Course: Summer Industrial Training 2015

Lab Report: LAB 7 - Data Acquisition System at PSoC 4

Report By: Ajinkya Padwad
Deep Bhojani

Executive Summary:

This project aimed to implement a basic Data Acquisition System (DAS) with the use of UART and ADC interfaces at the Cypress' CY8CKIT-042 PSoC 4 Pioneer development board along with the basic interface board for input using the PSoC Creator software.

Project Description:

The first step was creating the schematic of the system at the 'Top Design' layout at PSoC Creator. SAR-ADC and the UART modules were added along with the suitable input and output pins from the components menu. Next step was the component configuration after which, the ports were mapped to physical pins. The program code was then compiled into the 'main.c' file and lastly, the interface of the CY8CKIT-042 PSoC 4 Pioneer development board was carried out.

Components used:

Analog input pin

Analog input pin is used to interface with potentiometers, buttons, LEDs, and peripheral sensors such as proximity detectors and accelerometers.

SAR ADC module

The SAR ADC is the component used to access the ADC functionality in PSoC 4.

UART Module-

The UART provides asynchronous communications and can be configured for Full Duplex, Half Duplex, RX only, or TX only modes.

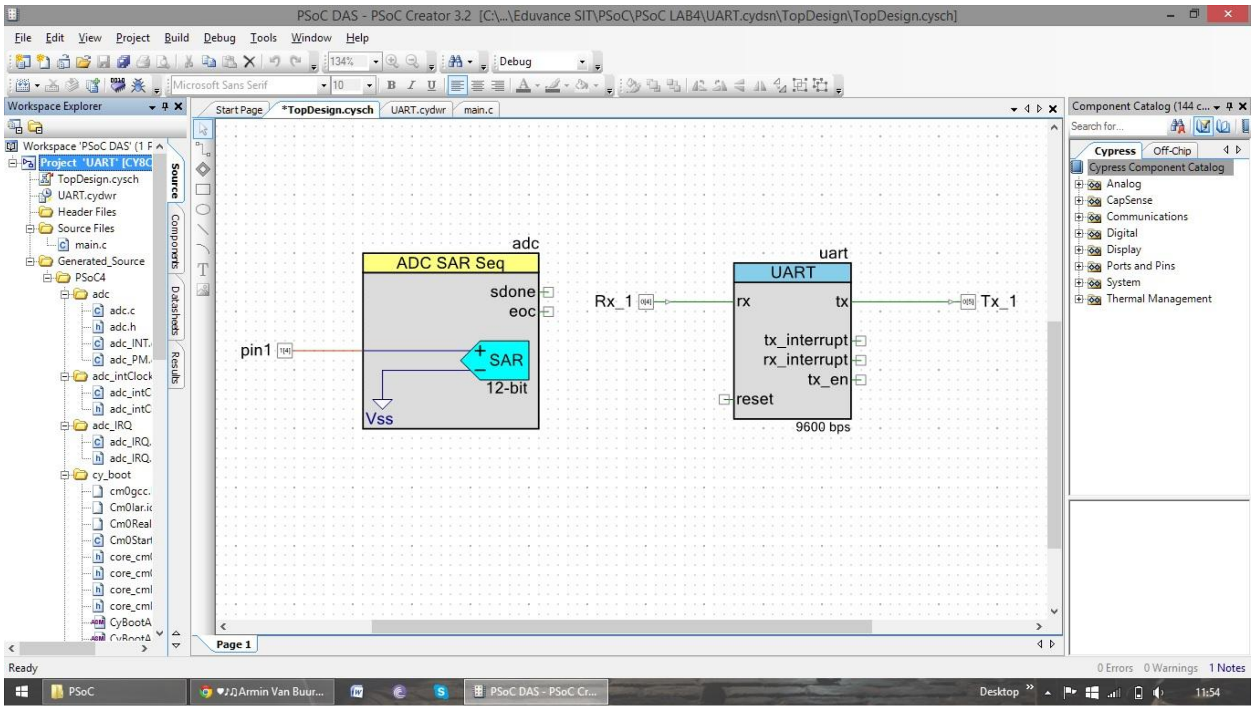
Digital Input Pin (Receiver)

The RX input pin transfers the input serial data from the LDR/ADC on the serial bus, to the microcontroller onboard.

Digital Output pin (Transmitter) -

The TX output pin transmits the output serial data to another device on the serial bus, through the microcontroller.

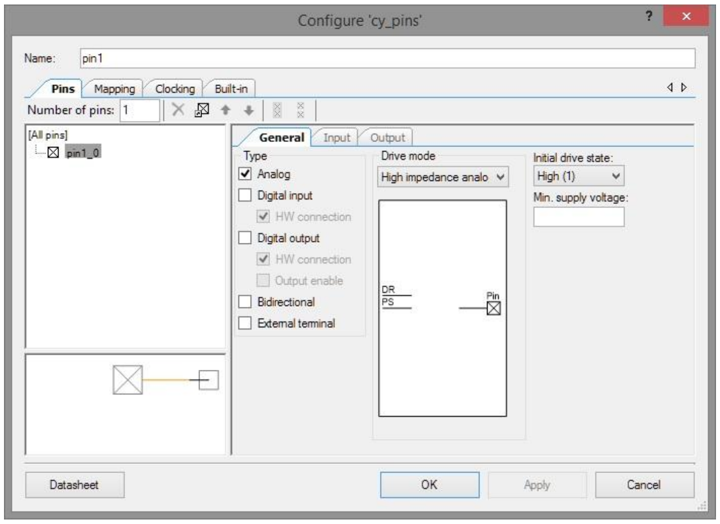
Project Schematic


PSoC DAS - PSoC Creator 3.2 [C:\...\Eduvance SIT\PSoC\PSoC LAB4\UART.cydsn\TopDesign\TopDesign.cysch]
adc
ADC SAR Seq
sdone
eoc
SAR
12-bit
Vss
pin1
Rx_1
uart
UART
rx
tx
tx_interrupt
rx_interrupt
tx_en
reset
9600 bps
Tx_1

Pin Component Configuration and Pin map:

Analog input pin

Drive Mode parameter configures the pin to provide one of the eight available pin drive modes. If the Type is Analog, the default is High Impedance Analog.


Configure 'cy_pins'

Name: pin1

Number of pins: 1

Type: [✓ Analog], Digital input, Digital output, Bidirectional, External terminal

Drive mode: High impedance analo

Initial drive state: High (1)

Min. supply voltage:

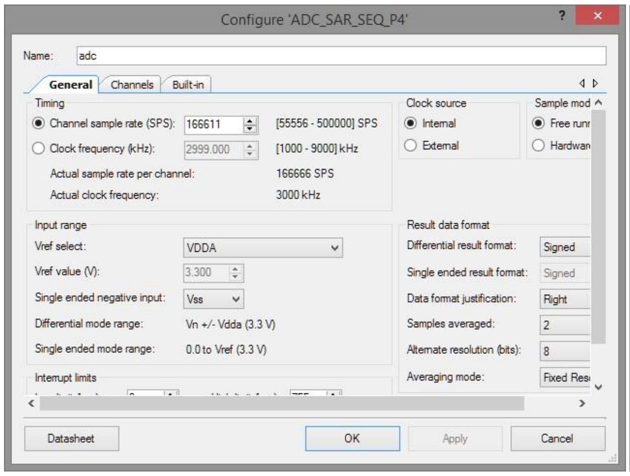
SAR ADC module

Clock rate is automatically calculated based on the number of channels, averaging, resolution, and acquisition time parameters to meet the entered sample rate.

Clock frequency can be anywhere between 1 MHz and 18 MHz (14.508 MHz in CY8C41).

Vref value parameter displays the reference voltage value that is used for the SAR ADC reference.

Channels parameter selects how many input signals are scanned. The maximum number of channels is either 8 or 16 depending on the device.


Configure 'ADC_SAR_SEQ_P4'

Name: adc

General | Channels | Built-in

Timing
- (●) Channel sample rate (SPS): 166611 [55556 - 500000] SPS
- ( ) Clock frequency (kHz): 2999.000 [1000 - 9000] kHz
- Actual sample rate per channel: 166666 SPS
- Actual clock frequency: 3000 kHz

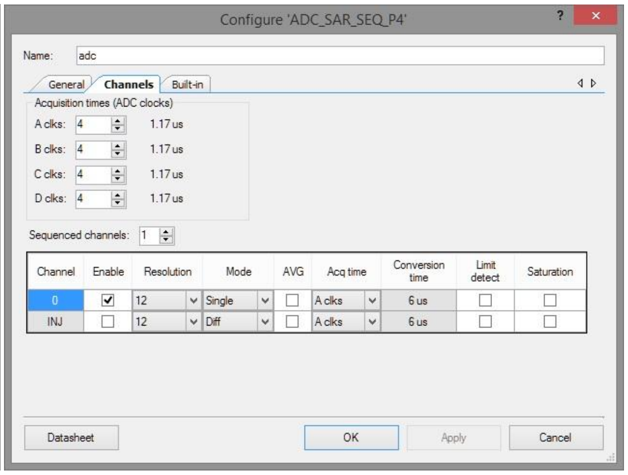
Clock source
- (●) Internal
- ( ) External

Sample mod
- (●) Free runr
- ( ) Hardwar

Input range
- Vref select: VDDA
- Vref value (V): 3.300
- Single ended negative input: Vss
- Differential mode range: Vn +/- Vdda (3.3 V)
- Single ended mode range: 0.0 to Vref (3.3 V)

Result data format
- Differential result format: Signed
- Single ended result format: Signed
- Data format justification: Right
- Samples averaged: 2
- Alternate resolution (bits): 8
- Averaging mode: Fixed Res

Interrupt limits


Configure 'ADC_SAR_SEQ_P4'

Name: adc

General | Channels | Built-in

Acquisition times (ADC clocks)

A clks:	4	1.17 us
B clks:	4	1.17 us
C clks:	4	1.17 us
D clks:	4	1.17 us

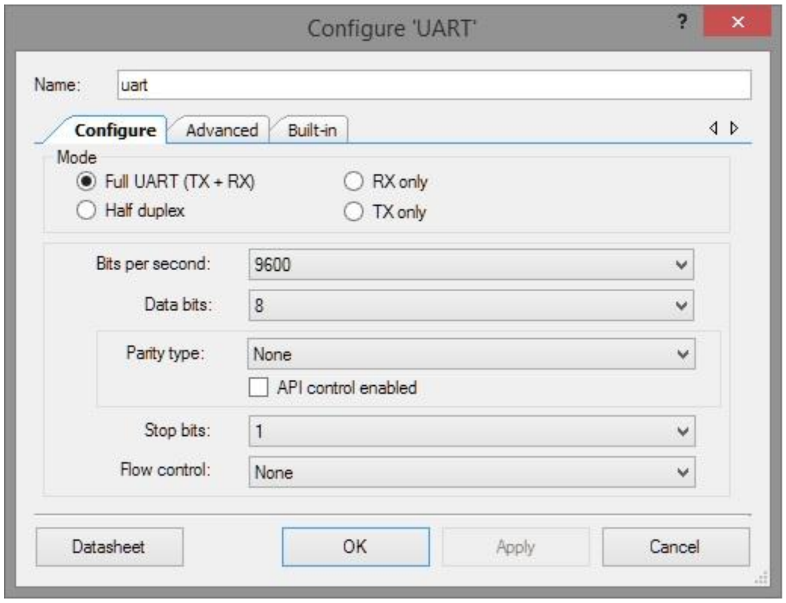
Sequenced channels: 1

Channel	Enable	Resolution	Mode	AVG	Acq time	Conversion time	Limit detect	Saturation
0	✓	12	Single	☐	A clks	6 us	☐	☐
INJ	☐	12	Diff	☐	A clks	6 us	☐	☐

UART Module-

Mode parameter defines the functional components you want to include in the UART. This can be setup to be a bidirectional Full UART (TX + RX) (default), Half Duplex UART (uses half the resources), Receiver (RX Only) or Transmitter (TX Only).

Bits per second parameter defines the baud-rate or bit-width configuration of the hardware for clock generation.


Configure 'UART'

Name: uart

Configure | Advanced | Built-in

Mode

- [✓ Full UART (TX + RX)]
- RX only
- Half duplex
- TX only

Bits per second: 9600

Data bits: 8

Parity type: None

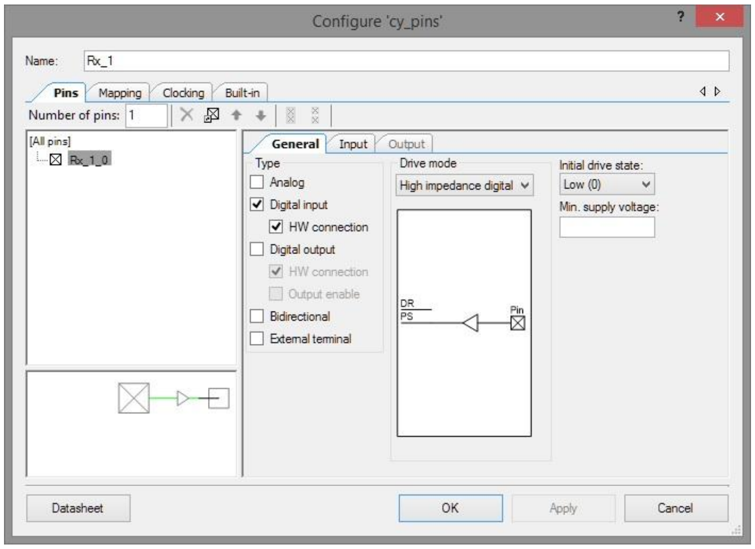
☐ API control enabled

Stop bits: 1

Flow control: None

Digital Input

HW Connection parameter determines whether the digital input terminal for an input pin is displayed in the schematic. It may or may not be API controlled depending upon the option being selected.


Configure 'cy_pins'

Name: Rx_1

Pins | Mapping | Clocking | Built-in

Number of pins: 1

[All pins]
Rx_1_0

General | Input | Output

Type

- ☐ Analog
- ☑ Digital input
  - ☑ HW connection
- ☐ Digital output
  - ☑ HW connection
  - ☐ Output enable
- ☐ Bidirectional
- ☐ External terminal

Drive mode: High impedance digital

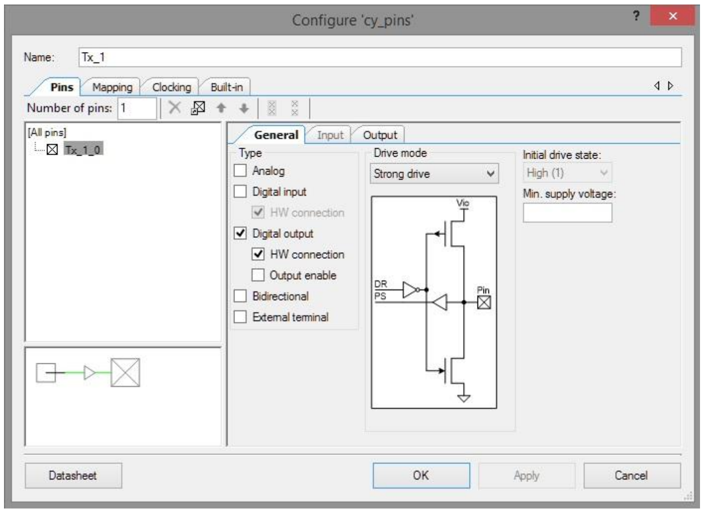
Initial drive state: Low (0)

Min. supply voltage:

Digital Output pin -

Drive Mode This parameter configures the pin to provide one of the eight available pin drive modes.

HW Connection parameter determines whether the digital input terminal for an input pin is displayed in the schematic. It may or may not be API controlled depending upon the option being selected.



Configure 'cy_pins'

Name: Tx_1

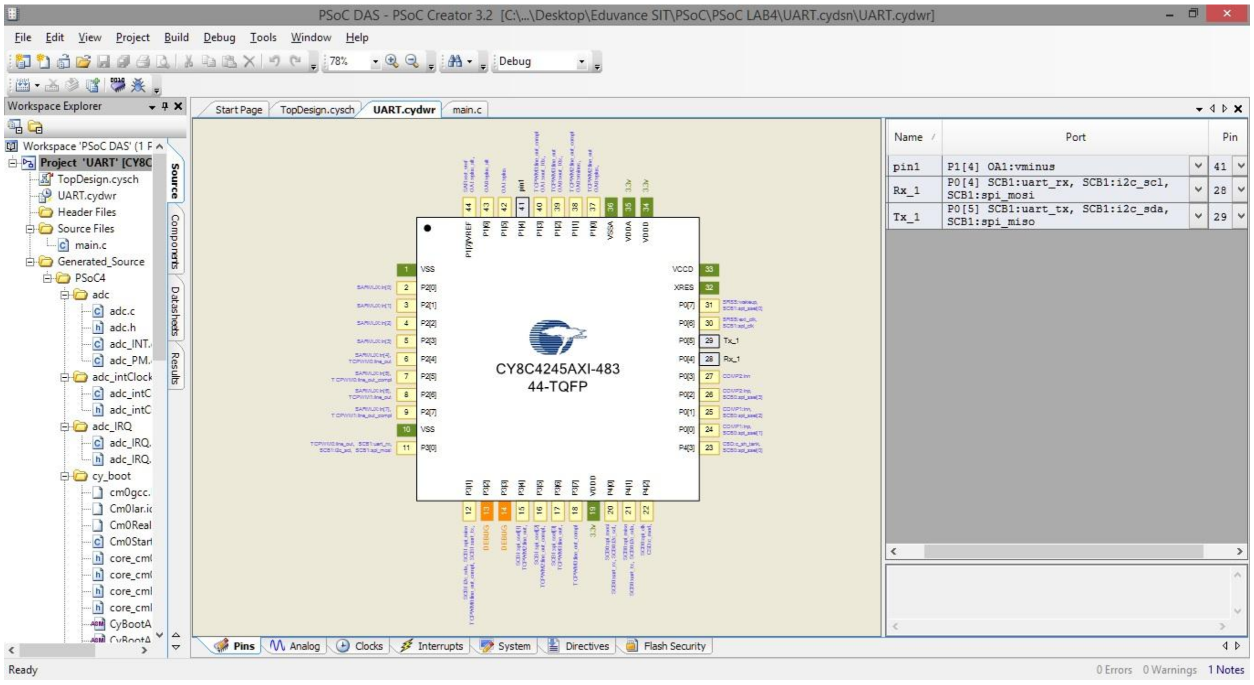
Number of pins: 1

Type: Digital output [✓], HW connection [✓]

Drive mode: Strong drive

Initial drive state: High (1)

CYDWR -



PSoC DAS - PSoC Creator 3.2 [C:\...\Desktop\Eduvance SIT\PSoC\PSoC LAB4\UART.cydsn\UART.cydwr]

Name	Port	Pin
pin1	P1[4] OA1:vminus	41
Rx_1	P0[4] SCB1:uart_rx, SCB1:i2c_scl, SCB1:spi_mosi	28
Tx_1	P0[5] SCB1:uart_tx, SCB1:i2c_sda, SCB1:spi_miso	29

API Description:

- UART API interface -
  It provides asynchronous communications and this component can be configured for Full Duplex, Half Duplex, RX only, or TX only versions. In this project we use the Full Duplex UART (TX + RX) mode for asynchronous transmission.
  Member functions used -

uart_Start()	- Initializes the UART block
uart_PutString()	- Returns a character value as stated by the variable.

- ADC Interface -
  The SAR ADC block is responsible for analog to digital conversion and transmitting the value to the UART receiver. Member functions used -

adc_Start()	- Initializes the ADC block
adc_StartConvert()	- Starts the analog to digital conversion
adc_IsEndConversion(adc_WAIT_FOR_RESULT)	- Waits for the user to put input.
adc_CountsTo_mVolts( )	- Converts the values into millivolts.

Output Observed:

The analog equivalent voltage values from the LDR are converted to digital values which are then serially transmitted and visible on the Terminal software for UART on the PC. The observations were as expected.

Test and Debug:

- ✓ Encountered a major bug regarding the use of "%f" in the 'sprintf()' and learnt the importance of unsigned variables.
- ✓ Error at adc_CountsTo_mVolts( ) while putting the channel value.
- ✓ Tested use of Potentiometer instead of LDR as the analog input.

Learning Outcomes:

- ➢ Learnt the implementation of UART and ADC at PSoC.
- ➢ Use of various APIs mentioned about.
- ➢ Importance of floating and integer type unsigned values.

APPENDIX A

Code:

```
#include <project.h>

#include <stdio.h>

int main()

{

   CyGlobalIntEnable;    // Enable global interrupts.

   uart_Start();         // UART interface initialized.

   adc_Start();          //ADC interface initialized.

   uint16 adcval;        //16-bt unsigned variable for final ADC value.

   char buff[25];        //character type variable for the UART transmission.

   int16 volt;           //16bit signed integer type variable for voltage reading.

While(1)

   {

          adc_StartConvert();                           //Analog to digital conversion started.

          adc_IsEndConversion(adc_WAIT_FOR_RESULT); // Wait for  input

          adcval  =  adc_GetResult16(0);               // Store 16 bit result from ADC into a variable.

          volt =  adc_CountsTo_mVolts( 0 , adcval );   // Calculate equivaent ADC values in millivolts,
                                                                at channel '0'

          sprintf ( buff, "%d  \n \r ", volt);         // Store the voltage value into equivalent
                                                                character value for UART

          uart_PutString(buff);                        // Print the final value using the UART variable.

    }

}
```