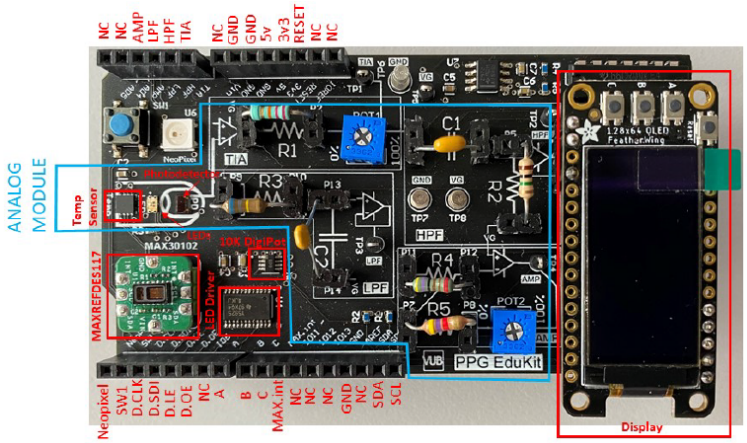
Lab 3: Plot raw PPG signal over UART with PSOC6

Introduction

This lab helps the user to get a first interaction with the PPG EduKit platform, both in terms of hardware and software configuration. The goal is to measure the plethysmograph signal using the analog PPG module and send the data through UART in order to visualize the signal.

The PPG EduKit platform is shown below. The module can be used with the CY8CPROTO-063-BLE board using the bridge adaptor provided. The analog PPG module includes one RGB LED, a photodetector, a transimpedance amplifier, a high pass filter, a low pass filter and an amplification stage. In this lab the PPG signal is acquired from the last amplification stage, thus the board has to be configured with the proper component values.



Objectives

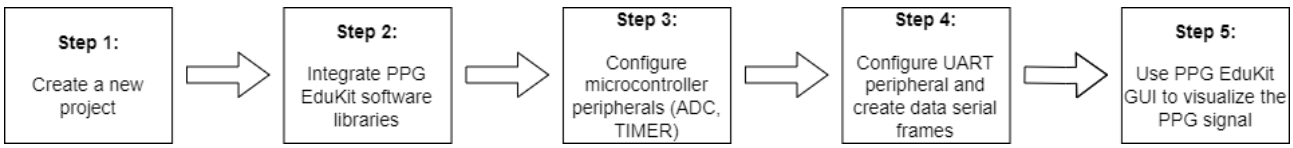
After completing this lab, you will be able to:

- Understand how to set up a PSOC Creator project
- Use and modify the PPG EduKit software libraries
- Configure microcontroller peripherals in PSOC Creator (ADC, Timer, IRQ, UART)
- Use the PPG EduKit GUI application

Procedure

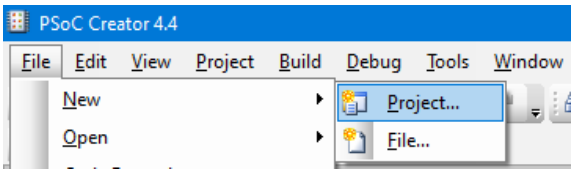
This lab is separated into steps that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

The lab includes 5 primary steps: create a project using PSOC Creator for CY8CPROTO-063-BLE board, integrate the software libraries in the newly created project (LED driver, digital potentiometer, etc), configure the peripherals in order to read the PPG signal, configure UART communication and send the data to PC, and finally run the PPG EduKit GUI application to read and visualize the data.

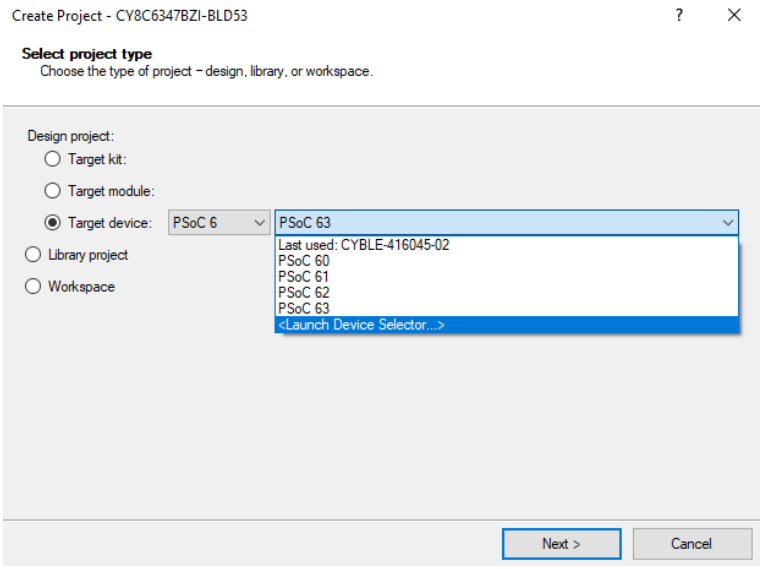


1 Creation of the Project

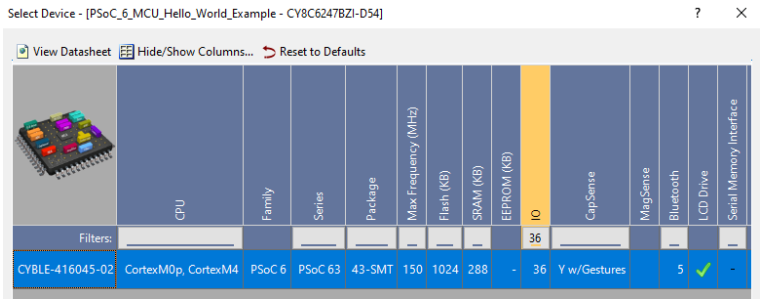
- Open PSoC Creator
- Go to File → New → Project



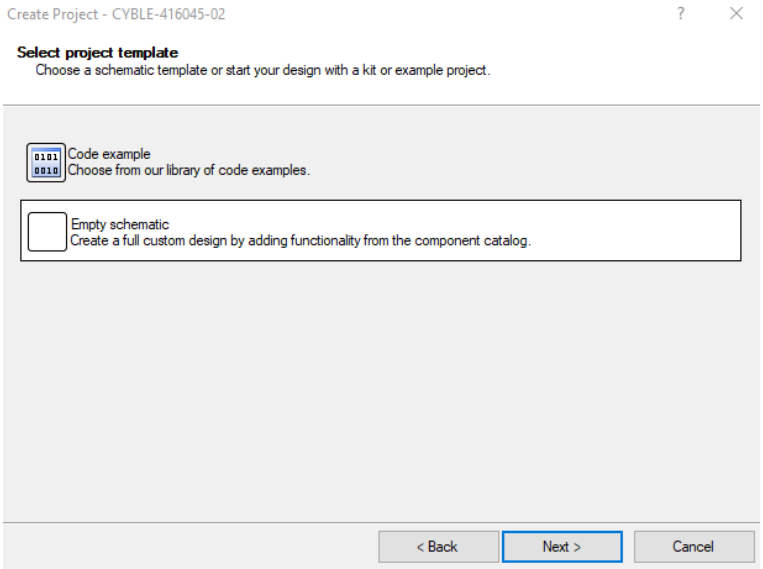
- Select the target device → PSOC6 → <Launch Device Selector>



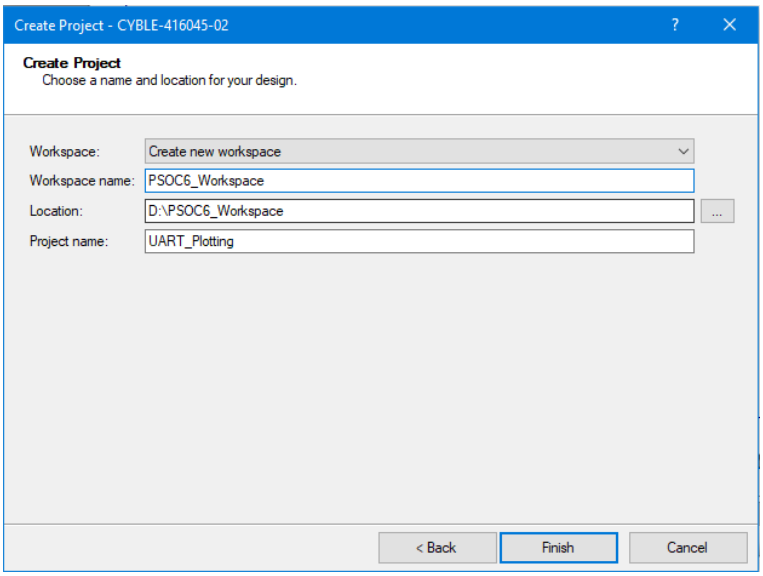
- Select the **CYBLE-416045-2** Device



- Select "Empty schematic" and click "Next"



- In the "Select target IDE(s)" window, click "Next"
- Create a new Workspace and the project name is **UART_Plotting**



2 Integrate PPG EduKit software libraries

The next step is to integrate the software drivers that are provided in the PPG EduKit package. As seen in the application note, some external components need to be interfaced using SPI or I2C. Libraries are provided to speed up and to facilitate the development of PPG related applications, such that the user can focus on PPG signal acquisition or other types of algorithms.

In order to plot the PPG signal using the PPG EduKit GUI the user has to integrate the following libraries: TLC5925 (LED driver library), AD5273 (digital potentiometer library), custom PPG, serial frame (UART) and utils library. Each library consists of a source file (.c) and a header file (.h) and can be found in PPG EduKit package.

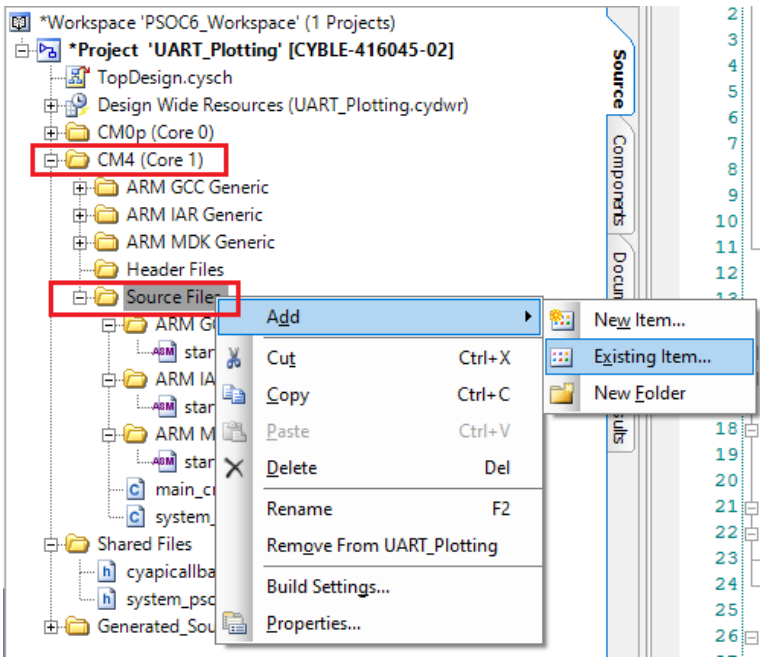
Utils library

The library imports all the common libraries for all the PPG EduKit libraries and defines a common error handler.

```
23
24  #include <stdint.h>
25  #include <stdbool.h>
26  #include <stdlib.h>
27  #include <string.h>
28  #include <math.h>
29
```

```
81  void HandleError(void)
82  {
83      /* Disable all interrupts. */
84      __disable_irq();
85
86      /* Infinite loop. */
87      while(1u) {}
88  }
```

- Import the source file (utils.c) in CM4 (Core1) → Source Files

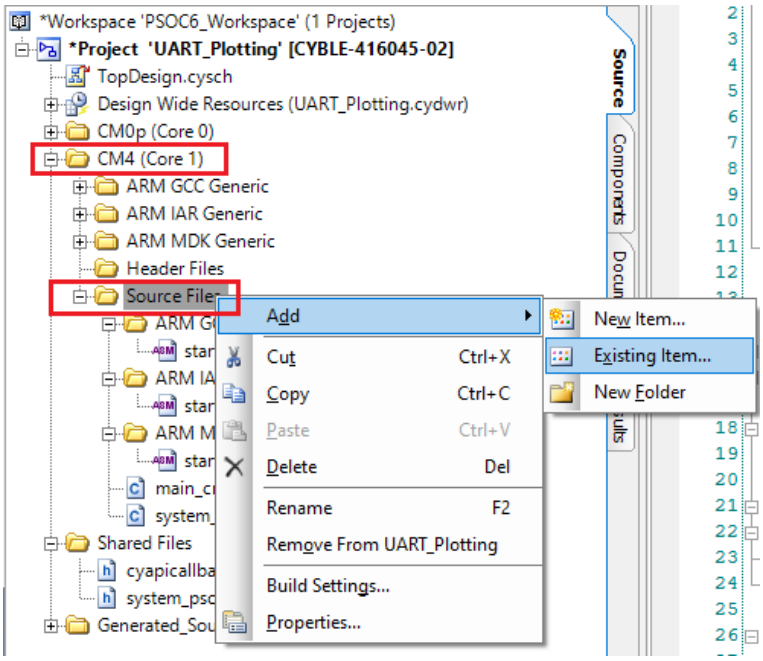


- Repeat the process for the header file (utils.h). Import the file in CM4 (Core1) → Header Files

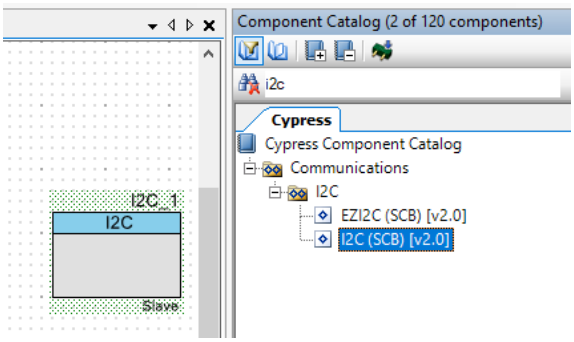
AD5273 driver

AD5273 is a 64-position, one-time programmable (OTP) digital potentiometer that employs fuse link technology to achieve permanent program setting and can be configured through an I2C-compatiblle interface.

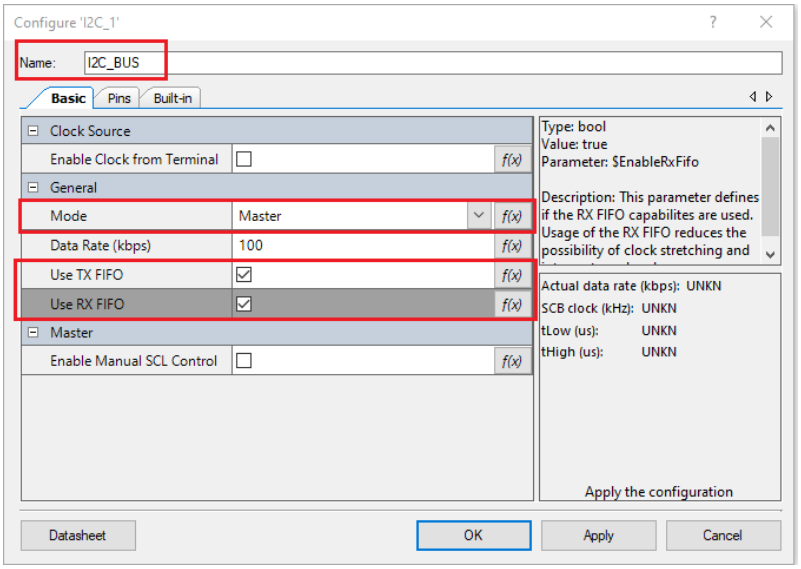
- Import the source file (AD5273.c) in CM4 (Core1) → Source Files



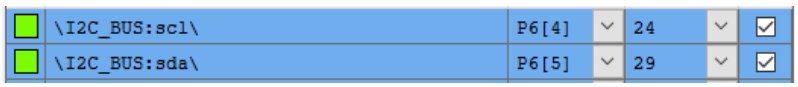
- Repeat the process for the header file (AD5273.h). Import the file in CM4 (Core1) → Header Files
- Go to the **TopDesign** schematic. In the component catalog (right panel at the right of the screen), write **I2C** and drag and drop the component into the schematic.



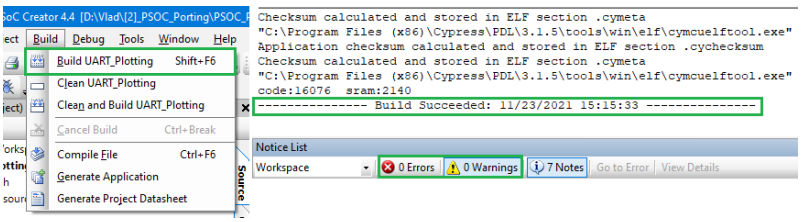
- Configure the I2C component by double click on it. Rename the component as **I2C_BUS**, set the mode as **master** and enable **TX/RX FIFO** buffers.



- Go to Design Wide Resources → Pins and assign the I2C SCL and SDA pins as follows:



- Build the project to check if there is any error.

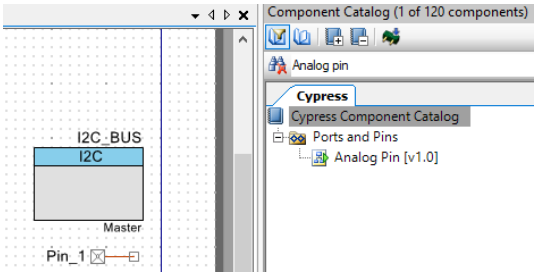


- Take a look at the application programming interface (AD5273.c) and try to understand driver functions.

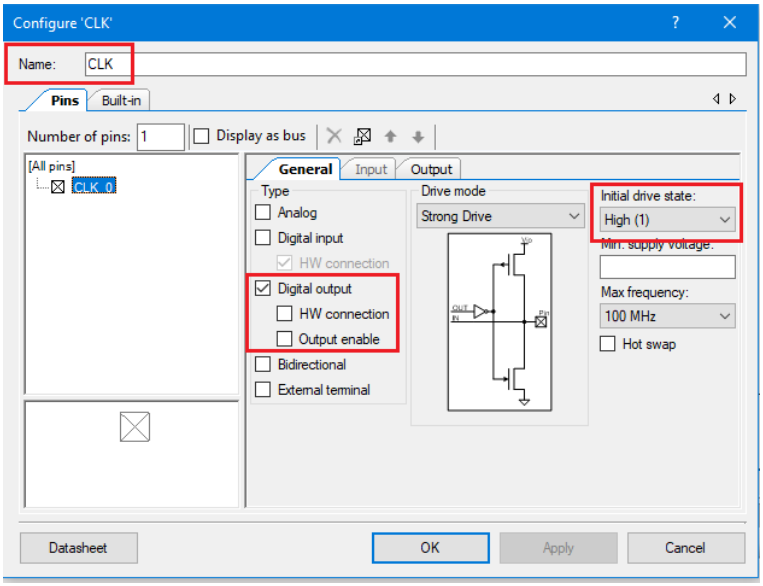
TLC5925 driver

TLC5925 low-power 16-channel constant-current LED sink drivers to contain a 16-bit shift register and data latches, leading to converted serial input data into a parallel output format. The serial data is transferred into the device via **SDI** line at every rising edge of the **CLK** line. **LE** line latch the serial data in the shift register to the output latch, and the **OE** line enables the output drivers to sink current.

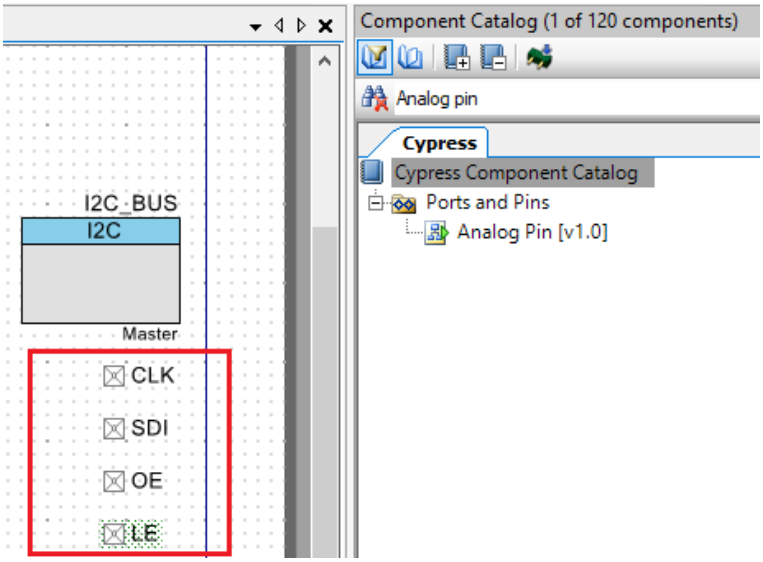
- Import the source file (TLC5925.c) in CM4 (Core1) → Source Files
- Repeat the process for the header file (TLC5925.h). Import the file in CM4 (Core1) → Header Files
- Go to the **TopDesign** schematic. In the component catalog (right panel at the right of the screen), write **Analog pin** and drag and drop the component into the schematic.



- Configure the Pin component by double click on it.



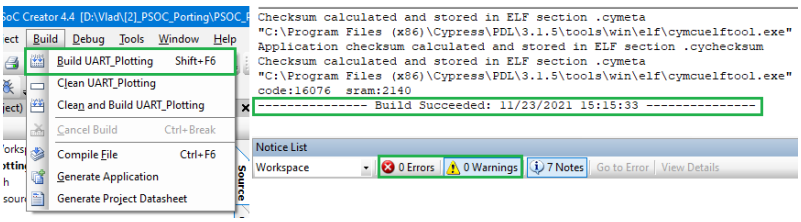
- Repeat the step for SDI, OE and LE pins.



- Go to Design Wide Resources → Pins and assign the pins as follows:

CLK	P12[6]	34	☑
LE	P12[7]	35	☑
OE	P7[2]	22	☑
SDI	P0[5]	2	☑

- Build the project to check if there is any error.

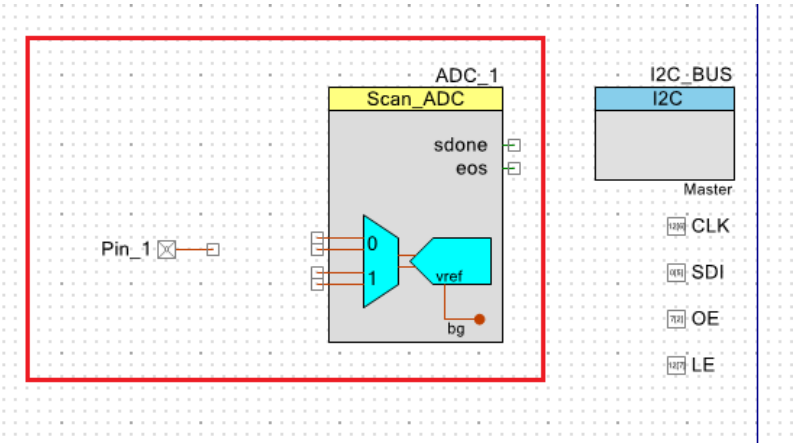


- Take a look at the application programming interface (TLC5925.c) and try to understand driver functions.

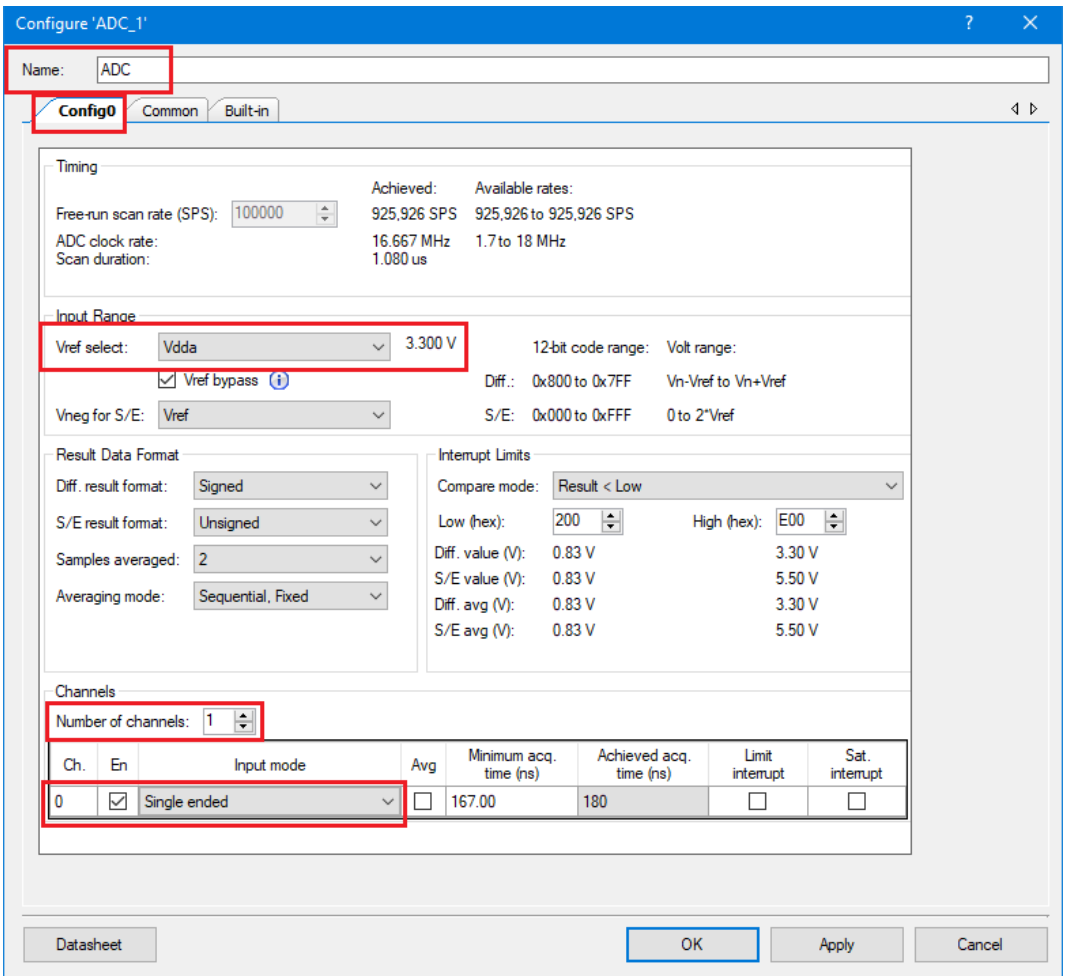
Custom PPG driver

The Custom PPG library is intended to include the user defined algorithms such as HR, SpO2 or digital filters. In this lab, the library includes only the ADC reading routine using an timer triggered ISR.

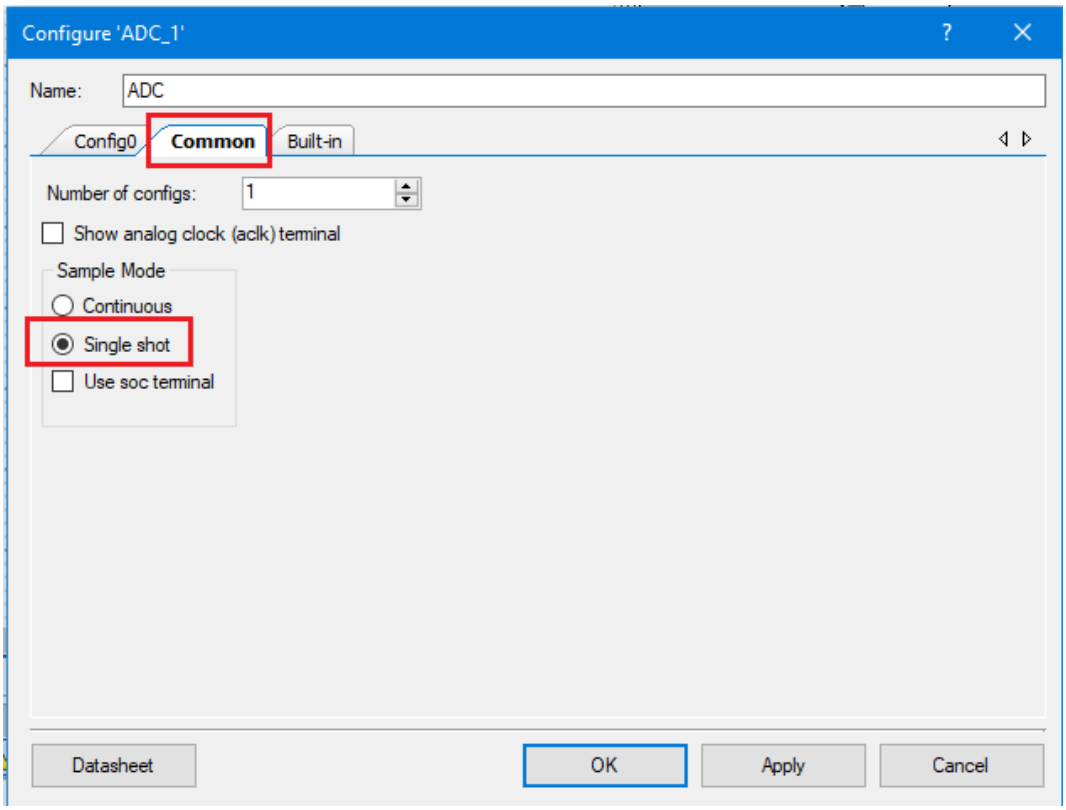
- Go to the **TopDesign** schematic. In the component catalog (right panel at the right of the screen), write **ADC** and drag and drop the component into the schematic. Search for **Analog pin** and drag and drop the component into the schematic.



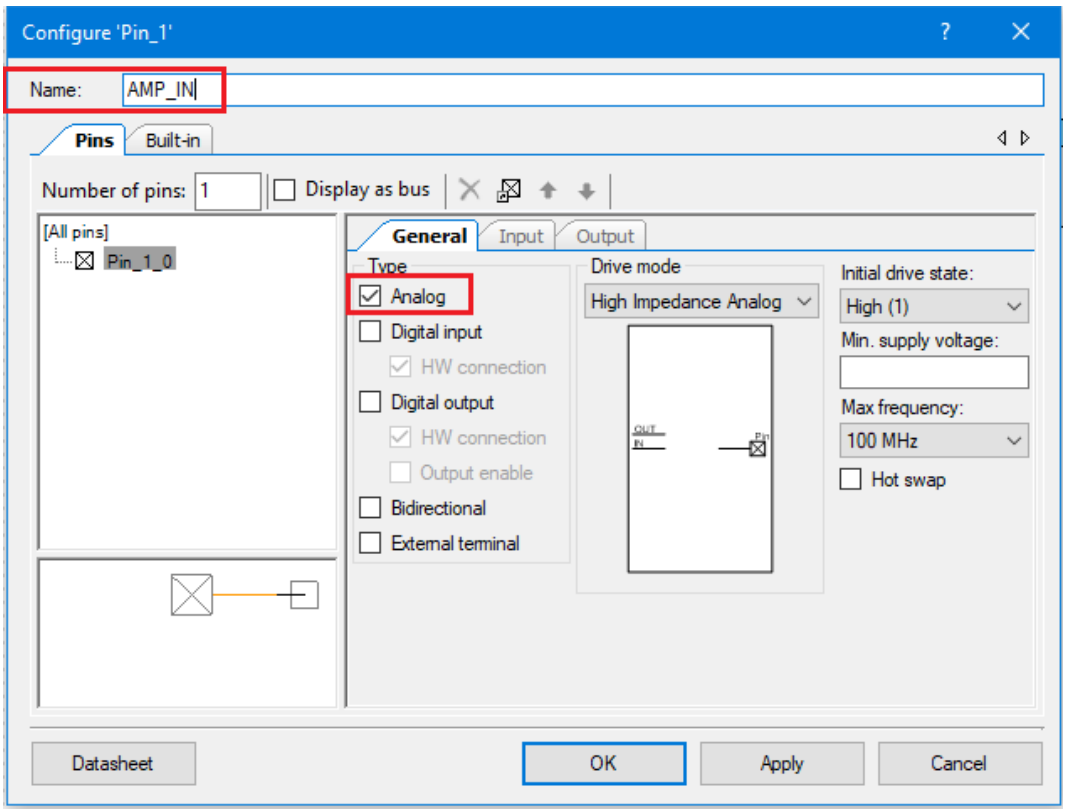
- Configure the ADC component by double click on it.



- Configure the ADC sample mode as single shot mode.



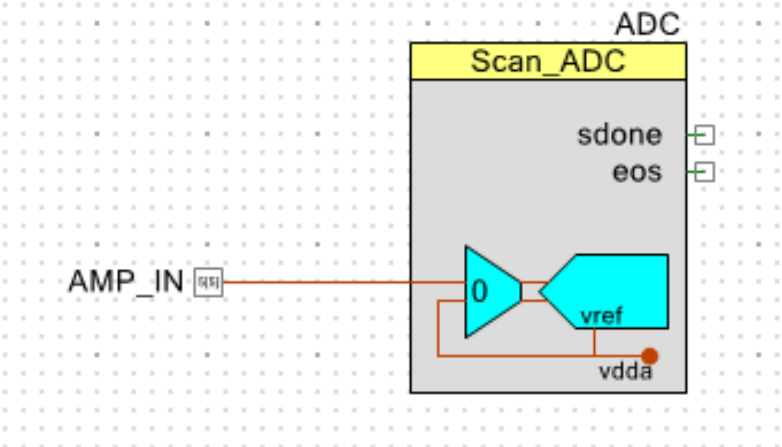
- Configure the Pin component by double click on it.



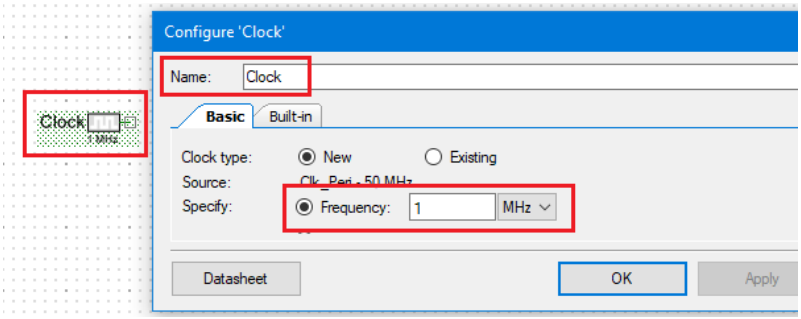
- Go to Design Wide Resources → Pins and assign the **AMP_IN** pin as follows:

	Name	Port	Pin	Lock
	AMP_IN	P5[5]	36	☑
	CLK	P12[6]	34	☑
	LE	P12[7]	35	☑
	OE	P7[2]	22	☑
	SDI	P0[5]	2	☑

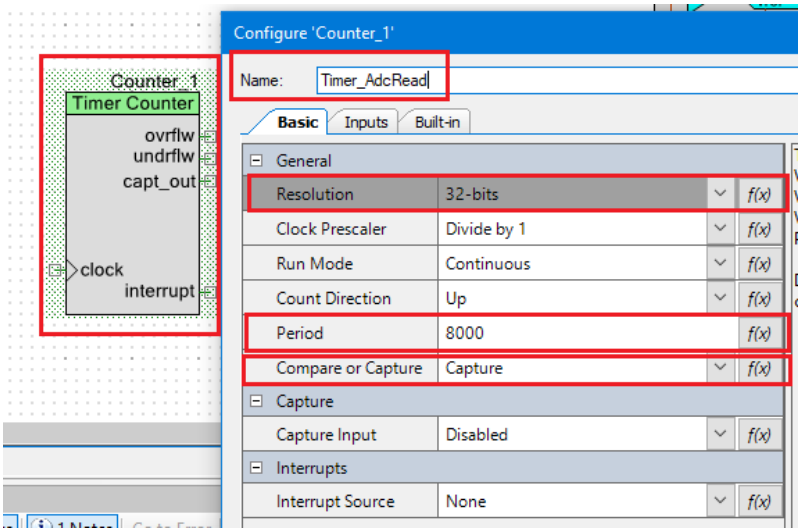
- Connect the analog pin to the ADC (press W to place a wire).



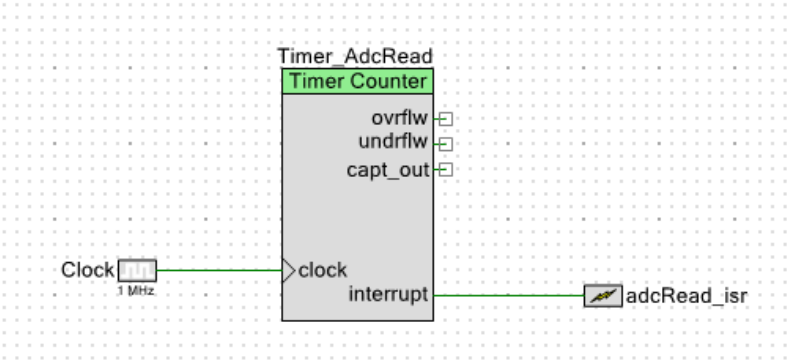
- Search for **Clock** and drag and drop the component into the schematic.
- Configure clock to 1 MHz



- Search for **Timer counter** and drag and drop the component into the schematic.
- Configure the counter.



- Search for **Interrupt** and drag and drop the component into the schematic. Rename the component as **adcRead_isr**.
- Interconnect the components

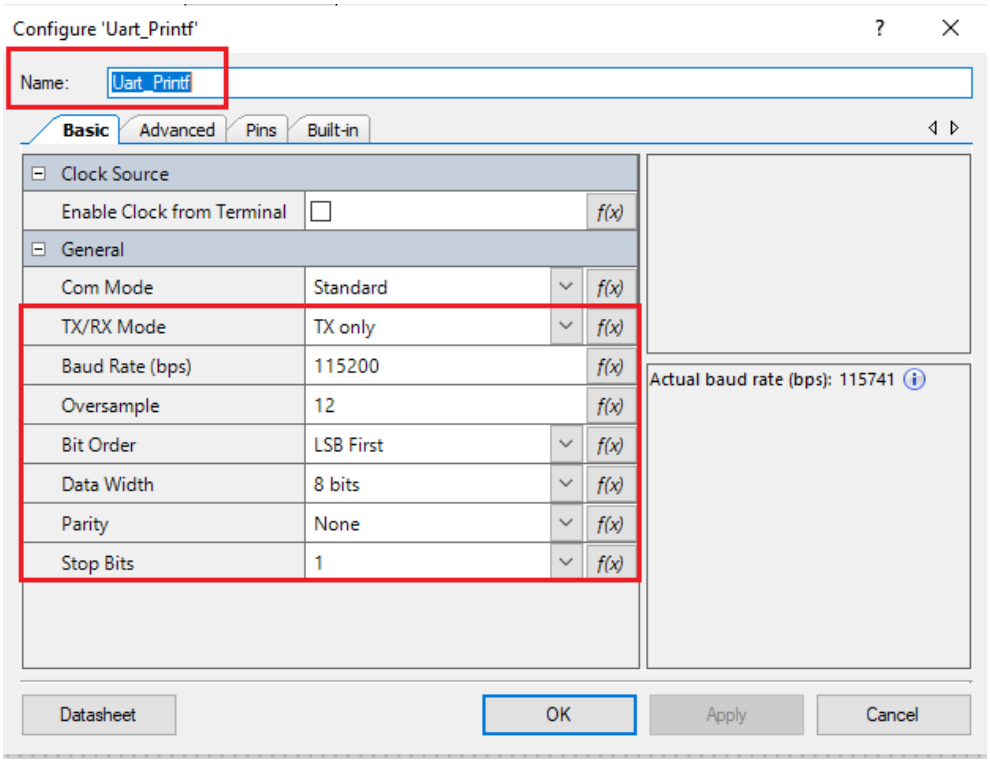


- Import the source file (custom_ppg.c) in CM4 (Core1) → Source Files
- Repeat the process for the header file (custom_ppg.h). Import the file in CM4 (Core1) → Header Files
- Build the project to check if there is any error.
- Take a look at the application programming interface (custom_ppg.c) and try to understand driver functions.

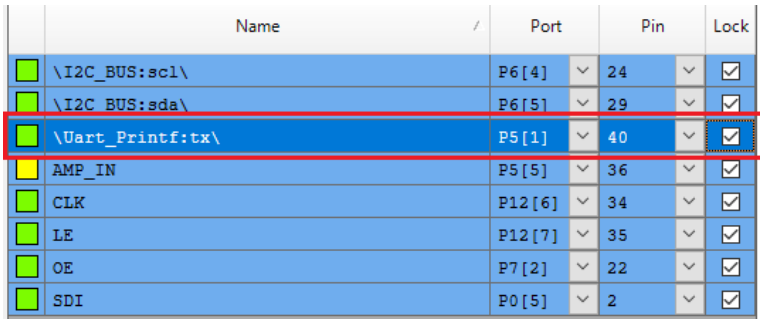
Serial Frame library

This library is a UART wrapper used to interface the PSOC with the PPG EduKit GUI. UART Communication stands for Universal asynchronous receiver-transmitter. It is a dedicated hardware device that performs asynchronous serial communication. It provides features for the configuration of data format and transmission speeds at different baud rates.

- Go to the **TopDesign** schematic. In the component catalog (right panel at the right of the screen), search for **UART** and drag and drop the component into the schematic.
- Configure the component.



- Go to Design Wide Resources → Pins and assign the UART pin as follows:



- Import the source file (serial_frame.c) in CM4 (Core1) → Source Files
- Repeat the process for the header file (serial_frame.h). Import the file in CM4 (Core1) → Header Files
- Build the project to check if there is any error.
- Take a look at the application programming interface (serial_frame.c) and try to understand driver functions.

3 Create the main program

The main program should merge together all the libraries in such a way to reach the goal of printing the PPG signal over UART. The main function can be found in **main_c4.c**.

- Go to **main_c4.c** and include the header file for each software driver.

```
/*=========================================================================================================
 *                                          INCLUDE FILES
 * 1) system and project includes
 * 2) needed interfaces from external units
 * 3) internal and external interfaces from this unit
 *  ========================================================================================================*/

/* @brief Include PSOC generated files */
#include "project.h"

/* @brief Include custom libraries for PPG EduKit */
#include "utils.h"
#include "AD5273.h"
#include "TLC5925.h"
#include "custom_ppg.h"
#include "serial_frame.h"
```

- Declare the global variables. The serial frame command is represented by the **frameParams_t** structure type. The variable is passed to the **createSerialFrame** function in order to create the desired data frame. The function returns the address location of the data frame buffer to be sent, and **gpFrame** pointer is used to point to that memory location.

```
/*=========================================================================================================
 *                                          GLOBAL VARIABLES
 *  ========================================================================================================*/

/* @brief Serial frame command */
frameParams_t frameParam;

/* @brief Pointer to a serial frame newly created*/
uint8_t *gpFrame = NULL;
```

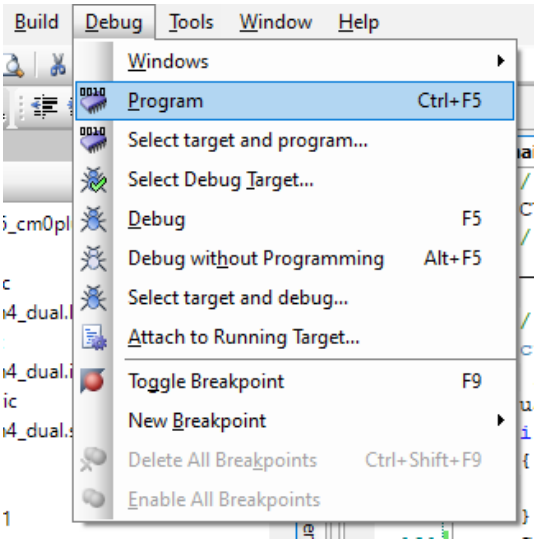
- The main function starts with the initialization of **adcRead_isr** interrupt by setting the priority and the afferent interrupt vector. Then, the interrupts are enabled and the UART peripheral is initialized. The first component that is initialized is the AD5273 digital potentiometer, which is used for setting the current for the LED driver. The **AD5273_Init** function initializes the I2C peripheral, sets the data rate and the clock frequency. The LED current is set to 5 mA by calling the **TLC5925_SetCurrent_mA** function. The function translates the current value to a 6 bit value that represents the digital potentiometer resistance and writes the value over I2C. Having the external resistance configured, the LED driver can be configured to enable the infrared LED by calling **TLC5925_enableIR**. The PPG signal can be measured right after the LED is turned on. Therefore, the ADC peripheral is started and the timer used to trigger the ADC conversion is initialized and started.

```c
int main(void)
{
    /* Assign ISR routines */
    CUSTOM_PPG_AssignISR_AdcRead();
    /* Enable global interrupts. */
    __enable_irq();

    /* UART initialization status */
    cy_en_scb_uart_status_t uart_status ;
     /* Initialize UART operation. Config and Context structure is copied from Generated source. */
    uart_status  = Cy_SCB_UART_Init(Uart_Printf_HW, &Uart_Printf_config, &Uart_Printf_context);
    if(uart_status != CY_SCB_UART_SUCCESS)
    {
        HandleError();
    }
    Cy_SCB_UART_Enable(Uart_Printf_HW);
    /* Init digital potentiometer */
    AD5273_Init();
    /* Set 5 mA current for the LED driver */
    TLC5925_SetCurrent_mA(5);
    /* Enable IR LED */
    TLC5925_enableIR();
    /* Start ADC and ADC Conversion */
    ADC_Start();
    /* Start timer used for ADC reading */
    CUSTOM_PPG_InitAndStartTimer_AdcRead();

    /* Main infinite loop */
    while(1)
    {
        /* Wait for the buffer to fill up.  */
        while(CUSTOM_PPG_BUFFER_LENGTH != CUSTOM_PPG_bufferHead);

        /* Prepare serial frame structure */
        memset(&frameParam, 0x00, sizeof(frameParam));
        frameParam.frameType = CHANNEL_DATA;
        frameParam.params.wavelength = IR_CHANNEL;
        frameParam.tissueDetected = TRUE;
        gpFrame = createSerialFrame((uint16_t *) &CUSTOM_PPG_bufferIR[0], (CUSTOM_PPG_BUFFER_LENGTH) * 2, &frameParam);
        /* Sens serial frame */
        sendFrame(gpFrame);

        /* Signal IRQ for new data */
        bBufferProcessed = TRUE;
        CUSTOM_PPG_bufferHead = 0UL;
    }
}
```

- The ISR handler for ADC reading has a frequency of 125 Hz and fills the **CUSTOM_PPG_bufferIR** as long as the buffer is not full and the application sends the entire buffer over UART.

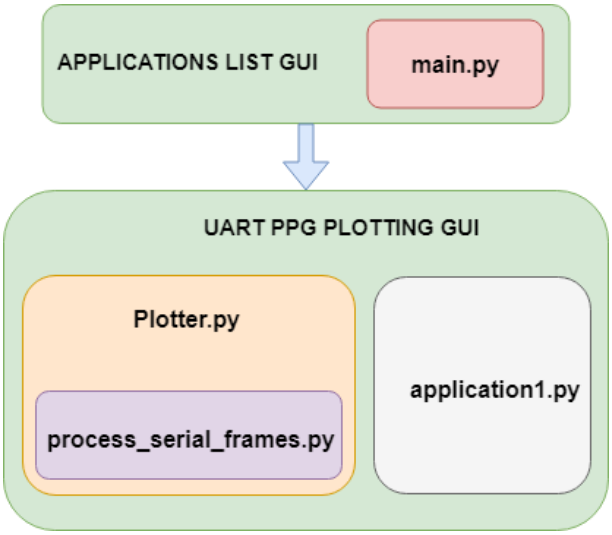
```c
/*=============================================================================================
 *                                      GLOBAL VARIABLES
 *  ============================================================================================*/

/* @brief Custom PPG buffers */
volatile uint16_t CUSTOM_PPG_bufferIR[CUSTOM_PPG_BUFFER_LENGTH];

/* @brief Points to the head of the buffer (last sample) */
volatile uint16_t CUSTOM_PPG_bufferHead;

/* @brief The application should set the buffer to TRUE when the buffer is processed by the main application */
volatile bool bBufferProcessed = TRUE;
/*******************************************************************************
 * Function: CUSTOM_PPG_InterruptHandler_AdcRead
 *
 * Description: ISR routine for Timer_AdcRead ovf.
 *******************************************************************************/
void CUSTOM_PPG_InterruptHandler_AdcRead(void)
{
    /* Clear the terminal count interrupt */
    Cy_TCPWM_ClearInterrupt(Timer_AdcRead_HW, Timer_AdcRead_CNT_NUM, CY_TCPWM_INT_ON_TC);
    ADC_StartConvert();

    if((CUSTOM_PPG_BUFFER_LENGTH != CUSTOM_PPG_bufferHead) && (TRUE == bBufferProcessed))
    {
        if(ADC_IsEndConversion(CY_SAR_RETURN_STATUS) != 0)
        {
            CUSTOM_PPG_bufferIR[CUSTOM_PPG_bufferHead] =  ADC_GetResult16(ADC_CHANNEL_0_INV_AMP);

            CUSTOM_PPG_bufferHead++;
        }
    }
    else
    {
        bBufferProcessed = FALSE;
    }

}
```

- The main loop includes a busy waiting as long as the PPG buffer is not full. When the buffer head reaches the maximum buffer size, the frame is created. The frame type is **CHANNEL_DATA**, the wavelength is set to **IR_CHANNEL**, **tissueDetected** flag should be always true since there is no need to signal this in this particular case. The frame is created by passing the buffer address to **createSerialFrame** function along with the frame structure. The size of the buffer should be **CUSTOM_PPG_BUFFER_LENGTH** multiplied by 2 since every sample is represented by 2 bytes (uint16_t). The frame is sent over UART by calling the **sendFrame** function.
- After the data is sent over UART and plotted, the application should signal the IRQ that the buffer was processed and the buffer head should be restarted, such that the buffer can be filled again with new values.
- Build the project and program the target.



4 Run PPG EduKit GUI application

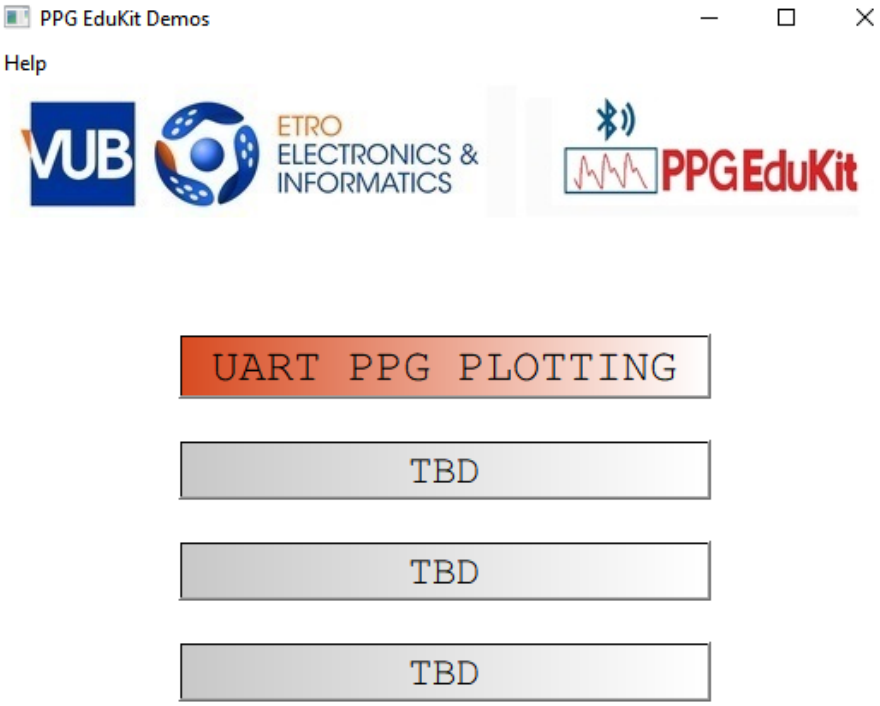
The PPG EduKit GUI is a Python program used to interface the PPG EduKit platform with different types of applications. The user interface is created using QT Designer and stored in **app1_gui.ui** file. The main window loads the requested application in the list. **UART PPG PLOTTING** application has two source files: **application1.py** and **Plotter.py**. The **process_serial_frames.py** file is used to interface the embedded application that communicates over UART with the PC. If the data frame structure is changed (by adding more functionalities), the change should be included both in the **serial_frame.c** file and in **process_serial_frames.py** script.



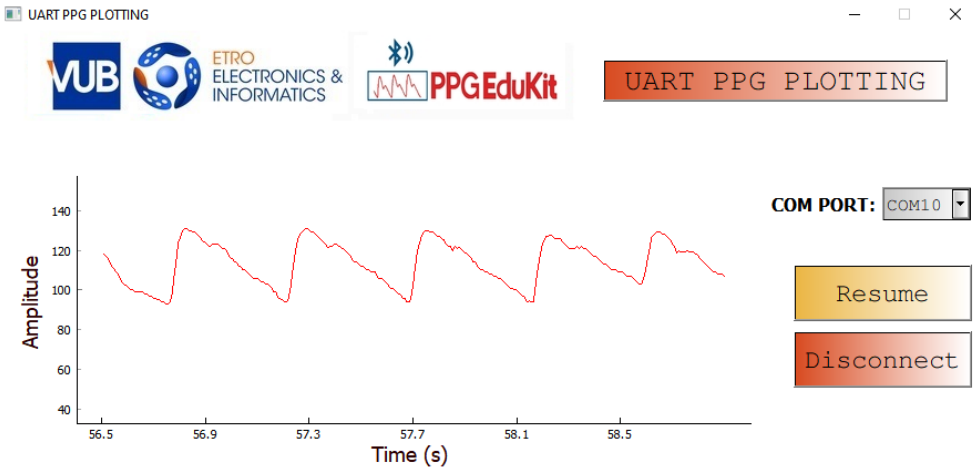
To run the application, one can use just a simple terminal, but the recommendation is to use Spyder IDE or PyCharm.

- Open the command line and go to the location of the PPG EduKit GUI application.

- Execute the command: **pip install -r requirements.txt**. If the pip package is not installed, please refer to [4] in order to set up your environment.
- Open the project in Spyder/PyCharm.
- Run the **main.py** file. In the application list interface, select the **UART PPG PLOTTING** application.



- Select the COM port and click **Connect**. Place your finger over the sensor's surface and visualize the volumetric changes in arterial blood, which is associated with your cardiac activity.



References

[1] Cypress, "PSoC 6 MCU: CY8C63x6, CY8C63x7 Datasheet", https://www.cypress.com/file/385921/, Nov. 2020

[2] Cypress, "PSoC 6 MCU Code Examples with PSoC Creator", https://www.cypress.com/documentation/code-examples/psoc-6-mcu-code-examples-psoc-creator, Mar. 2020

[3] https://www.cypress.com/file/137441/download

[4] https://phoenixnap.com/kb/install-pip-windows