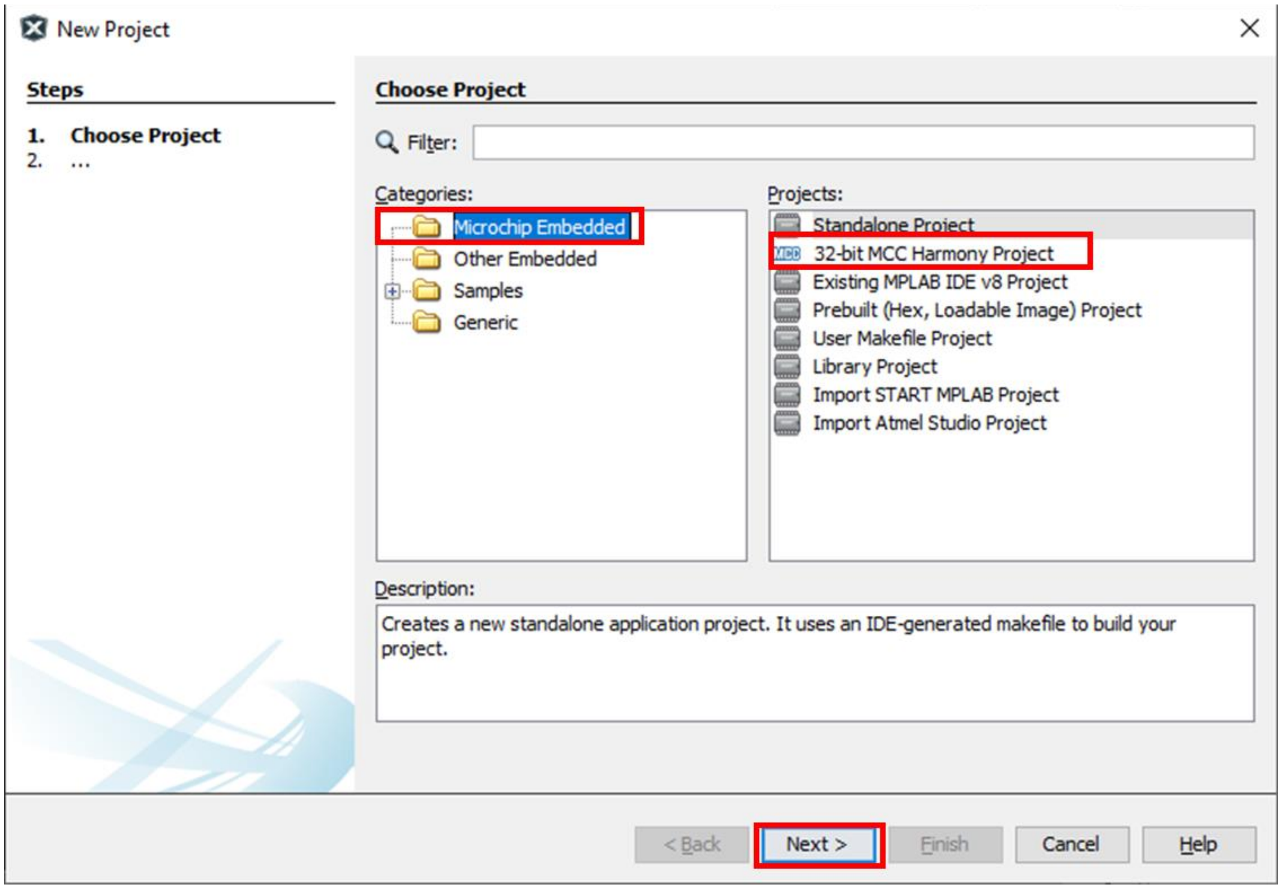
- When you launch MCC for the first time, it displays a prompt asking for the mode in which you would like to use MCC.
    - Standalone mode (as a separate window)
    - Native mode (as an embedded window in MPLAB X IDE)
- Standalone mode is the default mode.
- MCC will launch in the operating mode selected the first time, every time you launch MCC.
- For this lab, MCC is configured to operate in Standalone mode.

STEP 2: Create MPLAB Harmony Project using MPLAB X IDE

Step 2a: Close any currently Opened project in MPLAB IDE

Step 2b: Select ***File > New Project*** from the main IDE menu.

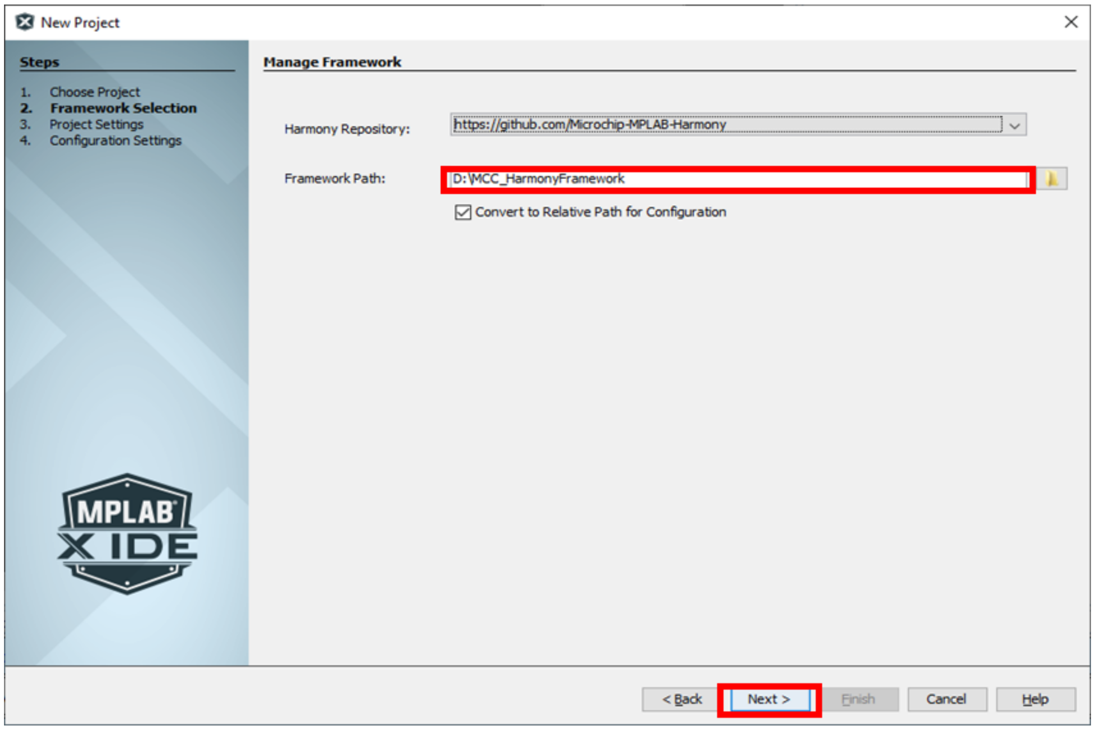
Step 2c: In the Categories pane of the New Project dialog, select **Microchip Embedded**. In the Projects pane, select **32-bit MCC Harmony Project**, and then click **Next.**



Creates a new standalone application project. It uses an IDE-generated makefile to build your project.

Note: If "32-Bit MCC Harmony Project" is not displayed, please download and install the MCC before continuing with further steps. Refer Step 1 for details.

Step 2d: In "Framework Path" edit box, enter the path to the folder in which MPLAB Harmony 3 packages are downloaded. For Example: **"D:\microchip\harmony_3".**



Harmony Repository: https://github.com/Microchip-MPLAB-Harmony

Framework Path: D:\MCC_HarmonyFramework

☑ Convert to Relative Path for Configuration

Step 2e: In the New Project window, perform the following settings:

- *Location*: Indicates the path to the root folder of the new project. All project files will be placed inside this folder. The project location can be any valid path, for example:

  "D:\microchip\harmony_3\lab_work\labs".

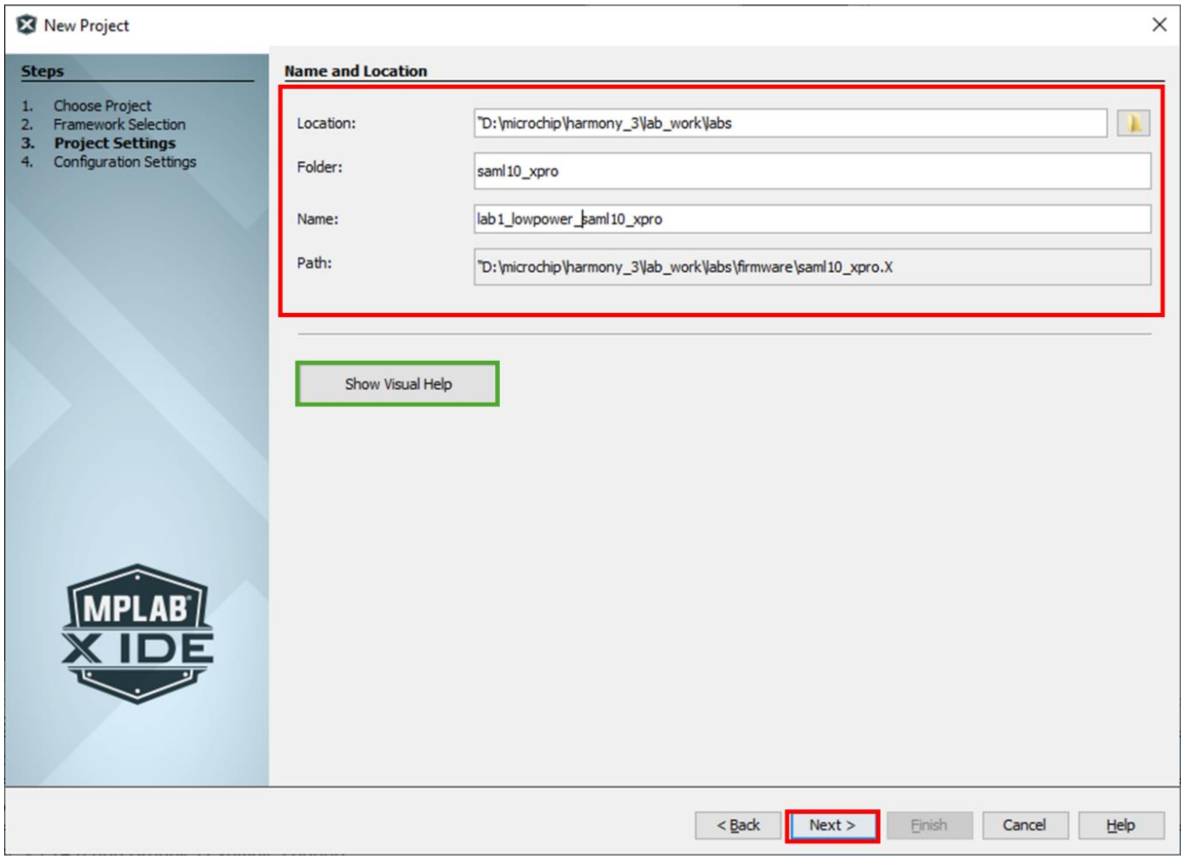
- *Folder*: Indicates the name of the MPLABX .X folder. Enter "**sam_l10_xpro**" to create a sam_l10_xpro.X folder.

  Note: This must be a valid directory name for your operating system.

- *Name*: Enter the project's logical name as **"lab1_lowpower_sam_l10_xpro.X"**. This is the name that will be shown from within the MPLAB X IDE.

Note: The *Path* box is read-only. It will update as you make changes to the other entries.

- Click **Next** to proceed to Configuration Settings.



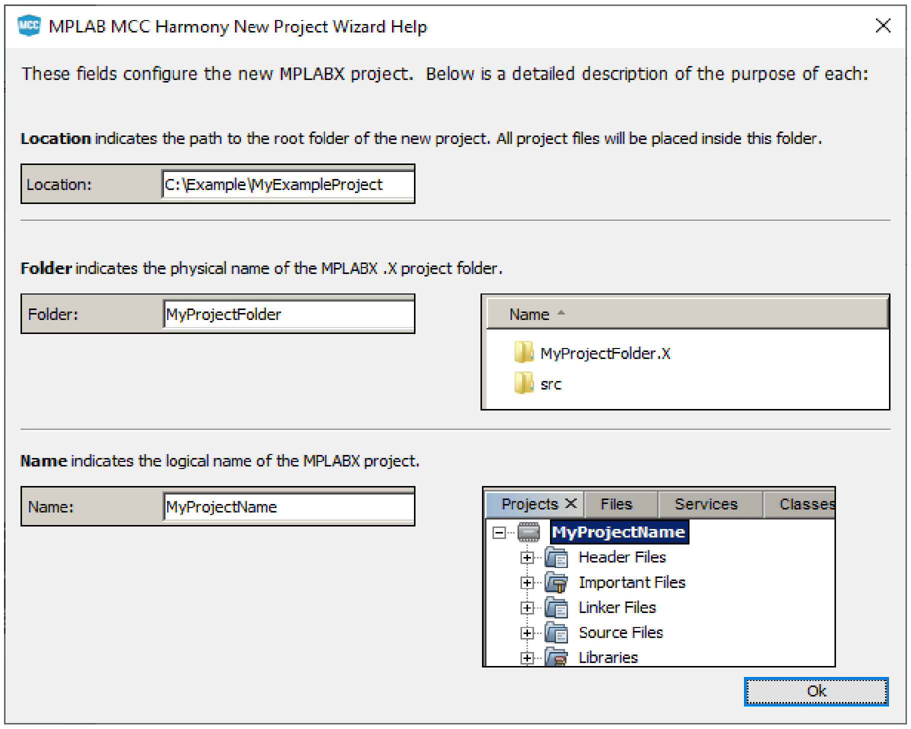
Location: "D:\microchip\harmony_3\lab_work\labs

Folder: saml10_xpro

Name: lab1_lowpower_saml10_xpro

Path: "D:\microchip\harmony_3\lab_work\labs\firmware\saml10_xpro.X

Note: Clicking on the "Show Visual Help" button will open a help window providing a detailed description of the various fields in the *Project Settings* window.



These fields configure the new MPLABX project. Below is a detailed description of the purpose of each:

Location indicates the path to the root folder of the new project. All project files will be placed inside this folder.

Location: C:\Example\MyExampleProject

Folder indicates the physical name of the MPLABX .X project folder.

Folder: MyProjectFolder

Name indicates the logical name of the MPLABX project.

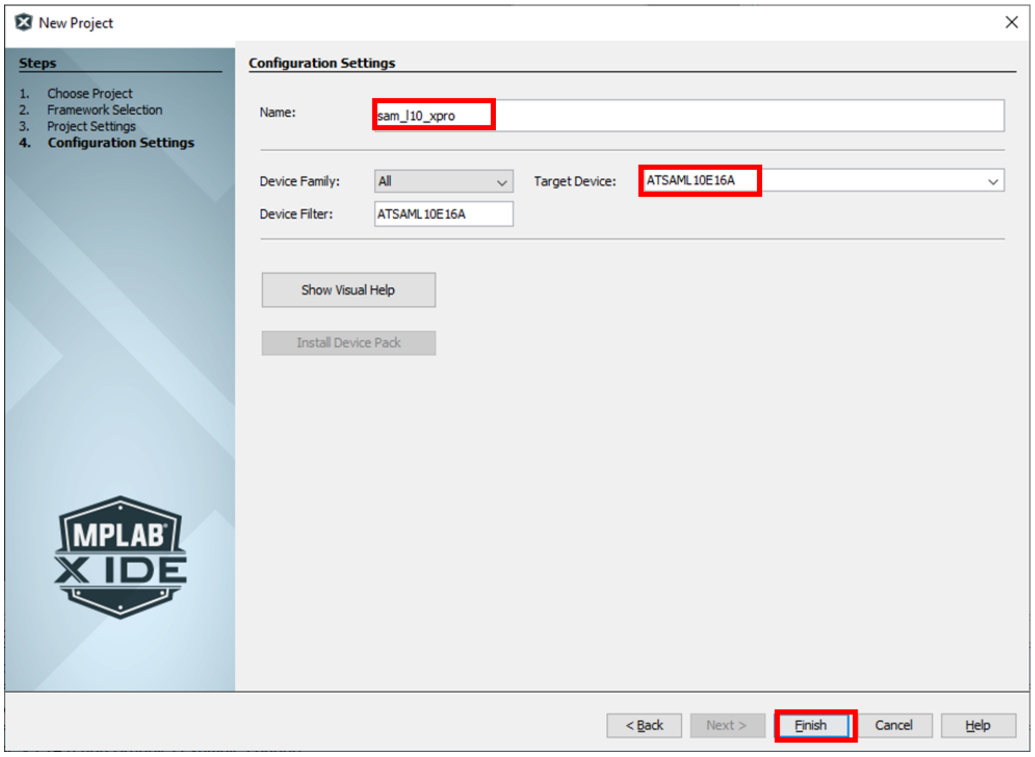
Name: MyProjectName

Step 2f: Follow the below steps to setup the project's *Configuration Settings*

- *Name*: Enter the configuration name as "**sam_l10_xpro**".
- *Target Device*: Select "**ATSAML10E16A**" as the target device.

Note: You can select the Device Family or enter a partial device name to filter the list in Target Devices in order to make it easier to find the desired device.

- After selecting the target device, click **Finish** to launch the project



Name: sam_l10_xpro

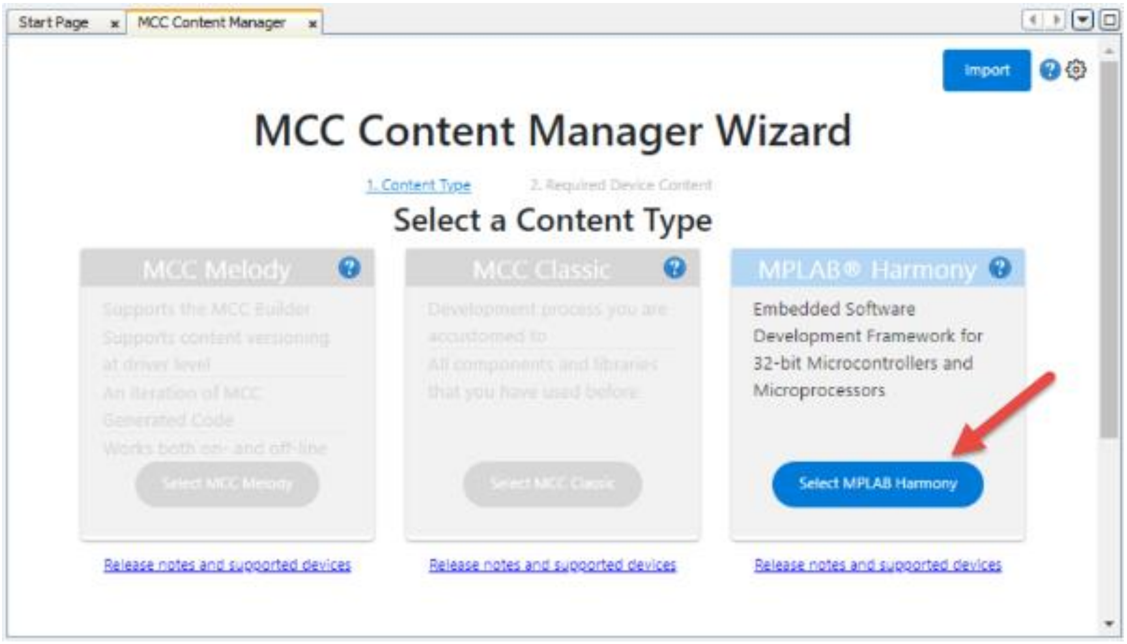
Device Family: All

Target Device: ATSAML10E16A

Device Filter: ATSAML10E16A

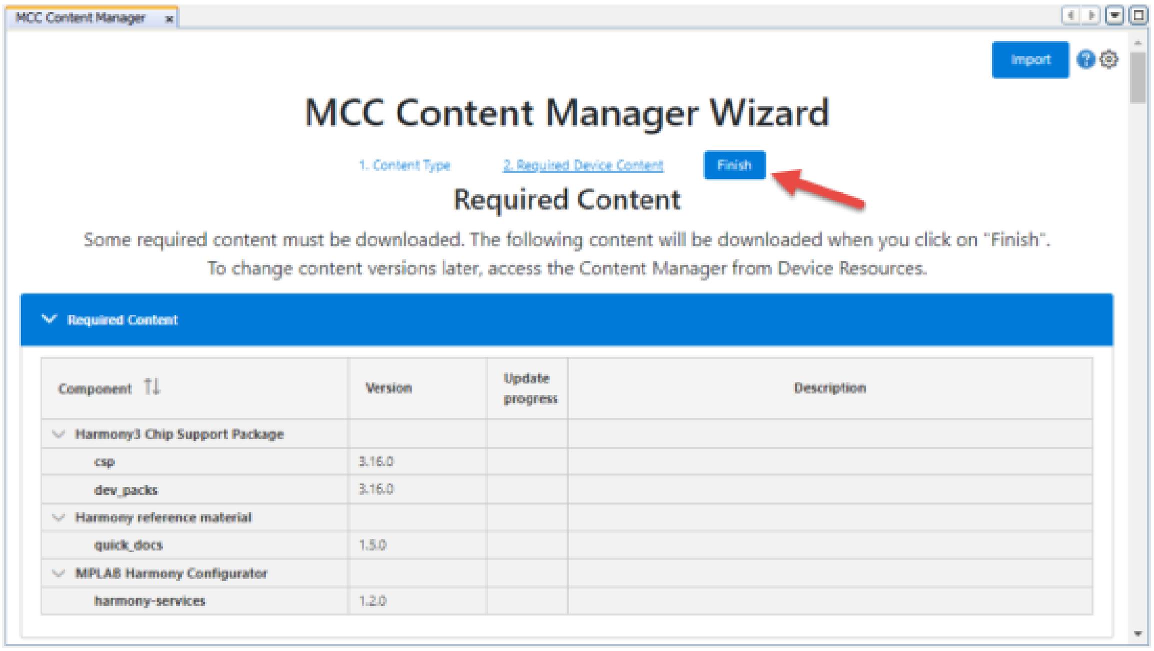
Step 2g: Download the MPLAB Harmony framework (if using the first time)

When the project opens, the MCC Content Manager Wizard window will be displayed. Click the **Select MPLAB Harmony** button to select which MPLAB Harmony framework components you want to download.


Start Page
MCC Content Manager
Import
MCC Content Manager Wizard
1. Content Type
2. Required Device Content
Select a Content Type
MCC Melody
MCC Classic
MPLAB® Harmony
Embedded Software Development Framework for 32-bit Microcontrollers and Microprocessors
Select MPLAB Harmony
Release notes and supported devices

The MCC Content Manager Wizard will compare the Harmony Repository with the Framework Path (as specified above), and provide you with a list of components you may want to download. This list will include updated components and never-downloaded components.

If this is your first time downloading the framework or if you've chosen a new framework path, the required components will already be selected for you. To keep your first Harmony project simple, just click **Finish** to download only the required components (about 5 GB):



Some required content must be downloaded. The following content will be downloaded when you click on "Finish".
To change content versions later, access the Content Manager from Device Resources.

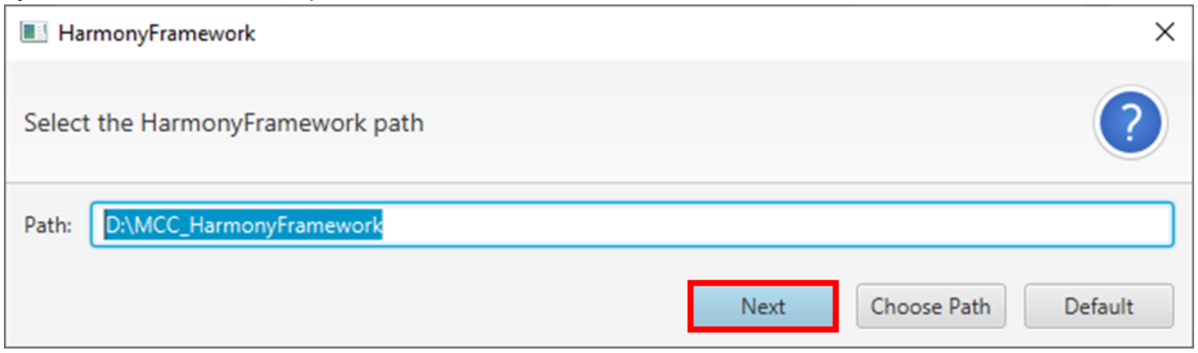
Required Content

Component	Version	Update progress	Description
Harmony3 Chip Support Package			
csp	3.16.0		
dev_packs	3.16.0		
Harmony reference material			
quick_docs	1.5.0		
MPLAB Harmony Configurator			
harmony-services	1.2.0		

- Harmony 3 Chip Support Package
  - csp (Chip Support Package includes low-level Harmony Peripheral Libraries (PLIBs))
  - dev_packs (includes ARM CMSIS and Microchip Device Family Packs)
- Harmony reference material
  - quick_docs component provides Harmony help through standalone HTML pages
- MPLAB Harmony Configurator
  - harmony-services provides all Harmony plugins for the MPLAB X IDE

MCC will automatically start after the download (about 5 GB) completes.

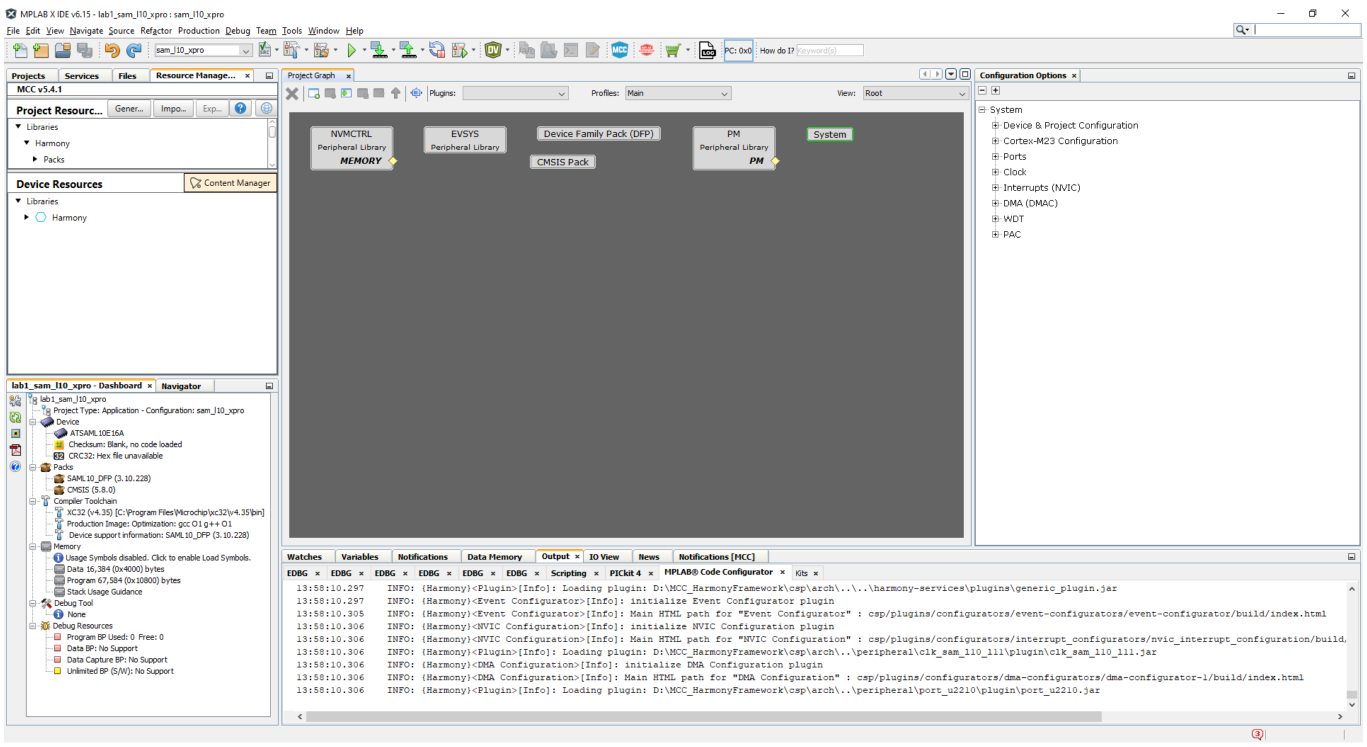
If already installed confirm the path as shown below


HarmonyFramework

Select the HarmonyFramework path

Path: D:\MCC_HarmonyFramework

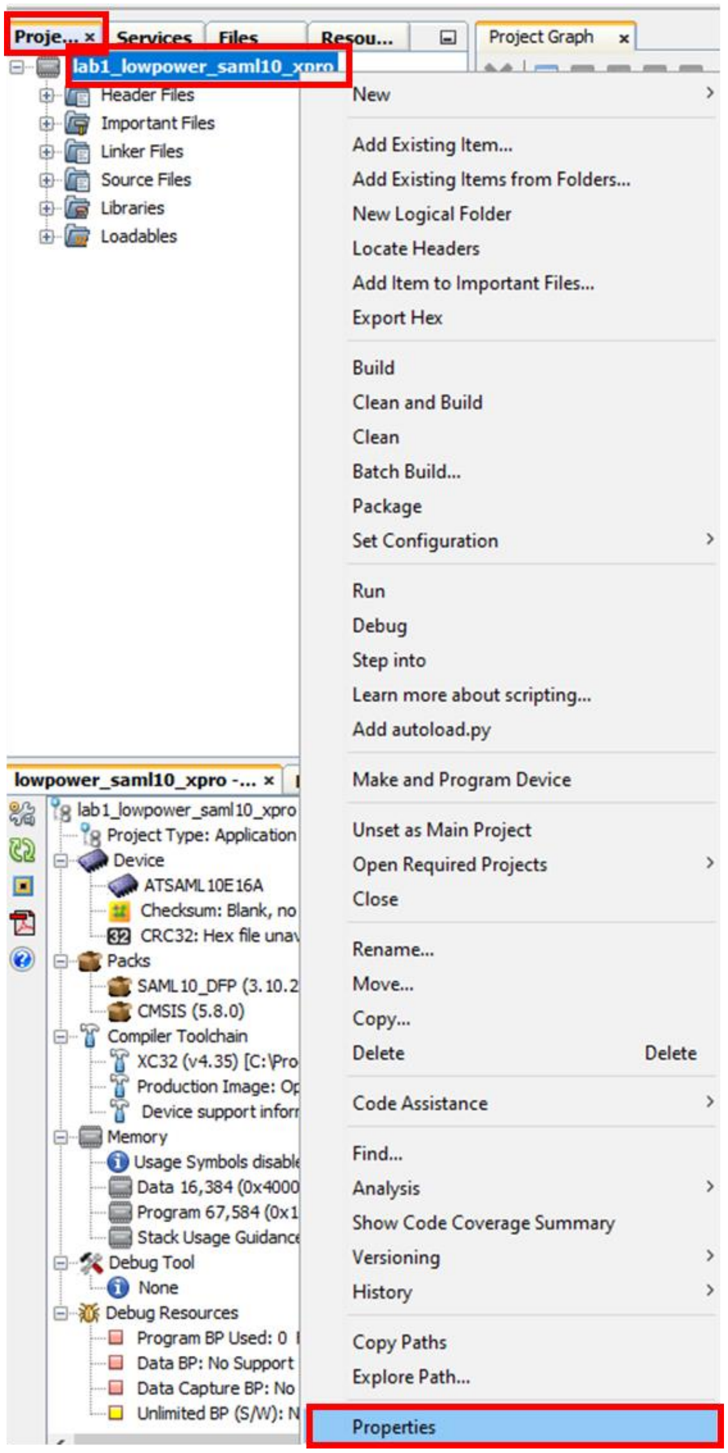
Step 2h: The MCC plugin's main window for the project will be displayed.



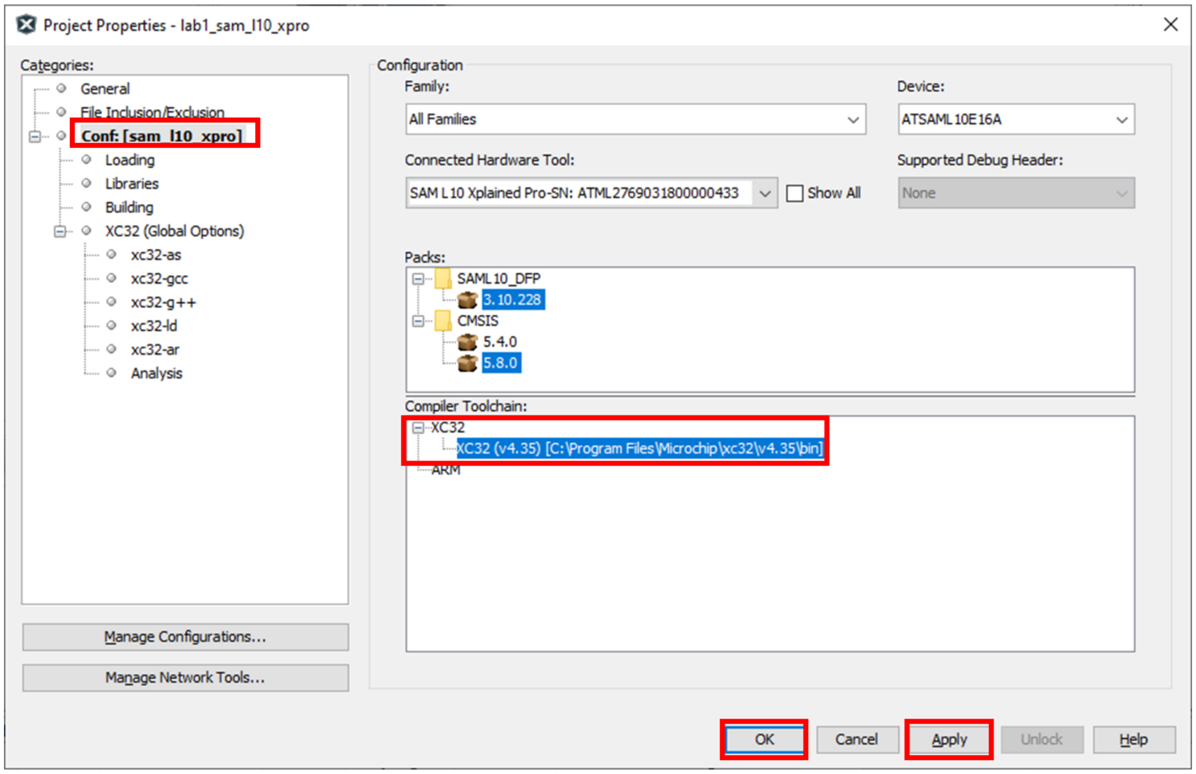
lab1_sam_l10_xpro
- Project Type: Application - Configuration: sam_l10_xpro
- Device
  - ATSAML10E16A
  - Checksum: Blank, no code loaded
  - CRC32: Hex file unavailable
- Packs
  - SAML10_DFP (3.10.228)
  - CMSIS (5.8.0)
- Compiler Toolchain
  - XC32 (v4.35) [C:\Program Files\Microchip\xc32\v4.35\bin]
  - Production Image: Optimization: gcc O1 g++ O1
  - Device support information: SAML10_DFP (3.10.228)
- Memory
  - Usage Symbols disabled. Click to enable Load Symbols.
  - Data 16,384 (0x4000) bytes
  - Program 67,584 (0x10800) bytes
  - Stack Usage Guidance
- Debug Tool
  - None
- Debug Resources
  - Program BP Used: 0 Free: 0
  - Data BP: No Support
  - Data Capture BP: No Support
  - Unlimited BP (S/W): No Support

```
13:58:10.297   INFO: {Harmony}<Plugin>[Info]: Loading plugin: D:\MCC_HarmonyFramework\csp\arch\..\..\harmony-services\plugins\generic_plugin.jar
13:58:10.297   INFO: {Harmony}<Event Configurator>[Info]: initialize Event Configurator plugin
13:58:10.305   INFO: {Harmony}<Event Configurator>[Info]: Main HTML path for "Event Configurator" : csp/plugins/configurators/event-configurators/event-configurator/build/index.html
13:58:10.306   INFO: {Harmony}<NVIC Configuration>[Info]: initialize NVIC Configuration plugin
13:58:10.306   INFO: {Harmony}<NVIC Configuration>[Info]: Main HTML path for "NVIC Configuration" : csp/plugins/configurators/interrupt_configurators/nvic_interrupt_configuration/build,
13:58:10.306   INFO: {Harmony}<Plugin>[Info]: Loading plugin: D:\MCC_HarmonyFramework\csp\arch\..\peripheral\clk_sam_l10_l11\plugin\clk_sam_l10_l11.jar
13:58:10.306   INFO: {Harmony}<DMA Configuration>[Info]: initialize DMA Configuration plugin
13:58:10.306   INFO: {Harmony}<DMA Configuration>[Info]: Main HTML path for "DMA Configuration" : csp/plugins/configurators/dma-configurators/dma-configurator-1/build/index.html
13:58:10.306   INFO: {Harmony}<Plugin>[Info]: Loading plugin: D:\MCC_HarmonyFramework\csp\arch\..\peripheral\port_u2210\plugin\port_u2210.jar
```

Step 2i: Before proceeding, setup the compiler toolchain. Click on the "Projects" tab on the top left pane. Right click on the project name "**lab1_lowpower_sam_l10_xpro**" and go to "**Properties**".



Make sure that **XC32 (v4.35)** is selected as the Compiler Toolchain.
Click on **Apply** and then click on **OK.**


Project Properties - lab1_sam_l10_xpro

Categories:
- General
- File Inclusion/Exclusion
- Conf: [sam_l10_xpro]
  - Loading
  - Libraries
  - Building
  - XC32 (Global Options)
    - xc32-as
    - xc32-gcc
    - xc32-g++
    - xc32-ld
    - xc32-ar
    - Analysis

Configuration
Family: All Families
Device: ATSAML10E16A
Connected Hardware Tool: SAM L10 Xplained Pro-SN: ATML2769031800000433
Show All
Supported Debug Header: None
Packs:
- SAML10_DFP
  - 3.10.228
- CMSIS
  - 5.4.0
  - 5.8.0

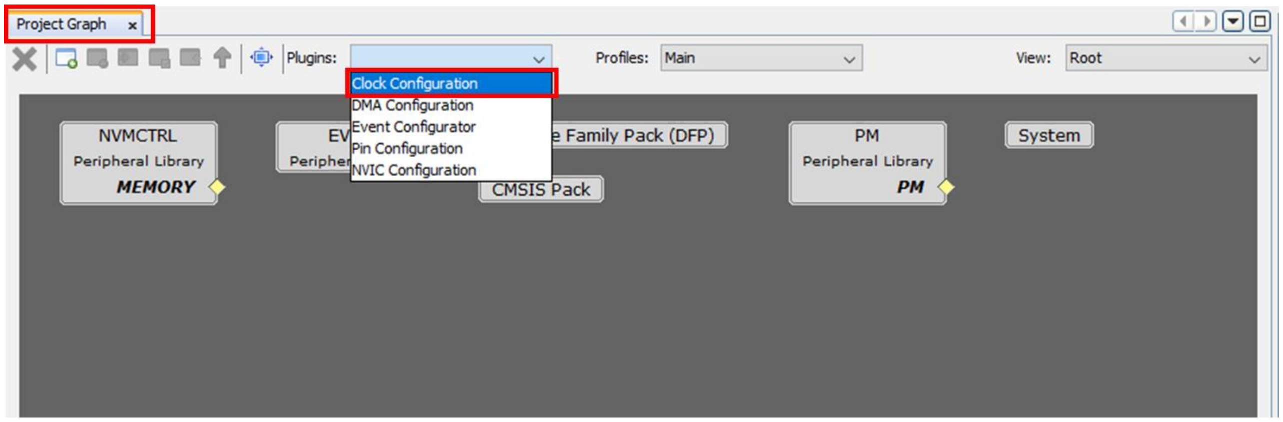
Compiler Toolchain:
- XC32
  - XC32 (v4.35) [C:\Program Files\Microchip\xc32\v4.35\bin]
- ARM

Tool Tip: - If you closed the Project graph accidently, and would like to open it again, go to *Tools* ▶ *Embedded* ▶ *MPLAB Code Configurator* v5:Open/Close

Part 2: Configure Oscillators and the clocks, Add and configure External interrupt controller peripheral

STEP 3: Configure clock peripheral

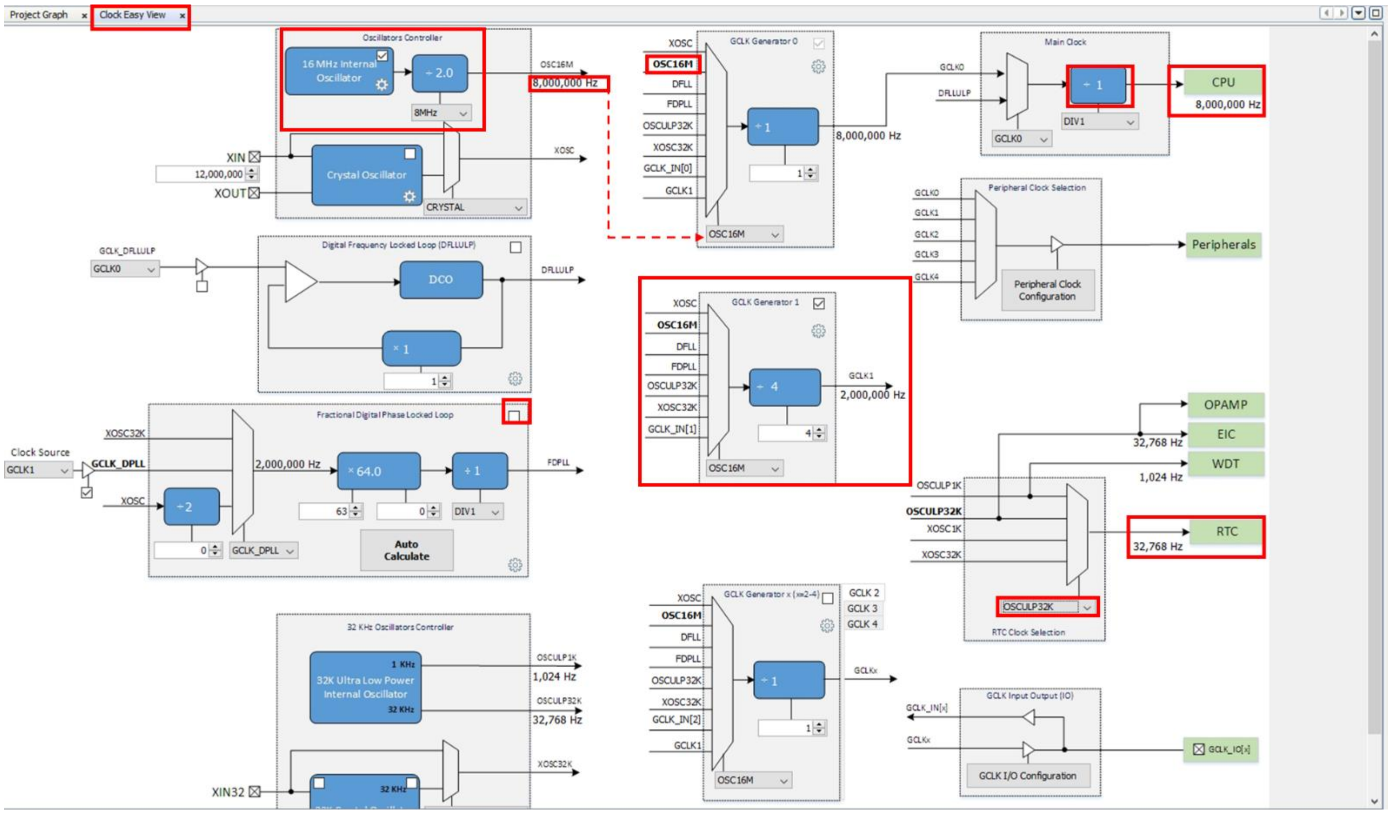
Step 3a: Launch Clock Easy View by going to **Project Graph** tab in MPLABX IDE and then selecting **Plugins > Clock Configuration**



A new window “Clock Easy View” is opened in the project’s Main Window.

Step 3b: Click on the **Clock Easy View** tab, scroll to the right and change the clocks settings as below.

Verify that the **Main Clock** is set to **8 MHz.**

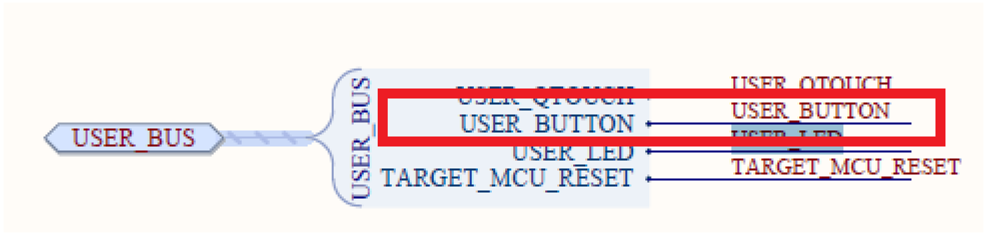


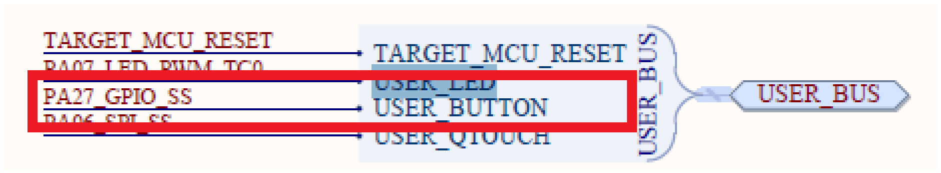
To measure the current consumptions in the different low power modes, we will setup our clocks and oscillators in the following conditions:

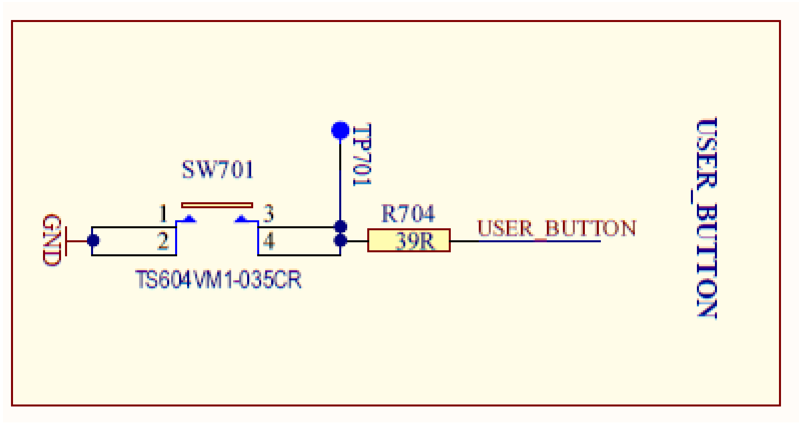
- OSC16M is used as main clock source running at 8MHz (PL0 mode)
- XOSC, XOSC32K, DFLL are stopped
- All AHB & APB clocks enabled (reset values)
- CPUDIV=1 (CPU Clocks Dividers)

STEP 4: Add and configure EIC, select pins

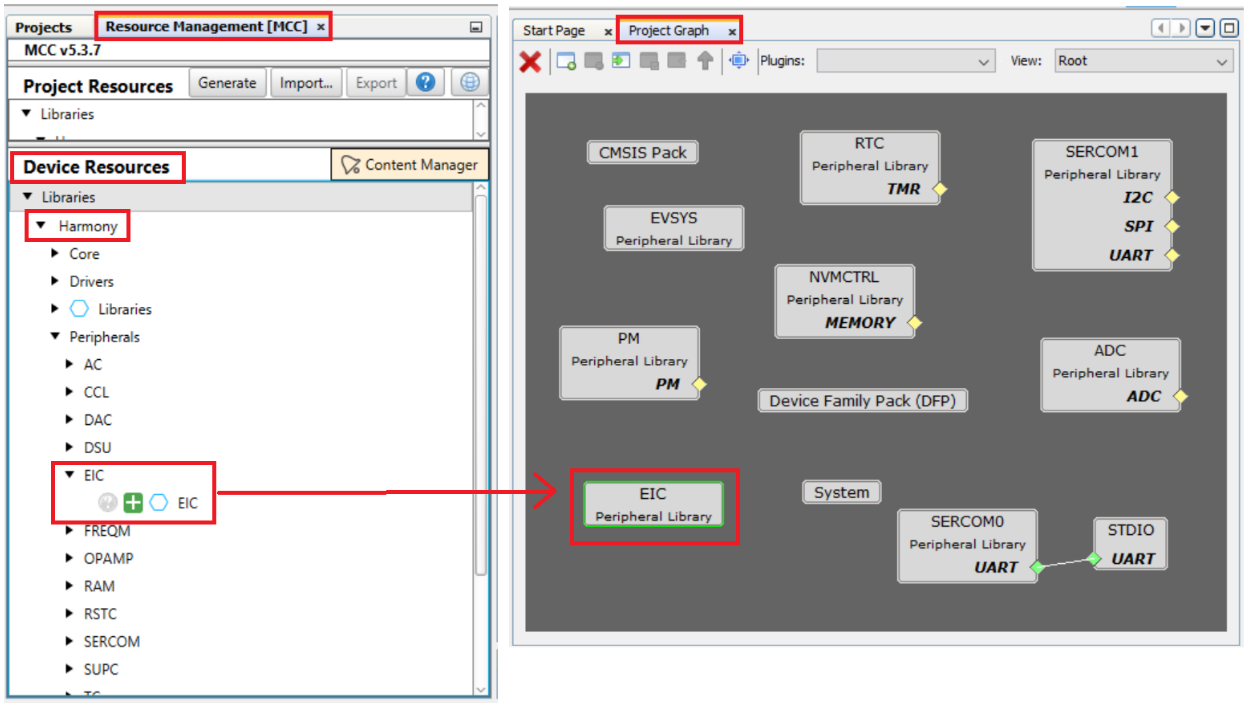
In this step, the switch button will be configured as per the following USER_BUTTON design schematic on SAM L10 Xplained Ultra Evaluation Kit.



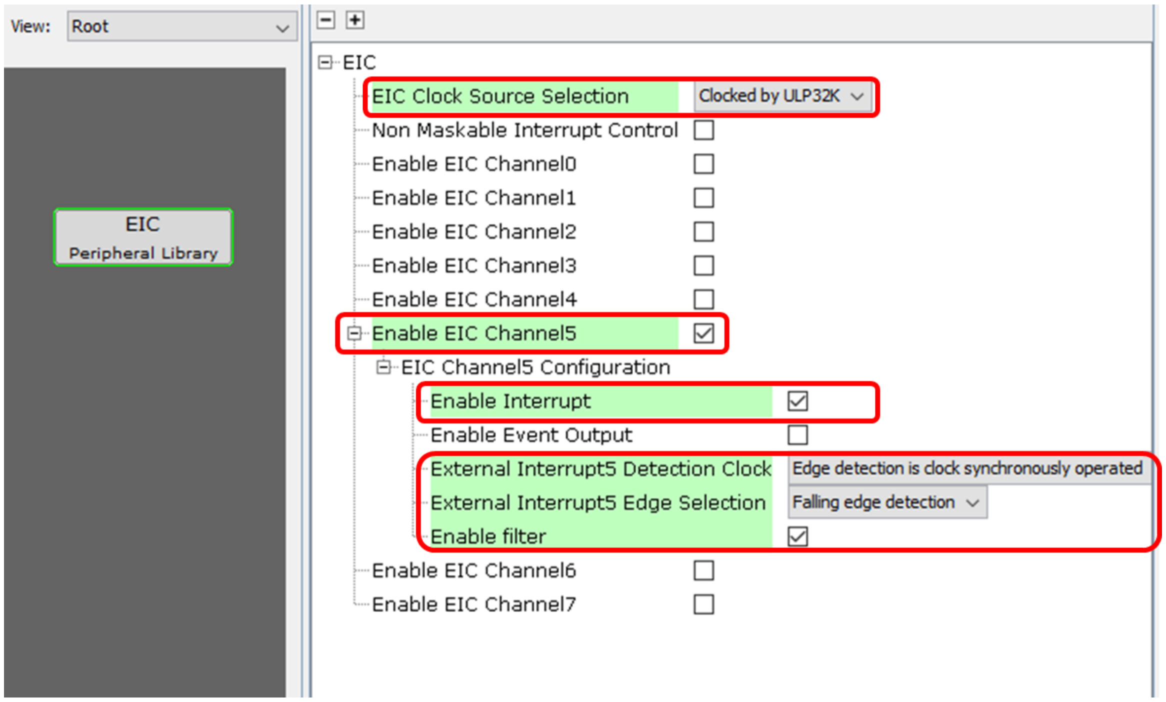




Step 4a: Under the **Device Resources** tab, expand **Harmony > Peripherals > EIC**. Select and double click on **EIC** to add the EIC module to the project.

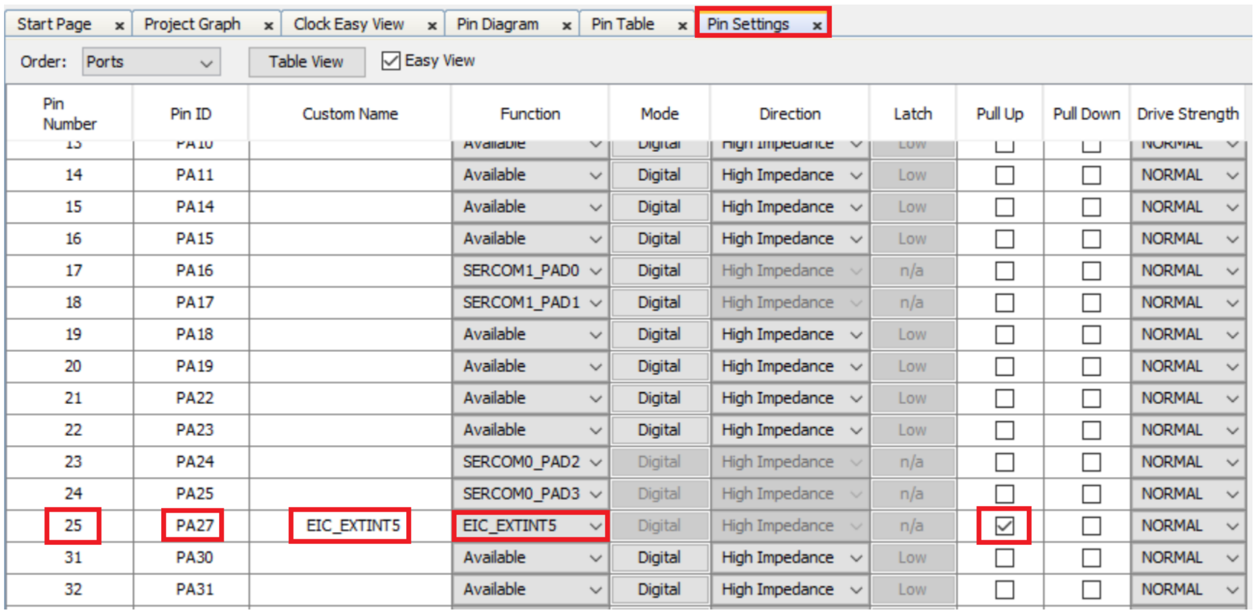


Step 4b: Configure the EIC block to generate an interrupt every time the user presses the switch SW0 as shown in the accompanying figure, and enable filter functionality to avoid electrical noise on switch pin.



EIC
- EIC Clock Source Selection: Clocked by ULP32K
- Non Maskable Interrupt Control: ☐
- Enable EIC Channel0: ☐
- Enable EIC Channel1: ☐
- Enable EIC Channel2: ☐
- Enable EIC Channel3: ☐
- Enable EIC Channel4: ☐
- Enable EIC Channel5: ☑
  - EIC Channel5 Configuration
    - Enable Interrupt: ☑
    - Enable Event Output: ☐
    - External Interrupt5 Detection Clock: Edge detection is clock synchronously operated
    - External Interrupt5 Edge Selection: Falling edge detection
    - Enable filter: ☑
- Enable EIC Channel6: ☐
- Enable EIC Channel7: ☐

Step 4c: In the Plugins, select the **Pin Settings** tab and then scroll down to 25 in the **Pin Number** column and configure the PORT pin **PA27** as an external interrupt pin for switch functionality as shown below. Internal pull-up is enabled to avoid false edge detection as there is no external pull-up on the SAM L10 Xplained Pro Evaluation Kit.



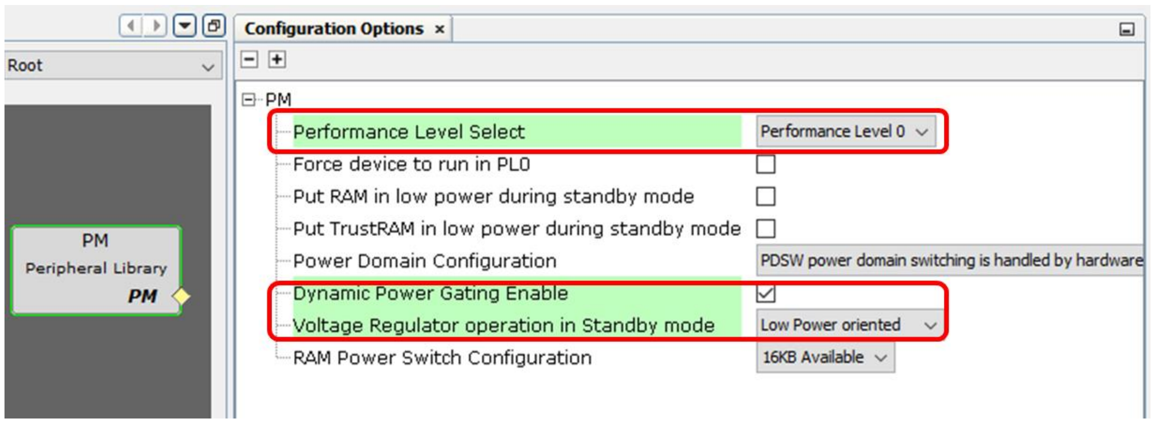
Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
13	PA10		Available	Digital	High Impedance	Low	☐	☐	NORMAL
14	PA11		Available	Digital	High Impedance	Low	☐	☐	NORMAL
15	PA14		Available	Digital	High Impedance	Low	☐	☐	NORMAL
16	PA15		Available	Digital	High Impedance	Low	☐	☐	NORMAL
17	PA16		SERCOM1_PAD0	Digital	High Impedance	n/a	☐	☐	NORMAL
18	PA17		SERCOM1_PAD1	Digital	High Impedance	n/a	☐	☐	NORMAL
19	PA18		Available	Digital	High Impedance	Low	☐	☐	NORMAL
20	PA19		Available	Digital	High Impedance	Low	☐	☐	NORMAL
21	PA22		Available	Digital	High Impedance	Low	☐	☐	NORMAL
22	PA23		Available	Digital	High Impedance	Low	☐	☐	NORMAL
23	PA24		SERCOM0_PAD2	Digital	High Impedance	n/a	☐	☐	NORMAL
24	PA25		SERCOM0_PAD3	Digital	High Impedance	n/a	☐	☐	NORMAL
25	PA27	EIC_EXTINT5	EIC_EXTINT5	Digital	High Impedance	n/a	☑	☐	NORMAL
31	PA30		Available	Digital	High Impedance	Low	☐	☐	NORMAL
32	PA31		Available	Digital	High Impedance	Low	☐	☐	NORMAL

Part 3: Implement active idle and standby mode with PM, SUPC and NVMCTRL configurations.

STEP 5: Configure PM and SUPC as required

Step 5a: The Power Manager (PM) is added by default to the project graph. Select the **PM** peripheral library in the **Project Graph** window and configure it as shown below.

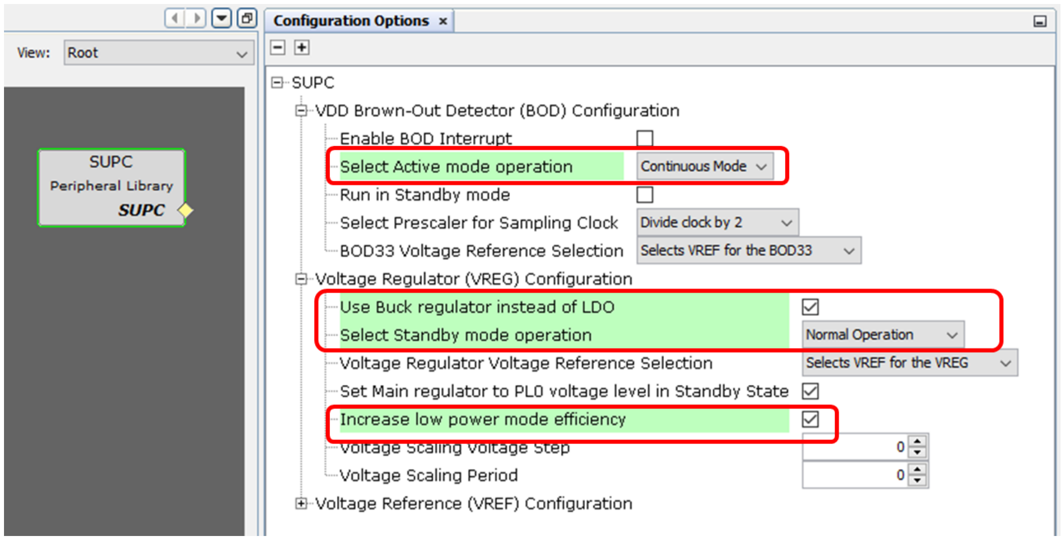
- Set the Performance Level Select to **Performance Level 0** to reduce the power consumption of the device during Standby mode.
- Enable **Dynamic Power Gating**: These options allow the device to turn off a power domain during Standby mode if no peripheral contained in this power domain requests its clock.
- Set the Voltage Regulator operation in Standby Mode to operate in **Low Power oriented** mode during standby to reduce the power consumption of the device even more.



PM
- Performance Level Select: Performance Level 0
- Force device to run in PL0: ☐
- Put RAM in low power during standby mode: ☐
- Put TrustRAM in low power during standby mode: ☐
- Power Domain Configuration: PDSW power domain switching is handled by hardware
- Dynamic Power Gating Enable: ☑
- Voltage Regulator operation in Standby mode: Low Power oriented
- RAM Power Switch Configuration: 16KB Available

Step 5b: Select the **SUPC** peripheral library and configure it as shown below.

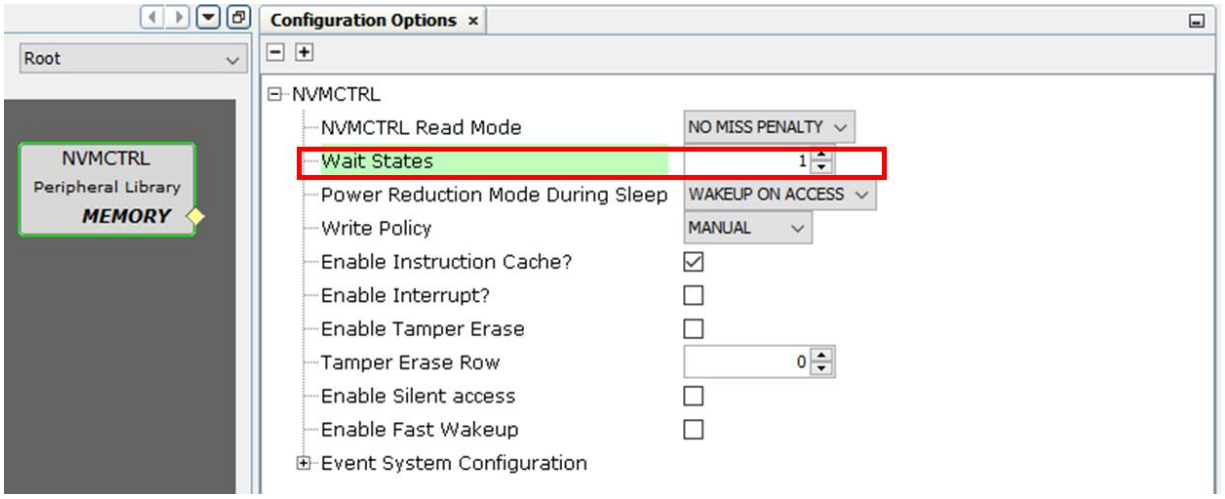
- Enable the **Increase low power mode efficiency** bit to optimize and reduce power consumption in Standby mode.



SUPC
- VDD Brown-Out Detector (BOD) Configuration
  - Enable BOD Interrupt: ☐
  - Select Active mode operation: Continuous Mode
  - Run in Standby mode: ☐
  - Select Prescaler for Sampling Clock: Divide clock by 2
  - BOD33 Voltage Reference Selection: Selects VREF for the BOD33
- Voltage Regulator (VREG) Configuration
  - Use Buck regulator instead of LDO: ☑
  - Select Standby mode operation: Normal Operation
  - Voltage Regulator Voltage Reference Selection: Selects VREF for the VREG
  - Set Main regulator to PL0 voltage level in Standby State: ☑
  - Increase low power mode efficiency: ☑
  - Voltage Scaling Voltage Step: 0
  - Voltage Scaling Period: 0
- Voltage Reference (VREF) Configuration

Step 5c: The Non-volatile Memory Controller (NVMCTRL) is added by default to the project graph. Select the **NVMCTRL** peripheral library and configure it as shown below.

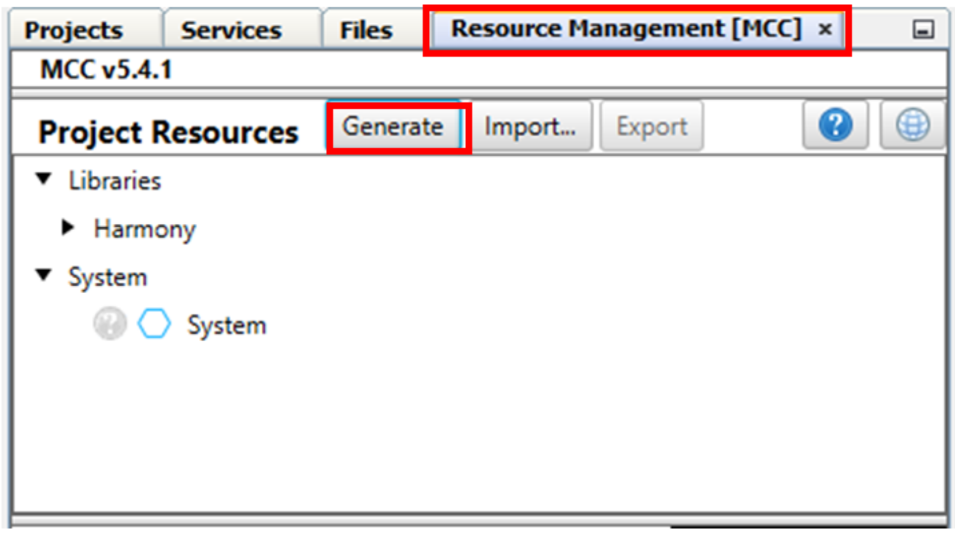
- Set Wait States to 1 to read the non-volatile memory. When the device is put in performance level 0, the device clock frequency cannot exceed 8 MHz and one wait-state is required.



NVMCTRL
- NVMCTRL Read Mode: NO MISS PENALTY
- Wait States: 1
- Power Reduction Mode During Sleep: WAKEUP ON ACCESS
- Write Policy: MANUAL
- Enable Instruction Cache?: ✓
- Enable Interrupt?: ☐
- Enable Tamper Erase: ☐
- Tamper Erase Row: 0
- Enable Silent access: ☐
- Enable Fast Wakeup: ☐
- Event System Configuration

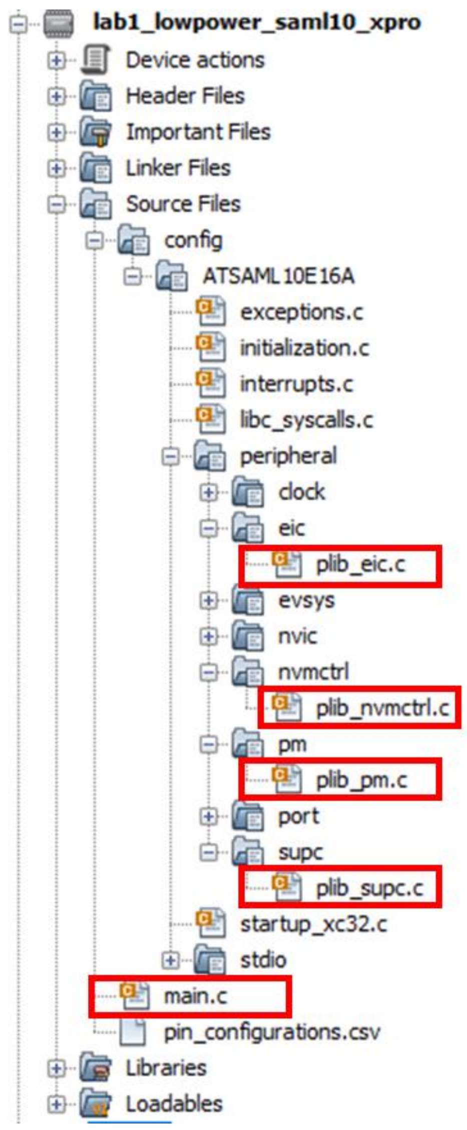
STEP 6: Generate code

Step 6a: Click on the **Generate** button in the Generate icon on Resouce management[MCC] window, as shown below.



Note: The MCC will include all the MPLAB® Harmony library files and generate the code based on the selections.

The generated code will add files and folders in your Harmony project as shown below.



Among the generated code notice the source and header files generated for PM, SUPC, NVMCTRL and EIC peripherals. MCC also generates the template main file.

Note: Navigate to **Projects** tab to view the project tree structure.

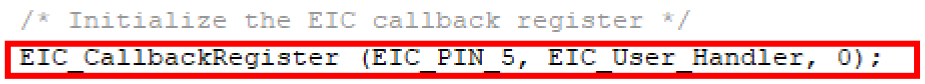
Step 6c: Build the code by clicking on the "Clean and Build" icon and verify the project builds successfully.

At this point you are ready to start implementing your application code.

Part 4: Add code and observe output.

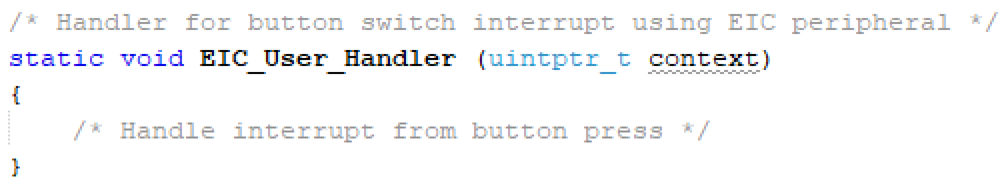
STEP 7: Add code to your application

Step 7a: Open **main.c** file. Register an event handler (callback) with the EIC PLIB



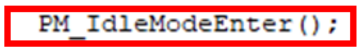
```
/* Initialize the EIC callback register */
EIC_CallbackRegister (EIC_PIN_5, EIC_User_Handler, 0);
```

Step 7b: Add code to implement the EIC event handler. The event handler is called when SW0 is pressed.

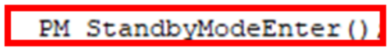


```
/* Handler for button switch interrupt using EIC peripheral */
static void EIC_User_Handler (uintptr_t context)
{
    /* Handle interrupt from button press */
}
```

Step 7c: Finally, add the following code for the MCU to go in standby and Idle mode



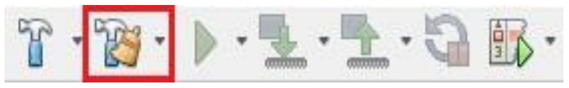
```
PM_IdleModeEnter();
```



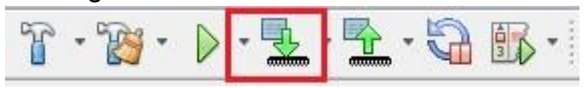
```
PM_StandbyModeEnter()
```

STEP 8: Build and Run the Application

Step 8a: Clean and build your application by clicking on the "Clean and Build" button as shown below.

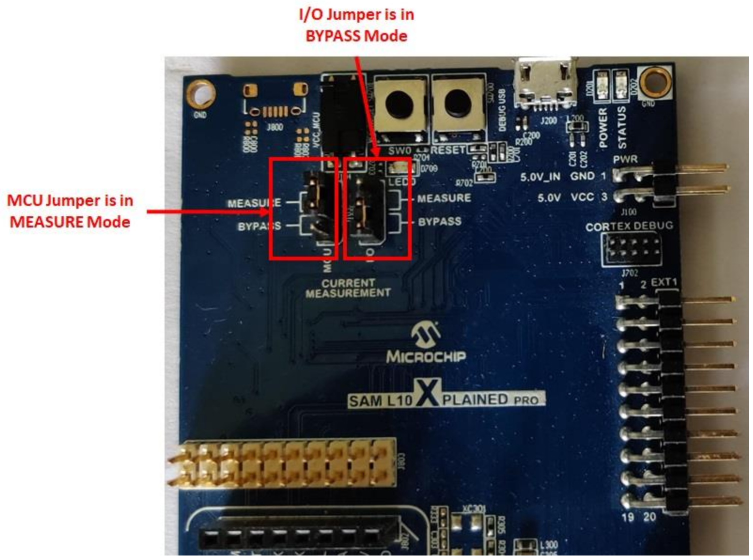


Step 8b: Program your application to the device, by clicking on the "Make and Program" button as shown below.

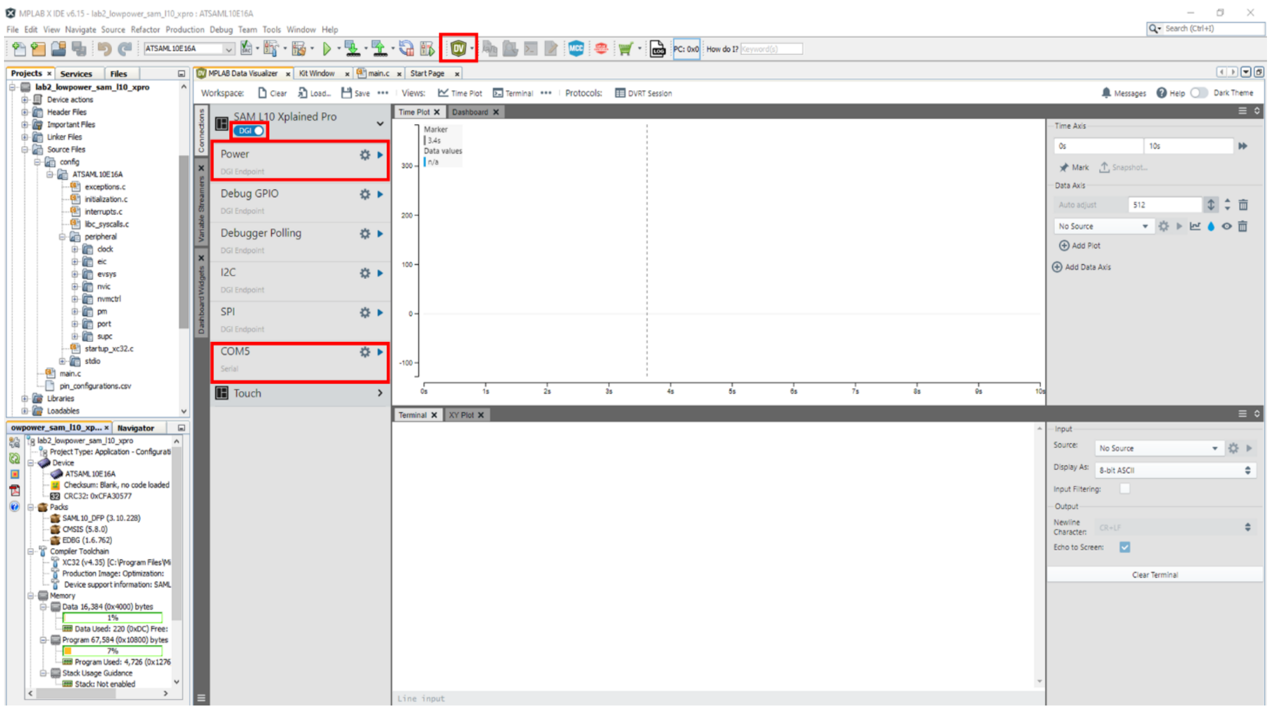


Results:

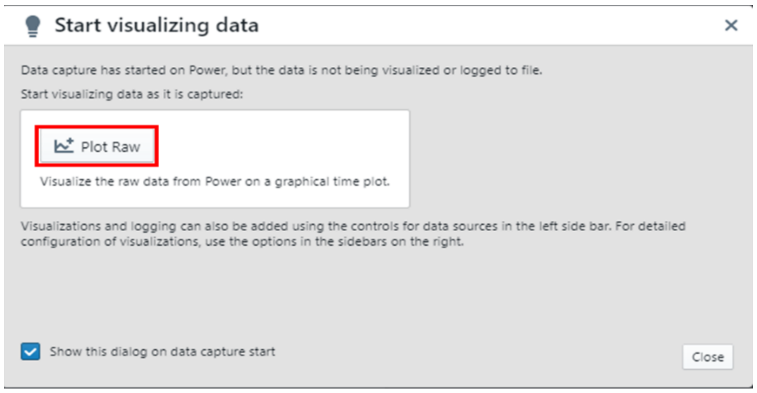
- Ensure that the jumpers for Current Measurement on the SAM L10 Xplained Pro are set to **MEASURE** for the MCU and **BYPASS** for the I/Os.



- Open the **Data Visualizer** application from your PC and select the connected **SAM L10 Xplained Pro** board on the **DGI Control Panel**, then click on **Connect**. The Data Visualizer will then start searching for protocols from the SAM L10 Xplained Pro board through the EDBG.



On clicking power icon, a pop up is displayed to plot values, Select plot raw option and close it.



Start visualizing data

Data capture has started on Power, but the data is not being visualized or logged to file.

Start visualizing data as it is captured:

Plot Raw

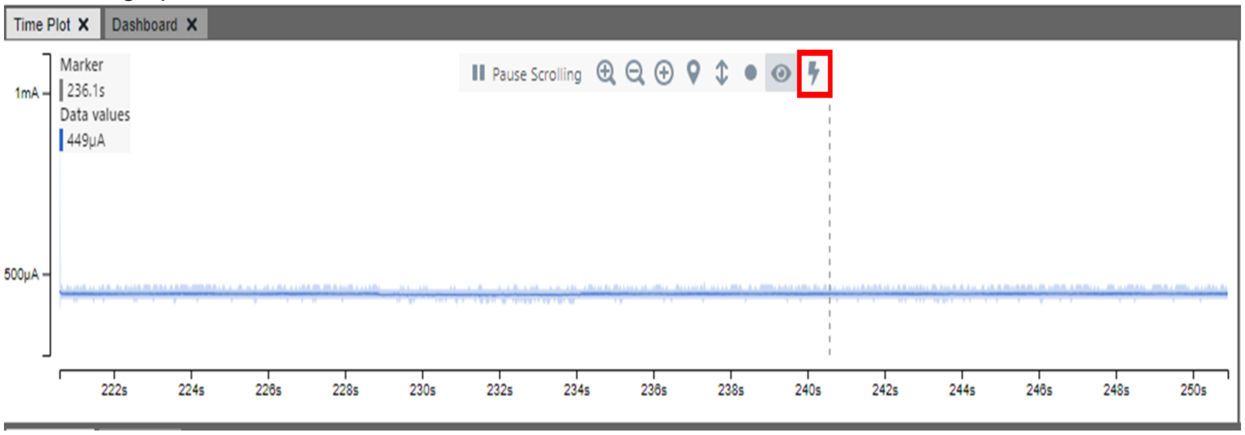
Visualize the raw data from Power on a graphical time plot.

Visualizations and logging can also be added using the controls for data sources in the left side bar. For detailed configuration of visualizations, use the options in the sidebars on the right.

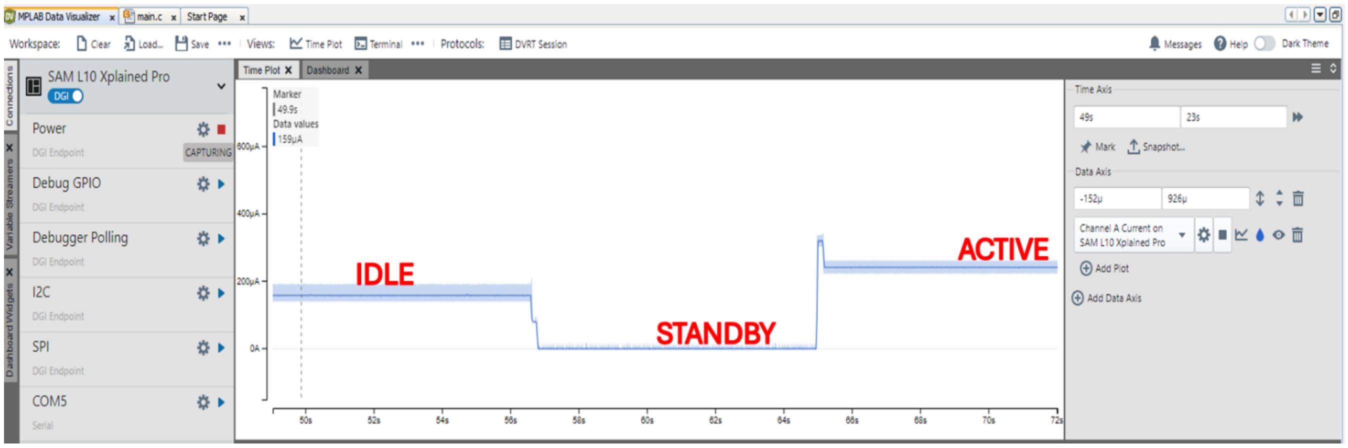
Show this dialog on data capture start

Close

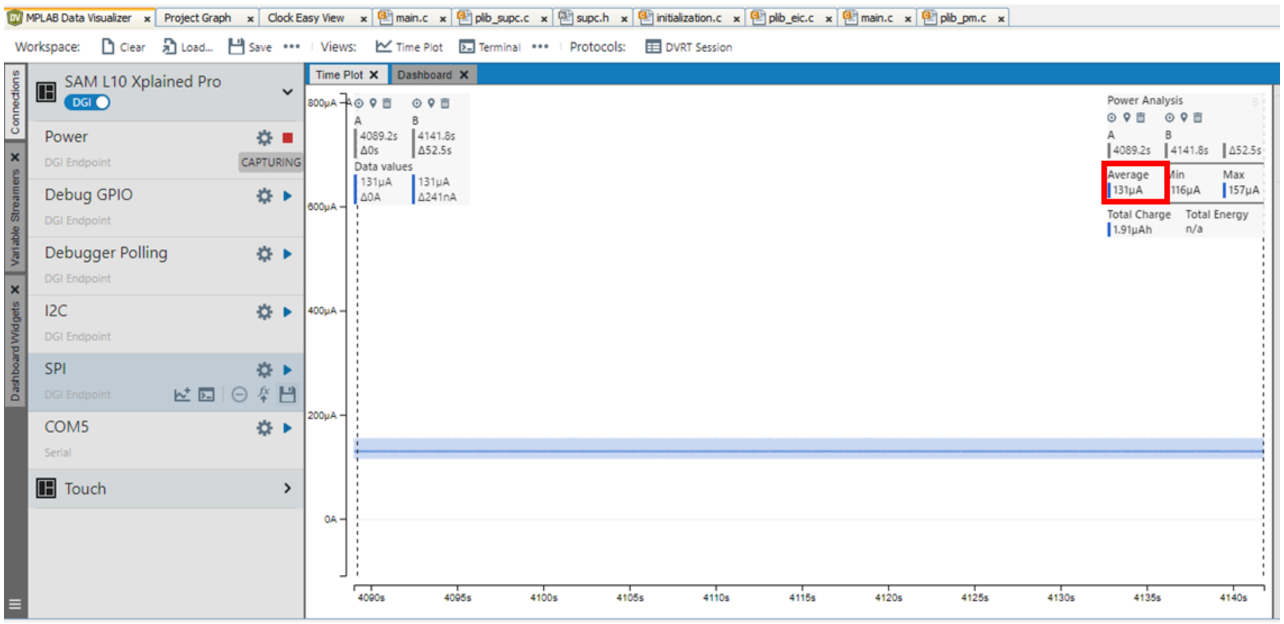
The Power Analysis window will appear on the Data Visualizer tool interface as follows. To measure the average power click the icon.



As you press the SW0 the following result is observed



Idle Mode


A: 4089.2s, Δ0s — B: 4141.8s, Δ52.5s

Data values: 131µA, Δ0A — 131µA, Δ241nA

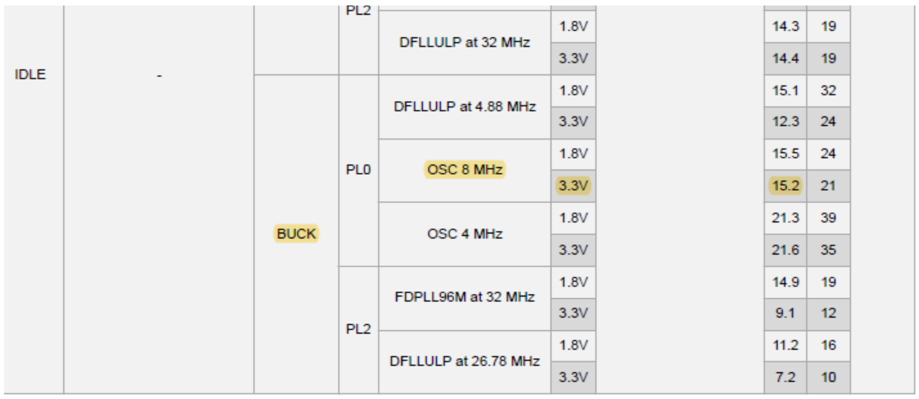
Power Analysis

A: 4089.2s — B: 4141.8s — Δ52.5s

Average	Min	Max
131µA	116µA	157µA

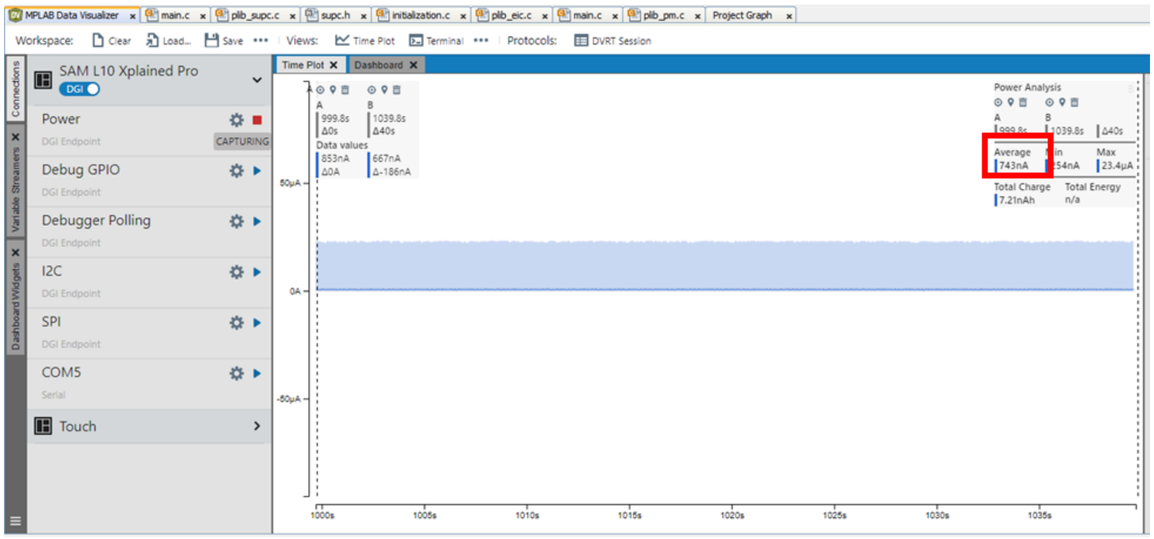
Total Charge	Total Energy
1.91µAh	n/a

- Current Consumption (IDLE mode): **131** µA
- µA/MHz ratio (f = 8MHz): **16.375** µA/MHz

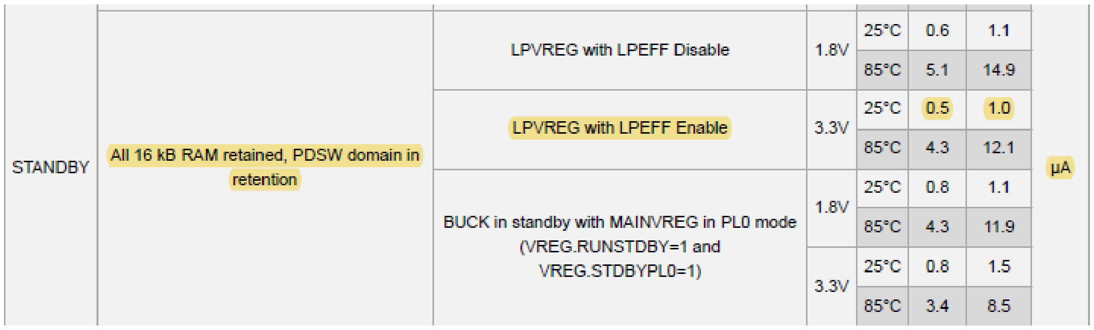


IDLE	-		PL2	DFLLULP at 32 MHz	1.8V		14.3	19
					3.3V		14.4	19
		BUCK	PL0	DFLLULP at 4.88 MHz	1.8V		15.1	32
					3.3V		12.3	24
				OSC 8 MHz	1.8V		15.5	24
					3.3V		15.2	21
				OSC 4 MHz	1.8V		21.3	39
					3.3V		21.6	35
			PL2	FDPLL96M at 32 MHz	1.8V		14.9	19
					3.3V		9.1	12
				DFLLULP at 26.78 MHz	1.8V		11.2	16
					3.3V		7.2	10

Standby Mode

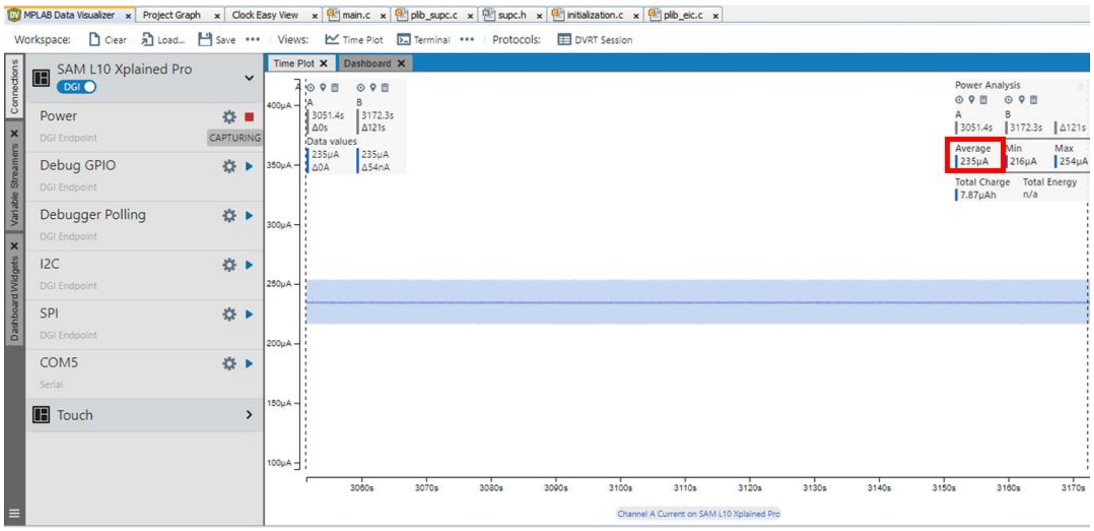


- Current Consumption (STANDBY mode): **743** nA = (0.743 µA)

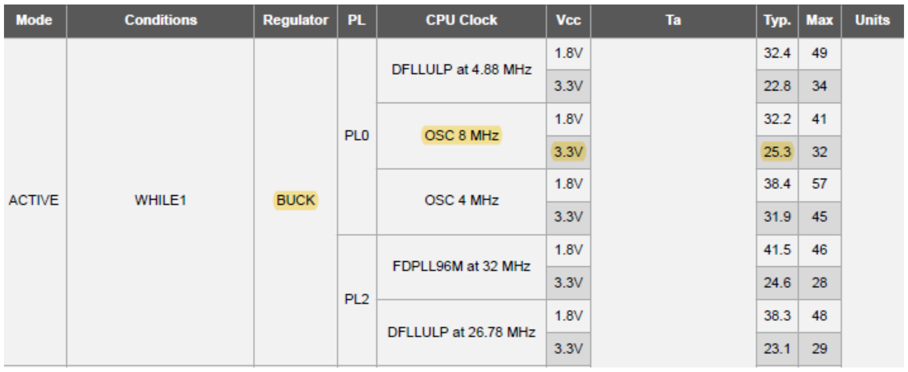


STANDBY	All 16 kB RAM retained, PDSW domain in retention	LPVREG with LPEFF Disable	1.8V	25°C	0.6	1.1	µA
				85°C	5.1	14.9	
		LPVREG with LPEFF Enable	3.3V	25°C	0.5	1.0	
				85°C	4.3	12.1	
		BUCK in standby with MAINVREG in PL0 mode (VREG.RUNSTDBY=1 and VREG.STDBYPL0=1)	1.8V	25°C	0.8	1.1	
				85°C	4.3	11.9	
			3.3V	25°C	0.8	1.5	
				85°C	3.4	8.5	

Active mode:



- Current Consumption (ACTIVE mode): **235** µA
- µA/MHz ratio (f = 8MHz): **29.37** µA/MHz



Mode	Conditions	Regulator	PL	CPU Clock	Vcc	Ta	Typ.	Max	Units
ACTIVE	WHILE1	BUCK	PL0	DFLLULP at 4.88 MHz	1.8V		32.4	49	
					3.3V		22.8	34	
				OSC 8 MHz	1.8V		32.2	41	
					3.3V		25.3	32	
				OSC 4 MHz	1.8V		38.4	57	
					3.3V		31.9	45	
			PL2	FDPLL96M at 32 MHz	1.8V		41.5	46	
					3.3V		24.6	28	
				DFLLULP at 26.78 MHz	1.8V		38.3	48	
					3.3V		23.1	29	

Lab 1.b

Part 5: Add code for OFF mode and observe output.

STEP 9: Add code to your application, build and run

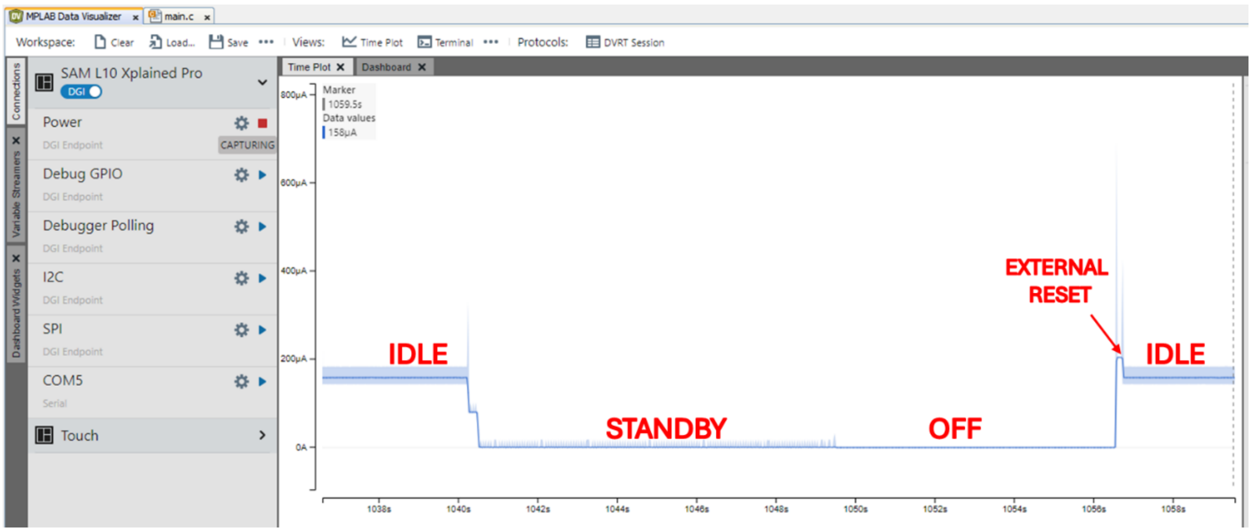
Step 9a: Open **main.c** file. Add the following line

```
PM_OffModeEnter();
```

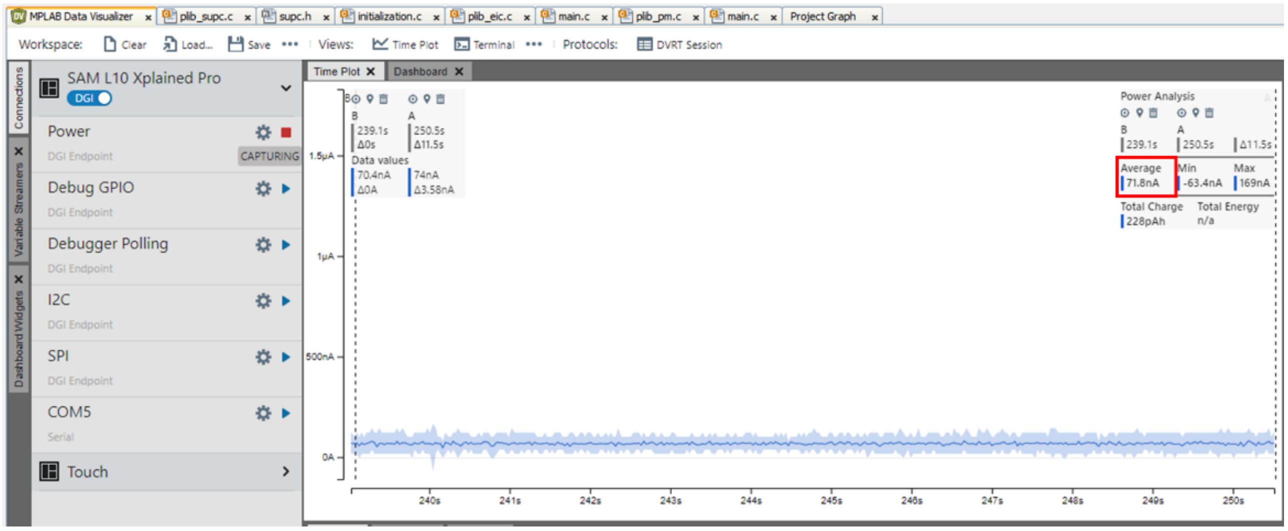
Step 9b: Compile and run as per step 8

Results:

As SW0 is pressed each time, the mode changes as shown in the diagram below:



OFF mode



- Current Consumption (STANDBY mode): **71.8** nA

OFF	1.8V	25°C	34.6	54.4	nA
		85°C	595.7	1197.3	
	3.3V	25°C	61.2	89.1	
		85°C	796.1	1622.8	

Fill the measured and calculated values and check if they match as per data sheet

Mode	Avg power measured in visualizer	Calculated	As per datasheet
Active			Typ: 25.3 µA/MHz Max: 32 µA/MHz
Idle			Typ: 15.2 µA/MHz Max: 21 µA/MHz
Standby			Typ: 0.5 µA Max: 1 µA
Off			Typ: 61.2 nA Max: 89.1 nA

Summary:

After completing it, you should now:

- Have a clear picture of most of the SAM L10 key power optimization features.
- Know how to use low power features.
- Know how to analyze power consumption on the SAM L10 Xplained with Data Visualizer tool.

We have demonstrated some of the SAM L10 low power techniques which can be used to optimize the power consumption of any application:

- The three different voltage regulators to optimize the device power consumption depending on the system load in the different active and sleep modes
- The BUCK converter power efficient mode
- The Performance Level capability (dynamic voltage scaling) to adjust regulator output voltage against operating frequency
- The flexible sleep modes: IDLE, STANDBY and OFF.

In addition, the SAM L10 brings a unique low power technique dedicated to the STANDBY sleep mode: the Power Gating feature. This technique allows you to automatically turn off unused power domain supplies individually while keeping others powered up.

This technique combined with the Sleepwalking feature (capability for a device in sleep mode, to temporarily wake-up clocks for a peripheral to perform a task without waking-up the CPU) allows to reduce the overall power consumption of the device in STANDBY mode. This will be demonstrated in the next lab.