

Lab 2 – Create an application to read light sensor data when torch is flashes and print it on a serial terminal, use data visualizer to measure the power consumption.

Purpose:

After completing Lab 2, you will learn how to add and configure the ADC, Event System and Power Management peripheral libraries. In the process, you will understand the use of EDBG and Data Visualizer tool for power measurements.

Overview:

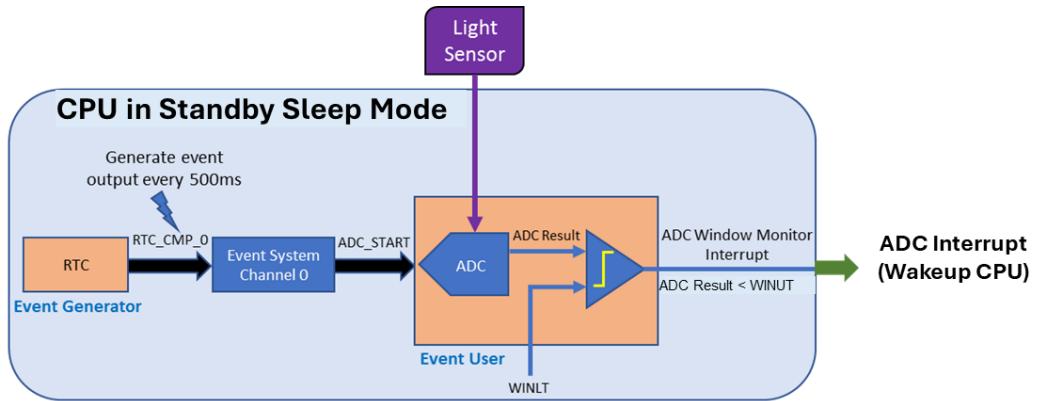
In this lab you will create a MPLAB Harmony project using MCC. The application you create will read light sensor value at every 500ms and check if it is in the window range then, print it on a serial terminal. The application will use:

- RTC Peripheral Library to generate periodic events
- SERCOM-USART Peripheral Library to print the temperature values on a serial port terminal application running on a PC
- ADC Peripheral Library to sample the light sensor analog input and detect whether the light sensor is flashed or not
- Event System Peripheral Library to trigger the start of ADC conversion on every RTC compare match event. The Event System allows peripheral-peripheral communication without CPU intervention. This reduces the burden on the CPU and other resources when compared to the conventional interrupt-based systems.
- PM (Power Manager) Peripheral Library to put the CPU in IDLE Sleep mode.

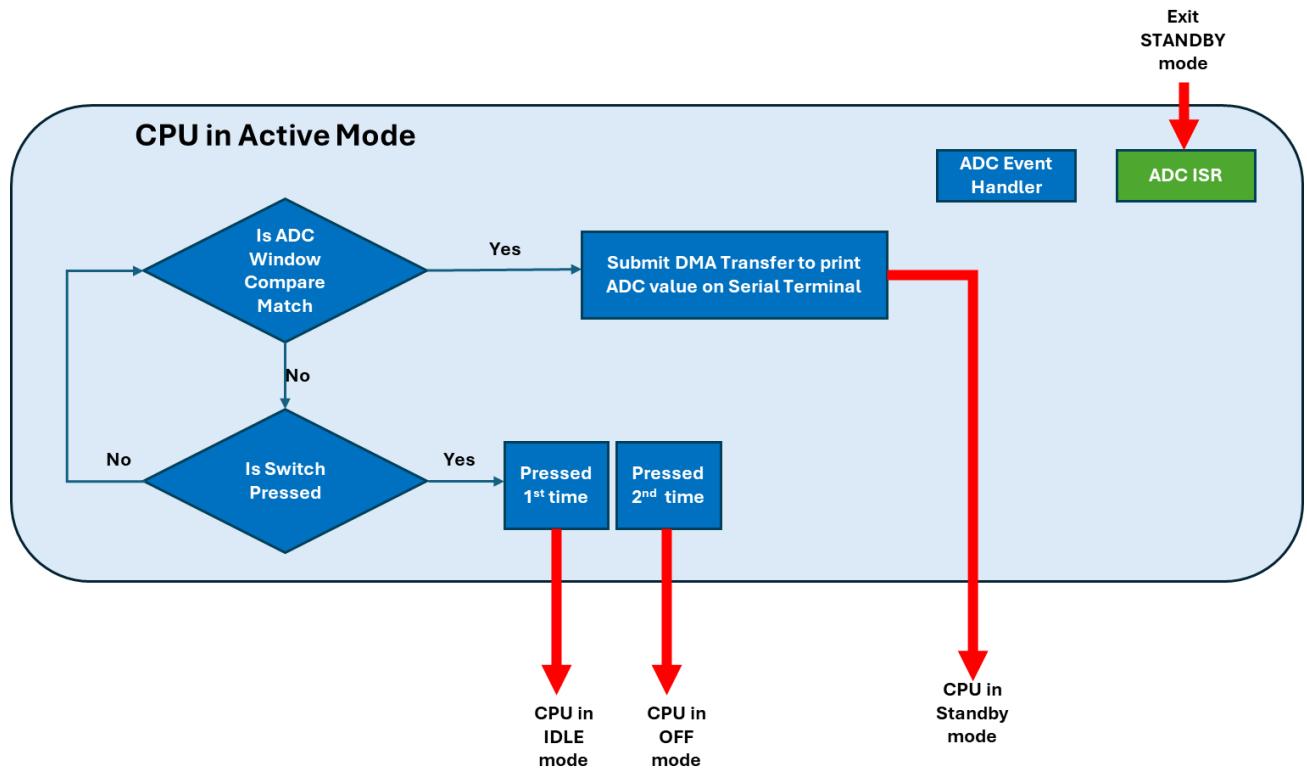
Initialization Flowchart:



CPU in Standby Sleep Mode:



CPU in Active Mode:



Procedure:

This lab is completed in the following steps.

Part 1: Configure ADC and SERCOM, RTC and PM Peripheral libraries.

STEP 1: Configure ADC and SERCOM Peripheral Library

STEP 2: Disable RTC interrupt and enable Compare Event Output

Part 2: Configure EVSYS, DMAC

STEP 3: Configure the EVSYS

STEP 4: Configure the DMA

STEP 5: Generate code

Part 3: Add application code and build the application

STEP 6: Add application code

STEP 7: Build and Run the Application

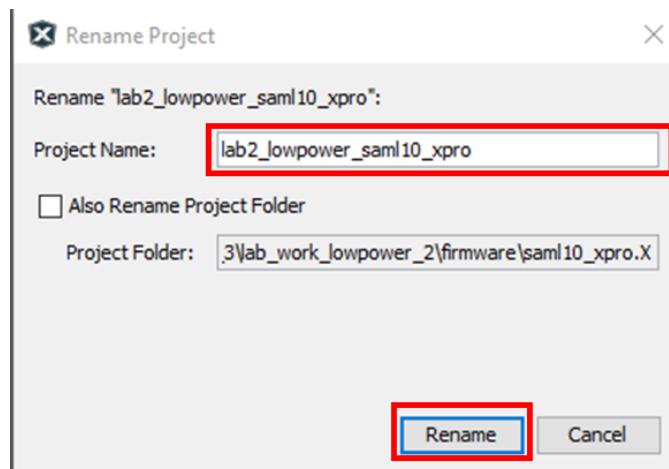
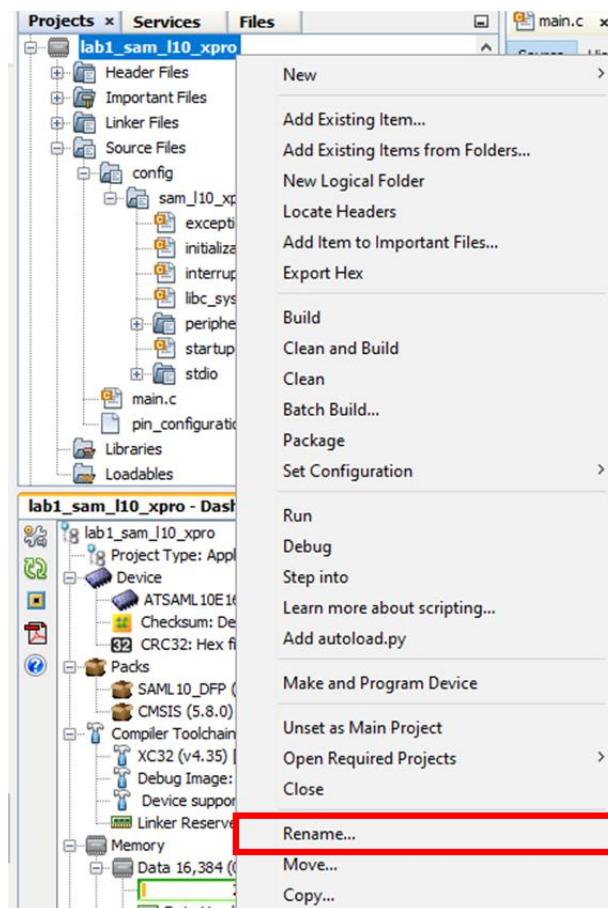
Note: You will develop Lab2 by extending the Lab1. If for some reason you were unable to complete the Lab1 successfully, then follow the below procedure (**otherwise, skip these steps**):

1. Close the Lab1 project (**lab1_lowpower_sam_l10_xpro**) in MPLAB® X IDE by going to **File>Close All Projects**
2. Delete the **firmware** folder created by you at **<your-project-path>** location. For example, for the path used in this lab manual, the firmware folder will be under **/lab_work/labs** folder.
3. Copy the ready-made solution for the Lab1 available under the **lab_work/solutions/saml10/lab1** folder and paste the solution (**firmware** folder) to the **<your-project-path>** location. For example, for the path used in this lab manual, paste the **firmware** folder under **lab_work/labs** folder.
4. Drag and drop the MPLAB X project file (**sam_l10_xpro.X**) available under **<your-project-path>/firmware/** to the MPLAB X IDE
5. In MPLAB X IDE, right click on the project and set it as the main project.
6. In MPLAB X IDE, go to **Tools > Embedded** and open the **MPLAB Code Configurator**.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Since you are working on Lab2, change the name of the project under project tree from **lab1_lowpower_sam_l10_xpro** to **lab2_lowpower_sam_l10_xpro**.

Click on the “Projects” tab on the top left pane. Right click on the project name “**lab1_sam_l10_xpro**” and click on “**Rename...**”.



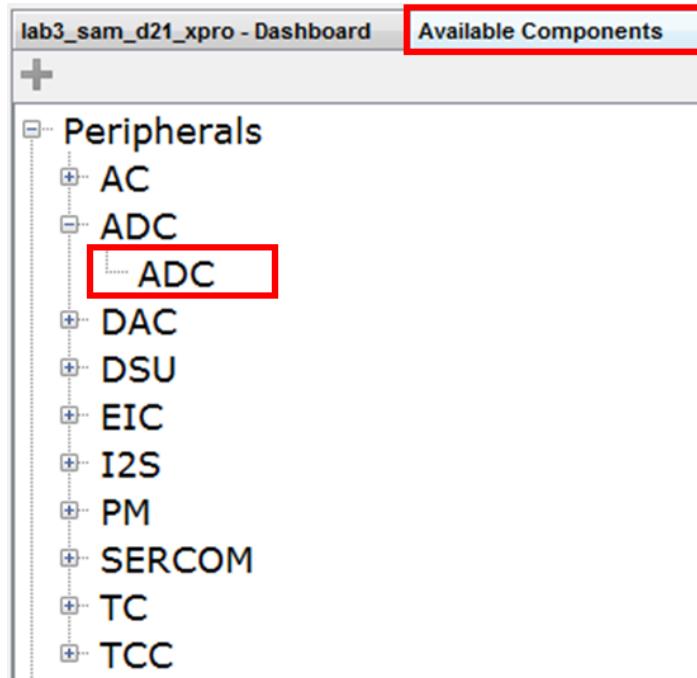
Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Part 1: Configure ADC, RTC and PM Peripheral libraries

STEP 1: Configure ADC and SERCOM Peripheral Library

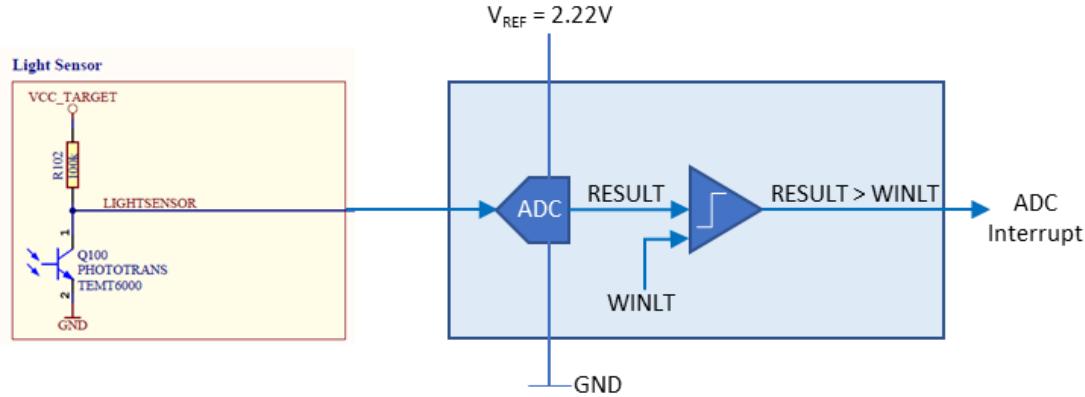
Step 1a: Under the bottom left tab “Available Components”, Expand Peripherals>ADC

Select and double click on ADC to add the ADC module to the project.



Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

You will configure ADC PLIB to sample and convert the light sensor input. You will also configure the ADC to generate an interrupt (and thereby wakeup CPU), when the ADC Result is greater than a set threshold value.



When light sensor is not covered (light is falling on the sensor), the phototransistor is turned on. The ADC input is $\sim 0V$ and the ADC RESULT register is close to 0x00.

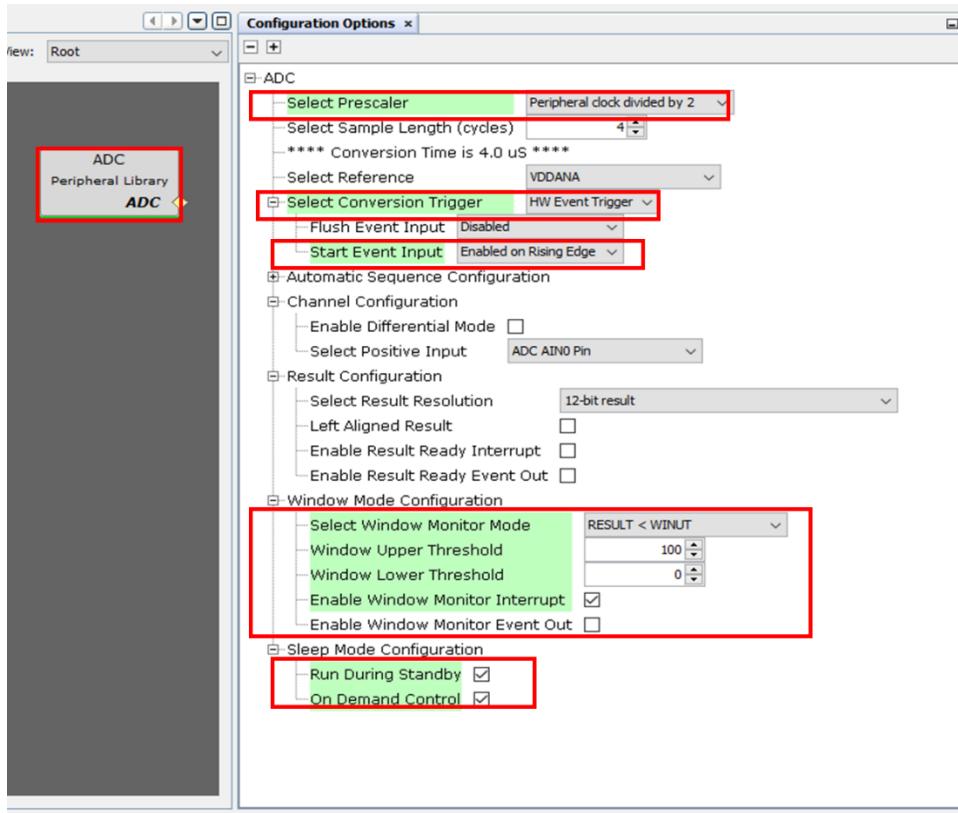
When the light sensor is covered (light is not falling on the sensor), the phototransistor is turned off. The ADC input is $\sim 3.3V$ and the **ADC RESULT** register will be saturated (0xFF).

The ADC RESULT register is compared with the **Window Lower Threshold (WINLT)** which is set to **100**.

When the **light sensor is flashed** with torch/ light source (**ADC RESULT < WINLT**), a ADC Window Monitor interrupt is generated. This interrupt is used to bring the CPU out of IDLE sleep mode, when the user covers the light sensor.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

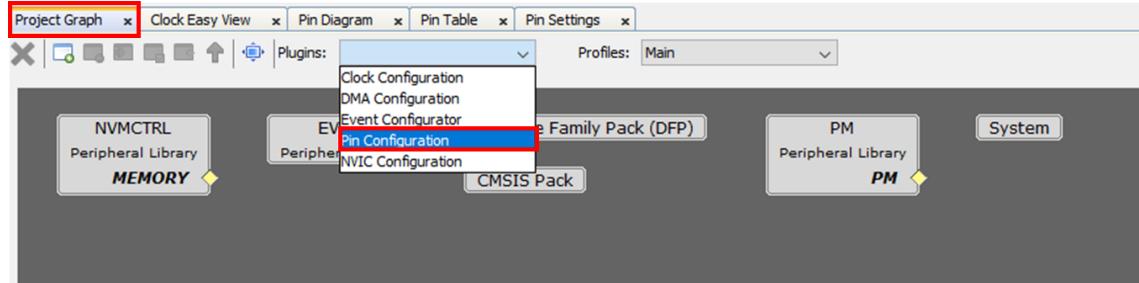
Select the ADC PLIB module and configure as shown below.



1. Select Prescaler: Peripheral clock divided by 2 divides the ADC input clock with the configured pre-scaler value and provides more sampling time ($\text{CLKADC} = 1 \text{ MHz}/2 = 500 \text{ kHz}$).
2. Select Reference: **VDDANA**.
3. Select Conversion Trigger: **HW Event Trigger**.
4. Start Event Input: **Enabled on Rising Edge**.
5. Select Positive Input: **ADC AIN0 Pin** (the light sensor is connected to AIN0 pin of ADC).
6. Select Result Resolution: **12-bit result** (ADC conversion value range from 0 to 4096).
7. Open the **Window Mode Configuration** panel:
 - o Select Window Monitor mode: **Result > WINLT**. A Window Monitor Interrupt is generated when the ADC result is greater than the configured WINLT value.
 - o Window Upper Threshold: Sets the upper threshold of the window comparator. It is set to **100** as the ADC resolution is 12-bit.
 - o Window Lower Threshold: Sets the lower threshold of the window comparator. It is set to **0**. This means that the ADC Window Monitor Interrupt will be generated when the ADC result is lesser than 100. This value is decided based on the light sensor voltage when we flash torch on light sensor.
 - o Enable Window Monitor Interrupt: A Window Monitor Interrupt is generated when the ADC result is greater than the configured WINLT.
 - o Window Upper Threshold – Sets the lower threshold of the window comparator. It is set to 100. This means that ADC Window Monitor Interrupt will be generated when the ADC result is less than threshold value.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Step 1b: Open the Pin Configuration tabs by clicking Plugins> Pin Configuration in the project Graph



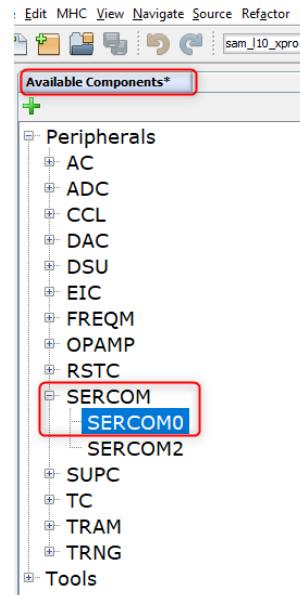
Go to the Pin Settings tab and configure PA02 (Pin #3) as ADC_AIN0 pin.

The screenshot shows the 'Pin Settings' tab of the Project Graph. The table lists pins from 1 to 5. Row 3 is highlighted with a red box, indicating it's the current pin being configured. The 'Custom Name' column for this row contains 'ADC_AIN0'. Other columns include Pin Number, Pin ID, Function, Mode, Direction, Latch, Pull Up, Pull Down, and Drive Strength. The 'Function' column for PA02 shows 'Available' with a dropdown arrow, while the other pins show 'Digital' or 'Analog'.

Pin Number	Pin ID	Custom Name	Function	Mode	Direction	Latch	Pull Up	Pull Down	Drive Strength
1	PA00		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
2	PA01		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
3	PA02	ADC_AIN0	Analog	Digital	High Impedance	n/a	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
4	PA03		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL
5	PA04		Available	Digital	High Impedance	Low	<input type="checkbox"/>	<input type="checkbox"/>	NORMAL

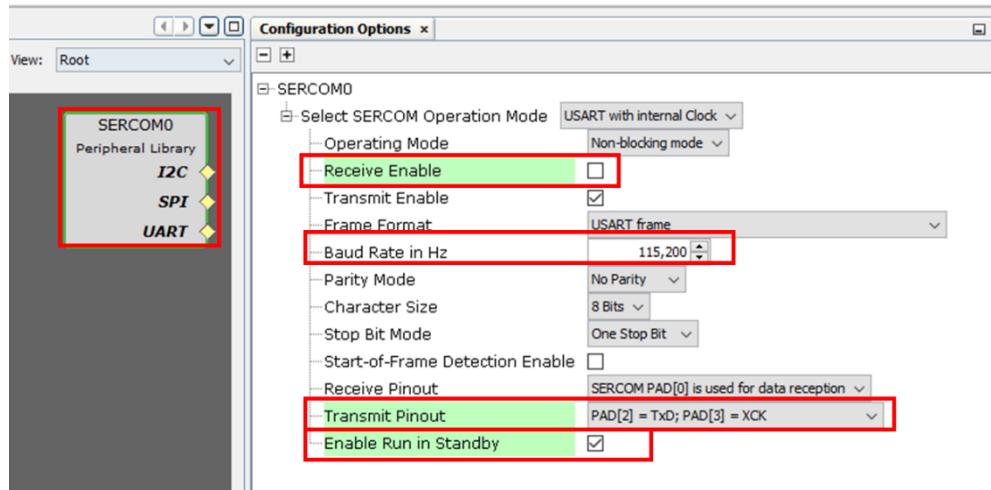
Step 1c: Under the bottom left tab “Available Components”, Expand Peripheral > SERCOM

Select and double click on **SERCOM0** to add the SERCOM instance 0 to the project.



Select the SERCOM0 PLIB in the Project Graph and configure it for USART protocol as shown below.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework



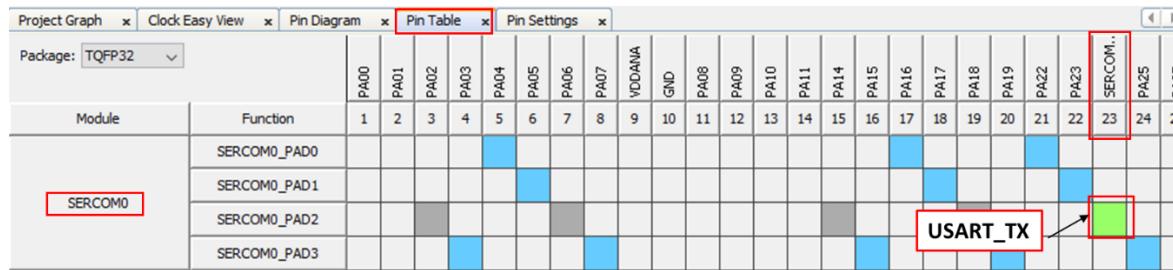
Verify the Operating Mode is set to “Non-blocking mode” and the Baud Rate is set to 115200 Hz.

Also, as per the "SAM L10 Xplained Pro Evaluation Kit" design, **SERCOM0 PAD2** is used for SERCOM0 (as USART) data transmission.

Note: The application will use the SERCOM0 USART PLIB for printing messages on the serial terminal. Hence, only the transmit functionality is enabled and the receive functionality is disabled.

Step 1d: Select the **Pin Table** tab and then scroll down to the **SERCOM0** module as shown below.

Enable USART_TX (SERCOM0_PAD2) on **PA24** (Pin #23)



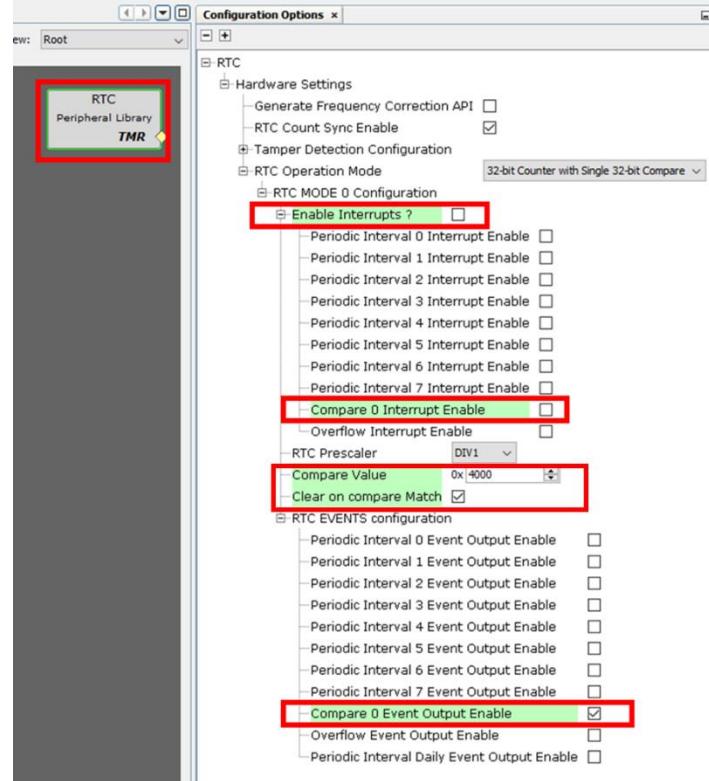
Note: In the SERCOM0 USART configuration, USART is enabled for transmit functionality alone. Hence, the USART receive pin is not configured. The IO pins may also be configured using the Pin Settings tab.

Note: The USART transmit pin can also be configured from the Pin Settings tab by selecting SERCOM0_PAD2 function for pin PA24.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

STEP 2: Disable the RTC interrupt and enable Compare Event Output

Under the “Project Graph” select the RTC PLIB block and disable the RTC Compare match interrupt. Also, enable the Compare 0 Event Output as shown below.



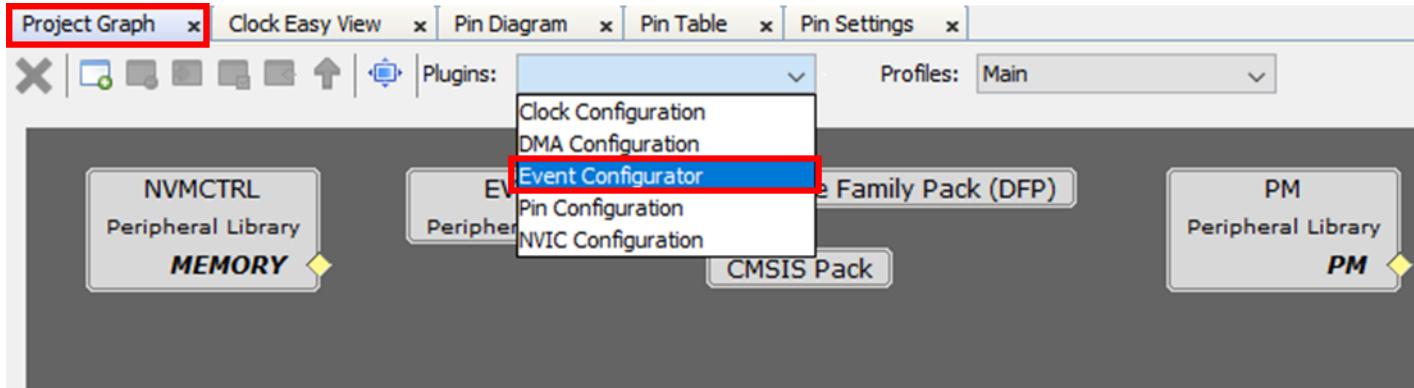
Note: The RTC Compare 0 Event (when RTC counter matches the compare value) output will serve as an input to the Event System module to trigger ADC to start conversion of the light sensor. As a result, CPU intervention is not required to start the ADC conversion on RTC interrupt. Hence, the RTC interrupt is disabled.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Part 2: Configure EVSYS, DMAC

STEP 3: Configure Event System

Step 3a: Launch Event System Configuration window by going to **Project Graph** tab in MPLABX IDE and then selecting **Plugins > Event System Configuration**



A new widow tab “EVENT0 Easy View” is opened.

A screenshot of the "Event Configurator" window. The title bar says "Event Configurator". The main area has two sections: "Channel Configuration" and "User Configuration".
Channel Configuration: A table with columns: Channel Number, Event Generator, Event Status, User Ready, and Remove Channel. A message below the table says "No results found". A blue "Add Channel" button is at the bottom.
User Configuration: A table with columns: User, Channel Number, and Remove User. A message below the table says "No results found". A blue "Add User" button is at the bottom.
A message in the center right says "Select the channel from Channel Configuration Table to view channel settings".

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Step 3b: Add and configure Event System Channel 0 with

- RTC compare 0 match as the event generator
- ADC Start of Conversion as the event user

Note: Make sure that the status of the Event and User (Event Status and User Ready) is indicated Green in color. If it is Red, verify if the Event Output and Event Input is enabled in the respective (RTC and ADC) PLIB configuration.

Enable generic clock on demand and run in standby mode check boxes as we require the event system to be running in standby mode also.

Event Configurator

EVENT CONFIGURATOR

Channel Configuration

Channel Number	Event Generator	Event Status	User Ready	Remove Channel
Channel 0	RTC_CMP_0	●	●	☒

Add Channel

Channel 0 Settings

Path Selection: ASYNCHRONOUS

Event Edge Selection: NO_EVT_OUTPUT

Generic Clock On Demand

Run In Standby Sleep Mode

Enable Event Detection Interrupt

Enable Overrun Interrupt

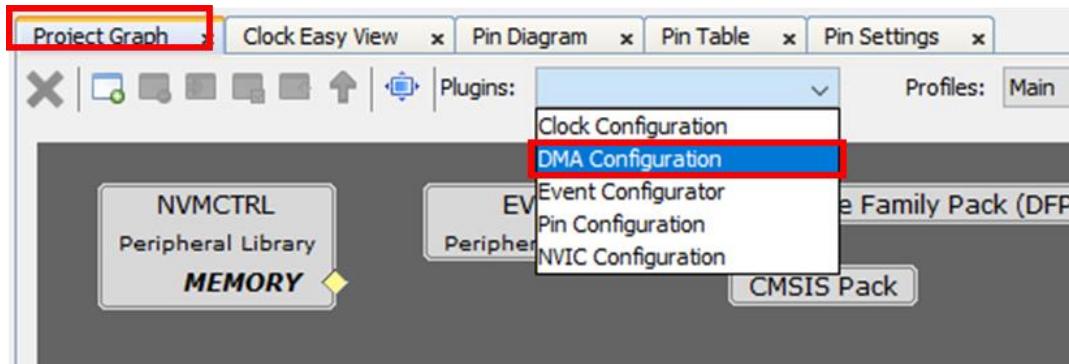
User Configuration

User	Channel Number	Remove User
ADC_START	CHANNEL_0	☒

Add User

STEP 4: Configure DMA

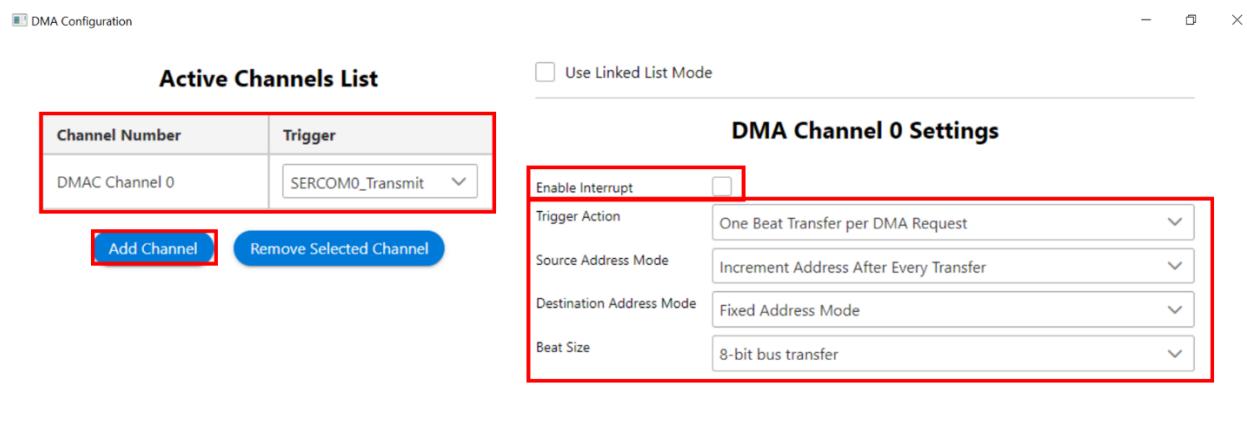
Step 4a: Launch DMA Configurator by going to **Project Graph** tab in MPLABX IDE and then selecting **Plugins > DMA Configuration**.



Step 4b: Click on the **DMA Settings** tab.

Click on *Add Channel* to add DMA Channel 0. Set the *Trigger* source to SERCOM0_Transmit.

This configures the DMA Channel 0 to transfer application buffer to USART TX register. The DMA transfers one byte from application buffer to USART transmit buffer on each trigger.



Based on the Trigger source, the DMA channel configuration is automatically set by the MCC.

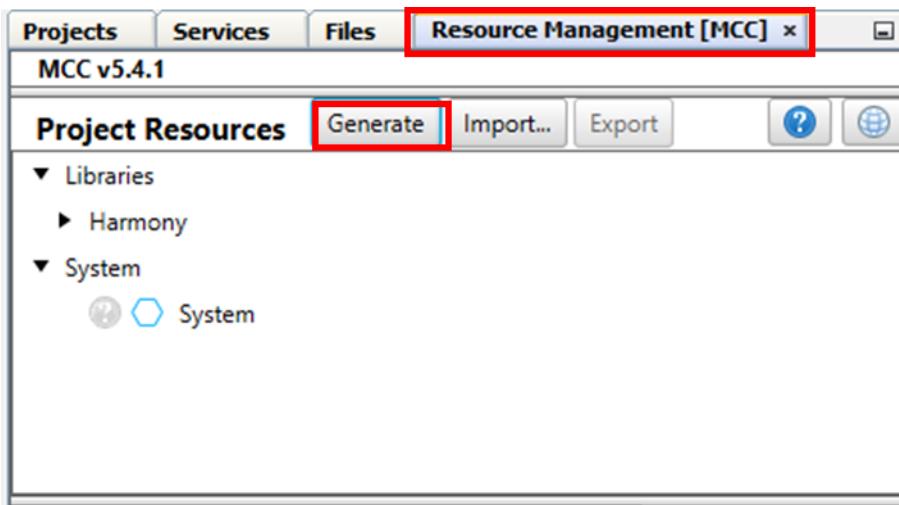
- Trigger Action – Action taken by DMA on receiving a trigger
 - One beat transfer – can be used during a memory to peripheral or peripheral to memory transfer.
 - One block transfer – can be used during memory to memory transfer on a software trigger.
 - One transaction transfer – can be used for memory to memory transfer with linked list to copy multiple blocks on a software trigger.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

- Source Address Mode, Destination Address Mode –
Select whether to increment Source/Destination Address after every transfer.
Automatically set by MCC based on the trigger type.
Example:
 - If the trigger source is USART transmit, then the Source Address is incremented, and the Destination Address is fixed
 - If the trigger source is USART receive, then the Source Address is fixed, and the Destination Address is incremented
- Beat Size – Size of one beat. The default value is 8-bits.
Example:
 - If the SPI peripheral is configured for 16-bit/32-bit mode, then the beat size must be set to 16-bits/32-bits respectively

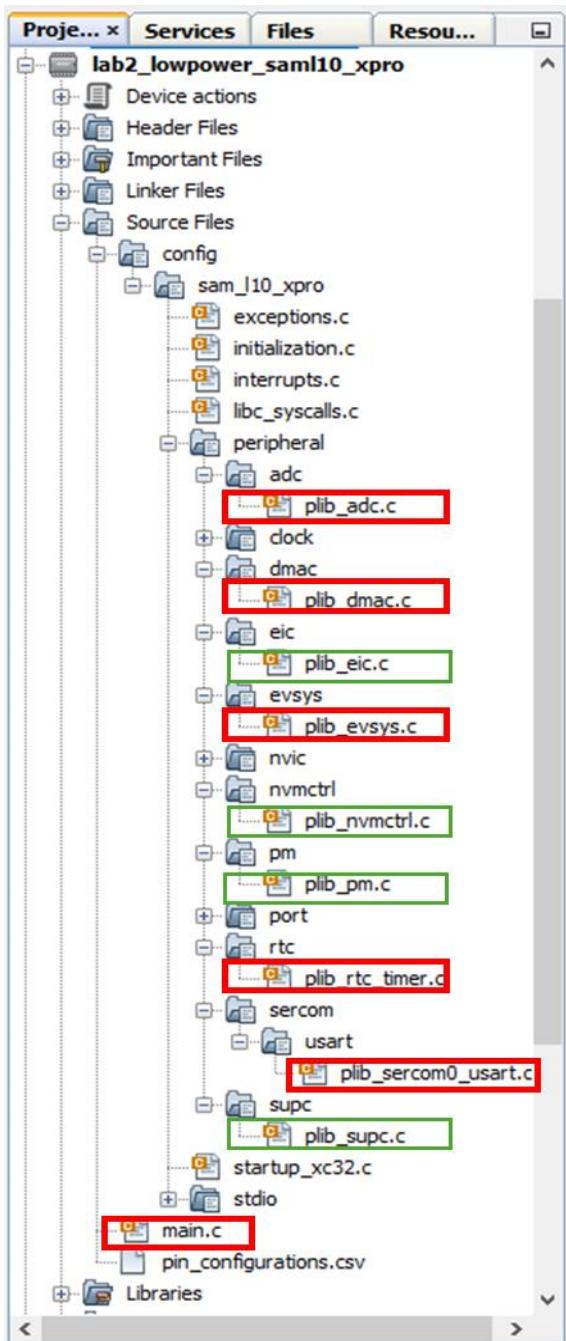
STEP 5: Generate code

Step 5a: Click on the **Generate** button in the Generate icon on Resource 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.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework



Among the generated code notice the source and header files generated for RTC, ADC, SERCOM0-USART, EVSYS and DMAC peripherals. MCC also generates a template main.c file.

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

Step 5b: 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.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

PART 3: Add application code and build the application

STEP 6: Add application code to the project

Step 6a: Open **main.c** file and register an event handler (callback) with the ADC PLIB. The event handler is called by the ADC PLIB when the ADC Window Monitor event occurs. The ADC Window Monitor event occurs when the ADC result is less than the Upper threshold value of 100, which was set when configuring the ADC PLIB.

```
/*Register ADC Window Compare match Event Handler. Here ---> Step #1 */
ADC_CallbackRegister (adcEventHandler, 0);
```

Step 6b: Enable ADC module by calling the *ADC_Enable* API. ADC is used to sample the light sensor when a trigger is received from the Event System Channel 0, every 500 milliseconds.

```
/*Start ADC. Here ---> Step #2*/
ADC_Enable();
```

Step 6c: Start the RTC timer. The event output from RTC is fed as input to the Event System and is used to start the ADC conversion.

```
/*Start RTC Timer. Here ---> Step #3*/
RTC_Timer32Start();
```

Step 6d Add code to put the CPU to STANDBY Sleep mode. After a call to the *PM_StandyModeEnter* API, the CPU will enter IDLE Sleep mode. The ADC Window Monitor interrupt will bring the CPU out of the Sleep mode.

```
/*Enter standby mode. Here ---> Step #4*/
PM_StandyModeEnter();
```

Step 6e: Add code to implement the ADC Window Monitor event handler. The event handler is called when the ADC result of the light sensor is greater than the window mode match value (100).

```
/* ADC Window Monitor Event Handler. Here -----> Step #5 */
static void adcEventHandler(ADC_STATUS status, uintptr_t context)
{
    if (status & ADC_STATUS_WINMON)
    {
        adcResult = ADC_ConversionResultGet();
        isADCDWinMonitorEvent = true;
    }
}
```

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

Step 6f: Add code to transfer data to DMA on ADC window generation, and again make the CPU go to sleep mode.

```
if (isADCWinMonitorEvent == true)
{
    isADCWinMonitorEvent = false;

    sprintf((char*)uartTxBuffer, "%02d\r\n", adcResult);

    /*Submit DMA transfer from memory to USART to display the latest
     *temperature value and toggle the LED at the same time. Here ---> Step #5 */

    status=DMAC_ChannelTransfer(
        DMAC_CHANNEL_0,
        uartTxBuffer,
        (const void *)&(SERCOM0_REGS->USART_INT.SERCOM_DATA),
        strlen((const char*)uartTxBuffer)
    );

    PM_StandbyModeEnter ();
}
```

Step 6f: Increment the button press value each time SW0 is pressed and interrupt is generated.

```
/* Handler for button switch interrupt using EIC peripheral */
static void EIC_User_Handler (uintptr_t context)
{
    /* Handle interrupt for button press */
    /* Increment button interrupt flag whenever button press is detected. Here ---> Step #6*/
    Button_press++;
}
```

Step 6g: Handle the Button_press value and make it go to idle or off mode based on the number of presses.

```
/*Handle the button press and change mode accordingly. Here --->Step #7*/
switch(Button_press)
{
    case 1:
        PM_IdleModeEnter();
        break;

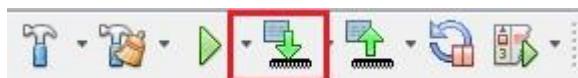
    case 2:
        PM_OffModeEnter();
        break;
}
```

STEP 7: Build and Run the Application

Step 7a: Clean and build your application by clicking on the “Clean and Build” button as shown below.

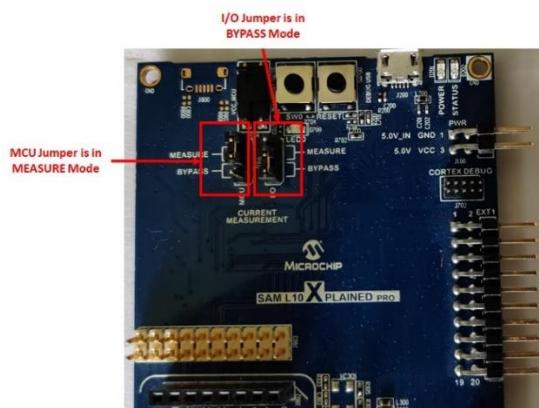


Step 7b: Program your application to the device, by clicking on the “Make and Program” button as shown below.



Results:

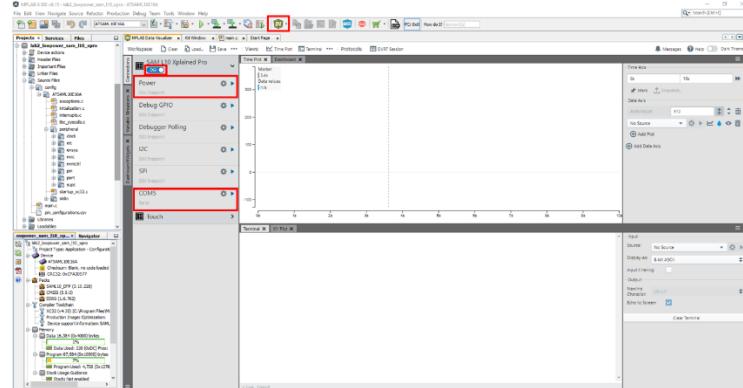
- Ensure that the jumpers for Current Measurement on the SAM L10 Xplained Pro are set to **MEASURE** for the MCU and **BYPASS** for the I/Os.



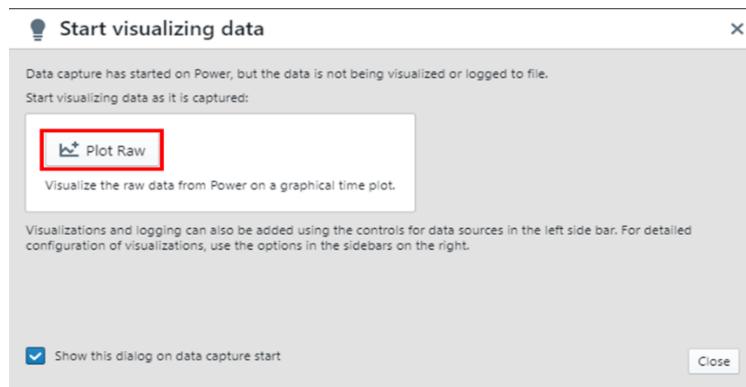
- Open the **Data Visualizer** application from your PC and select the connected **SAM L10 Xplained Pro** board on the **DGI Control Panel**, then click on **Connect**.

Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

The Data Visualizer will then start searching for protocols from the SAM L10 Xplained Pro board through the EDBG.

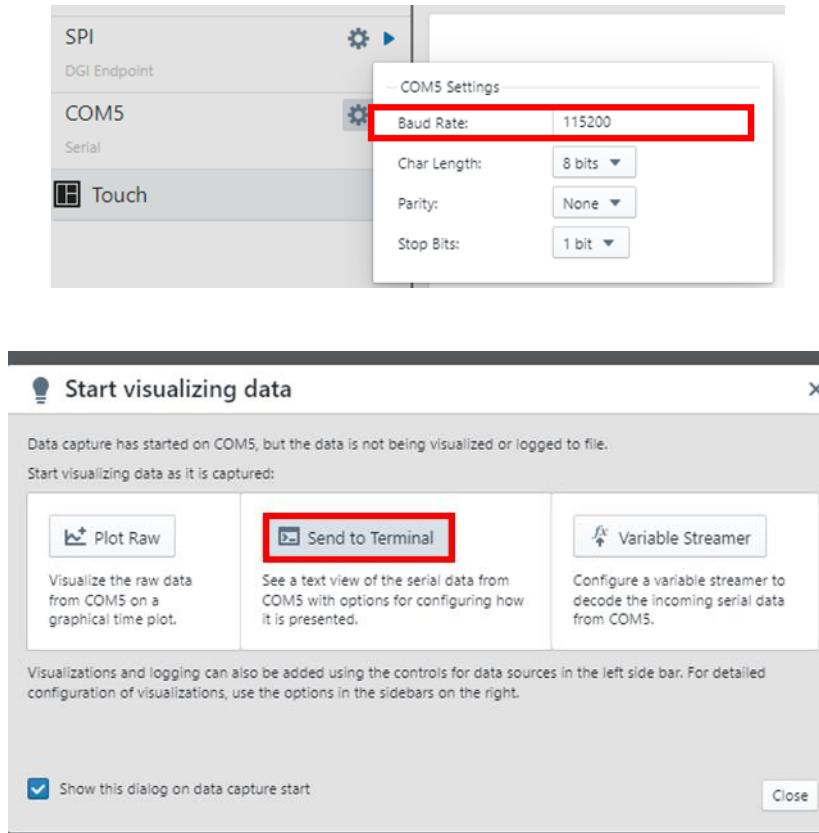


On clicking power icon, a pop up is displayed to plot values, Select plot raw option and close it.

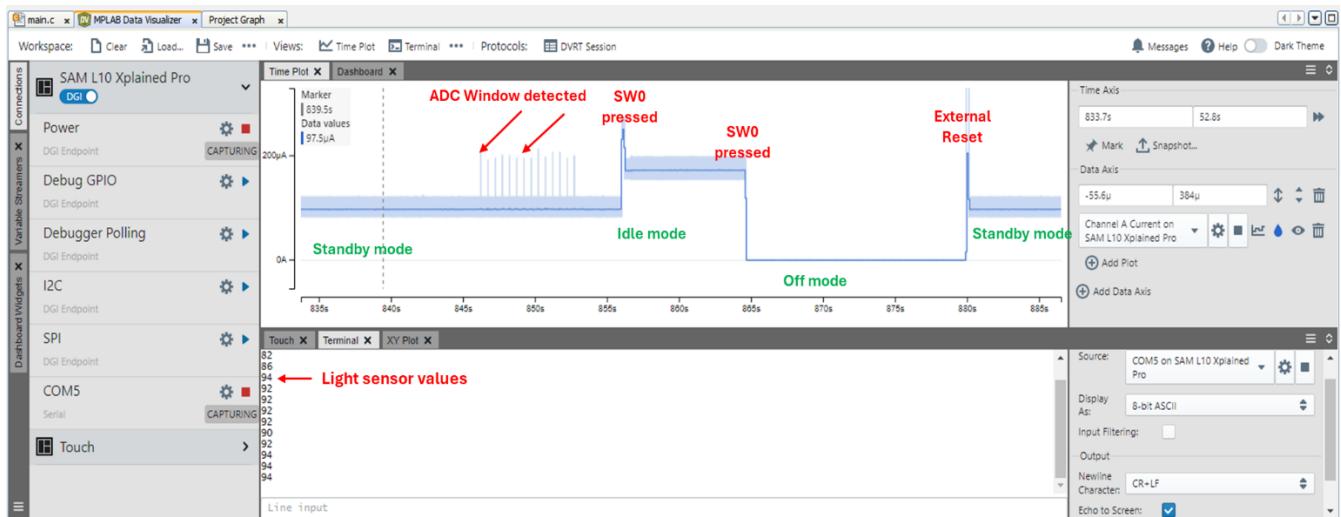


Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

For the COM Port select as follows and set the baud rate as 115200, from 9600 as per the SERCOM0 settings and set to send the data to the terminal window as shown below.



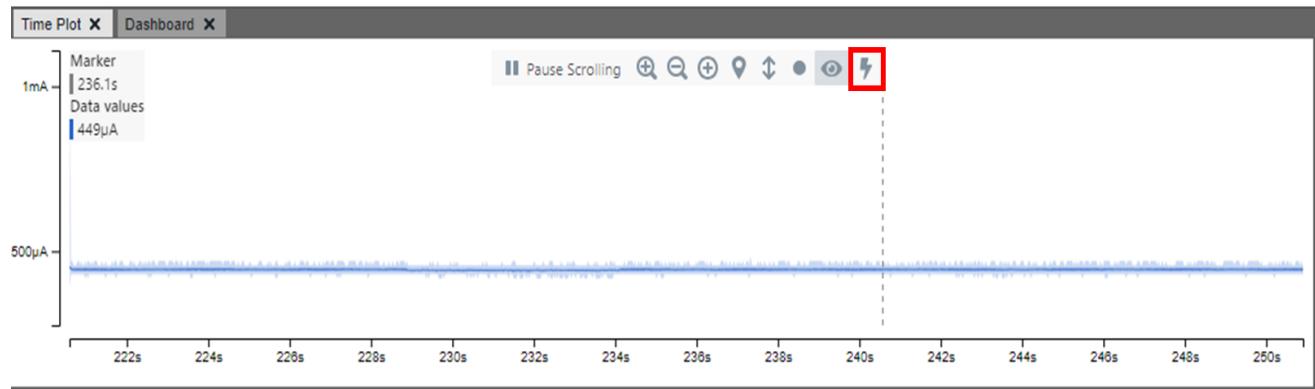
- The output is observed as follows:



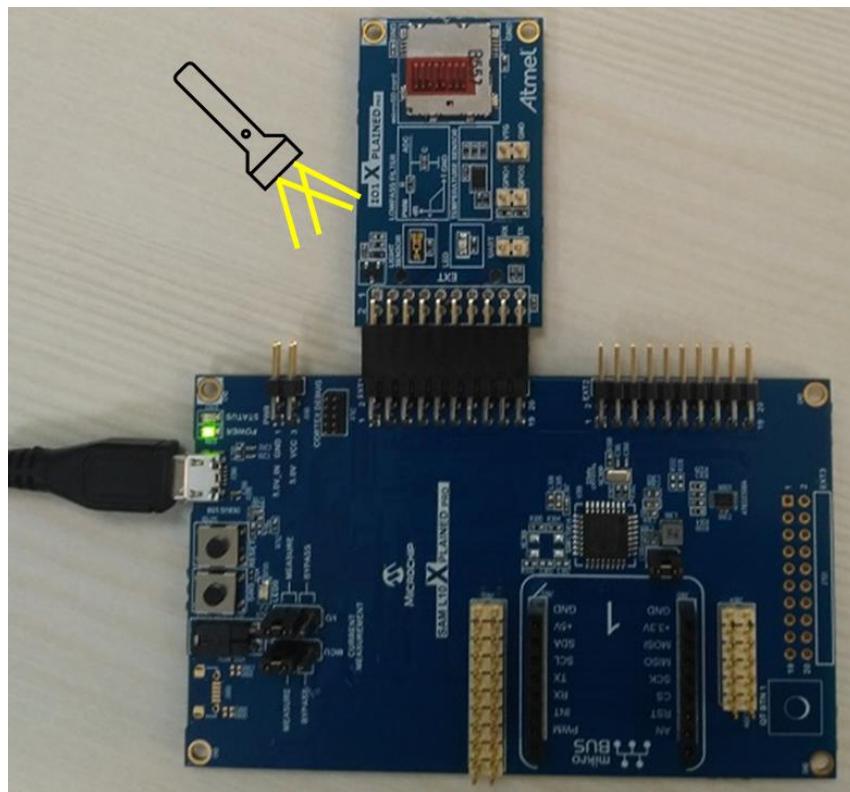
Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

The average value is considered when measuring the power consumption of the device because the instant value is not stable.

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

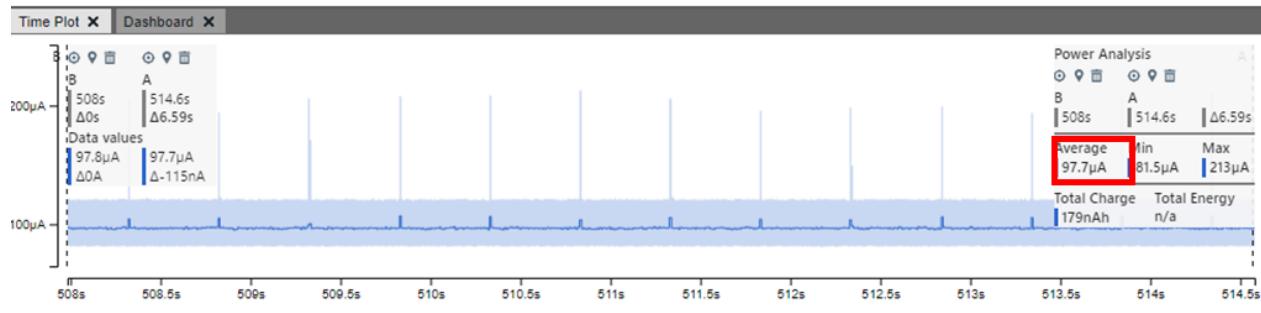


The image below shows the device in Standby mode with its measured power consumption.



Lab Manual for Creating Low Power Embedded Applications with 32-bit MCUs/MPUs using the MPLAB® Harmony Software Framework

You can observe small peaks that illustrate the 500 milliseconds Real-Time Clock (RTC) timer expiry. Flash torch on the light sensor of the I/O Xplained Pro board to print the temperature on the terminal and observe the power consumption of the device.



The average value is considered when measuring the power consumption of the device because the instant value is not stable. Then, the power consumption of the device in Standby mode is _____ µA.

Summary:

In this lab, we have described and illustrated how to create a project from scratch with MPLAB x IDE and get an application up and running.

We have also demonstrated different features such as the Data Visualizer tool which can be used to display graphs of user data using an easy-to-use interface.

After completing this hands-on, you should now:

- Know how to print debug messages on a Virtual COM Port using the Xplained Pro Embedded Debugger Virtual COM Port interface.
- Know how to use Data Visualizer using the Xplained Pro Embedded Debugger DGI Interface.
- How to configure a device in sleep mode.
- How to use an Event System to drive events received from the peripherals without CPU intervention
- You also learned how to configure a device to work in Sleep modes and measure Power consumption.