PSoC BLE IoT

Report By:

Pratik Suresh Sheth

Executive Summary:

Objective is to design a project using FRDM KL25Z, BLE and IoT. We used LDR and LM35 as sensing element and the accelerometer on the FRDM board and displayed it on the IoT.

Project Description:

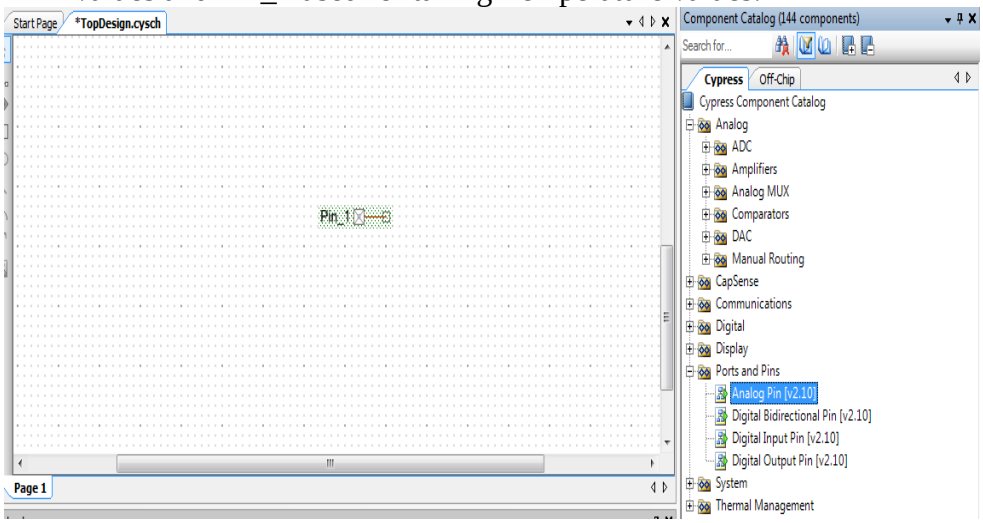
We have created a flow so that we obtain the light intensity and temperature values from the sensing board which consists of LDR and LM35. We also obtain the accelerometer values from the FRDM board.

Components Used:

The following are the components used in this project:

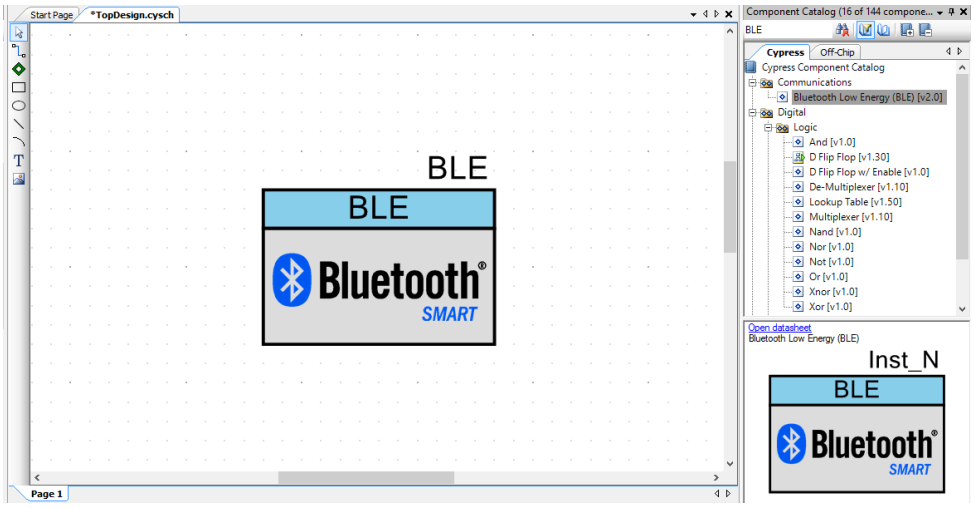
1) Analog pin:

We have used 2 analog pin for taking the values from the sensing board. Pin_1 is used for taking LDR values and Pin_2 used for taking Temperature values.

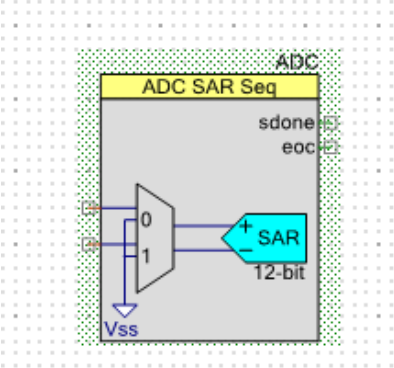


2) Bluetooth Low Energy:

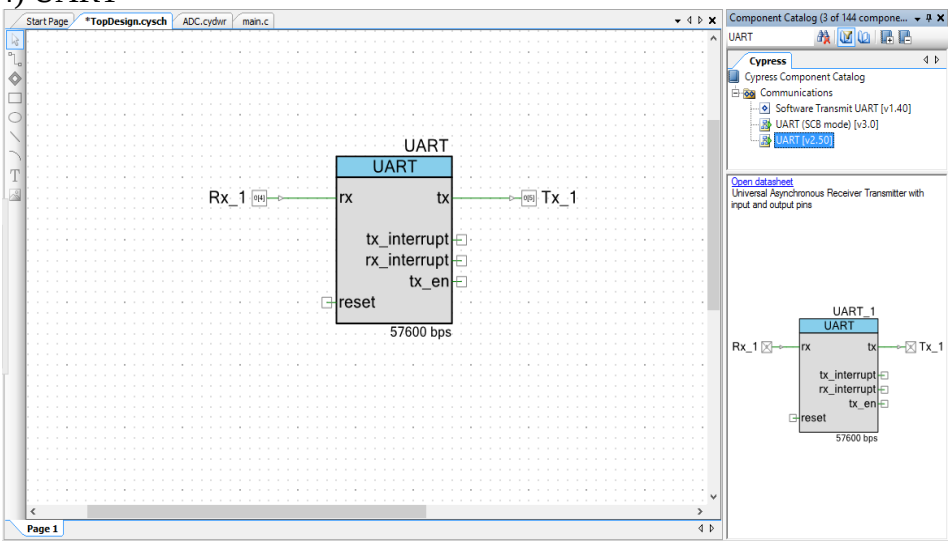
We have also Bluetooth Low Energy Block.



3) Sequencing Successive Approximation ADC:



4) UART

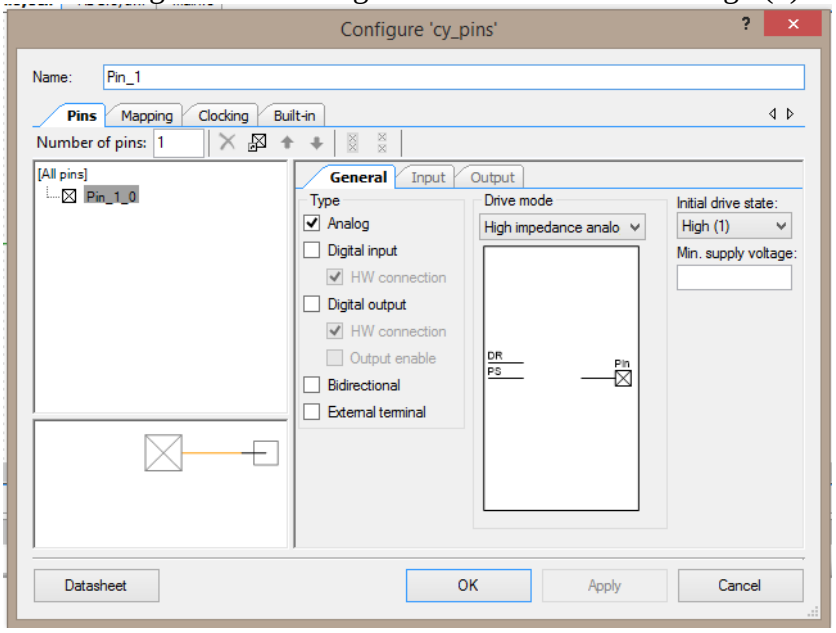


Component Configuration and Pin Map :

The following are the component configuration and pin mapping was done in this project:

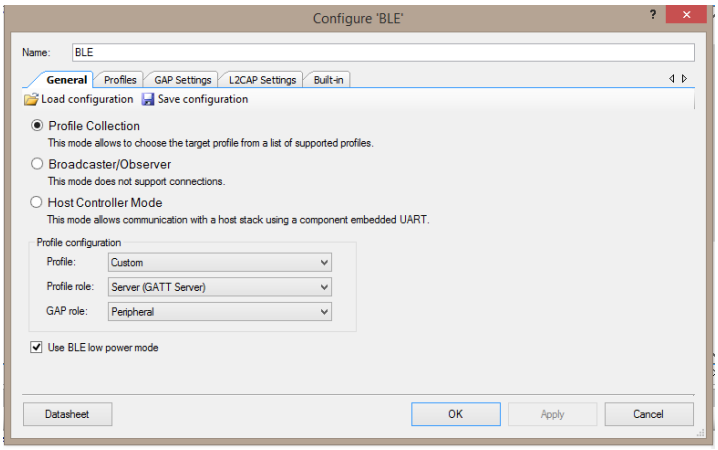
1) Analog pin:

We selected Analog and also configured Initial drive state to High (1).



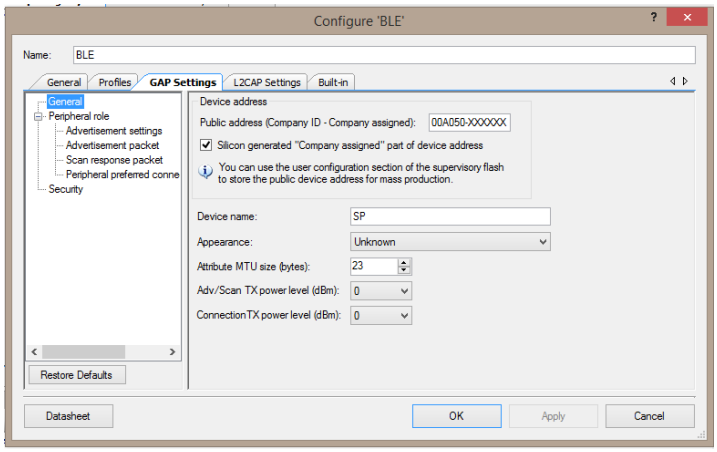
2) Bluetooth Low Energy (BLE) :

We changed the name to BLE. Under General Settings we changed the profile to Custom and profile role to Server (GATT Server) and also defined that GAP role as peripheral.



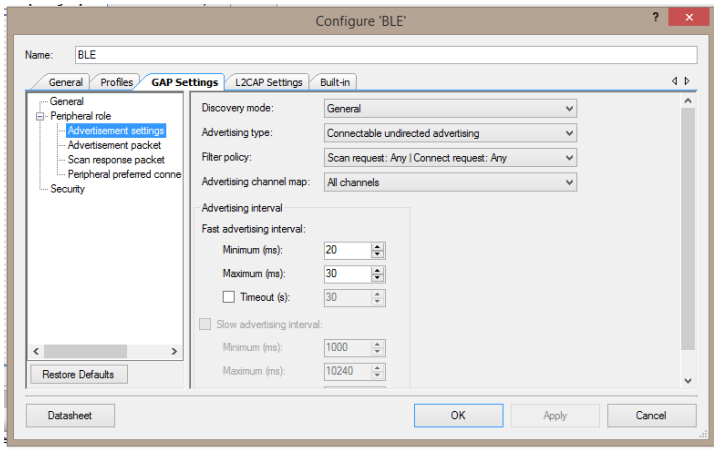
In GAP Settings,

General Settings: We assigned a device name and selected appearance to unknown. We kept the transmission power level to 0 dBm.

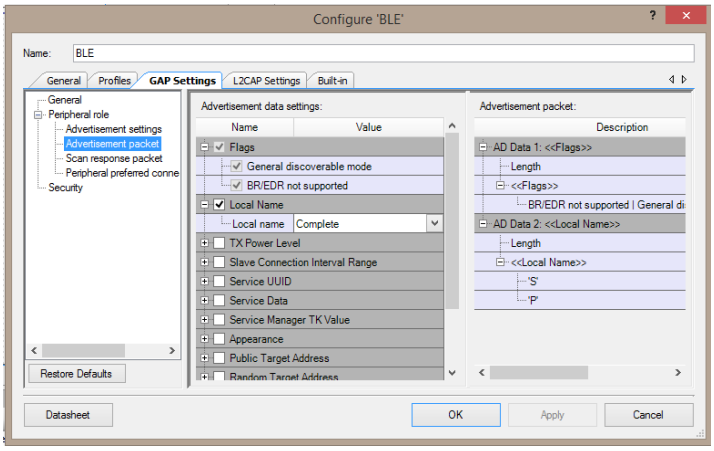


Peripheral Role :

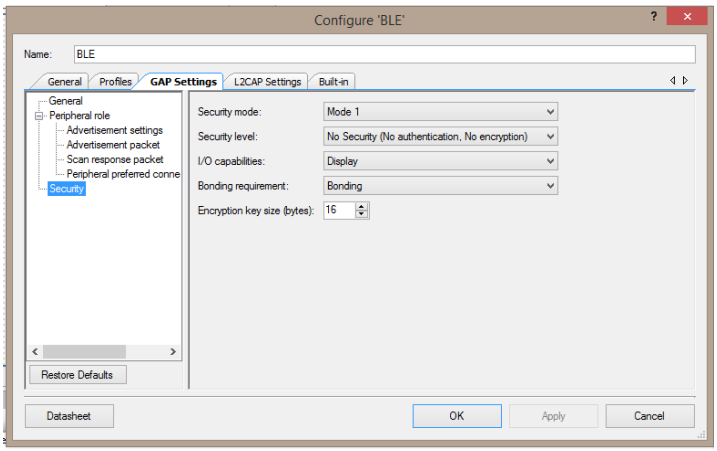
Advertisement Setting: We assigned fast transmission interval ie Minimum 20ms and maximum of 30ms. So that after every 20ms the peripheral will send 1 advertising packet.



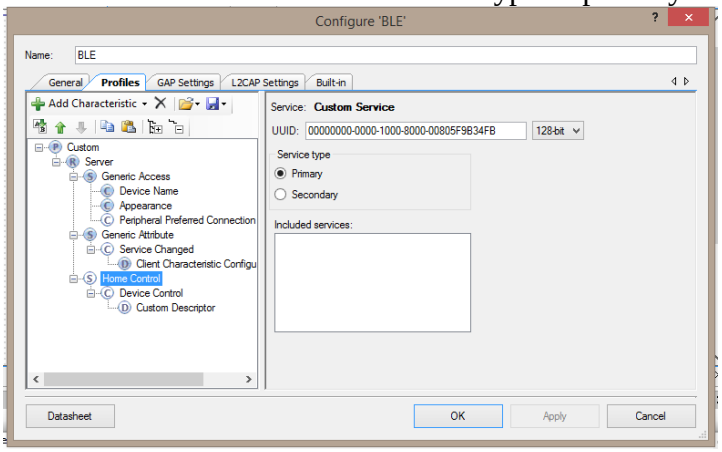
Advertisement packet : We clicked on Local name so that our device name would be visible on CySmart app.



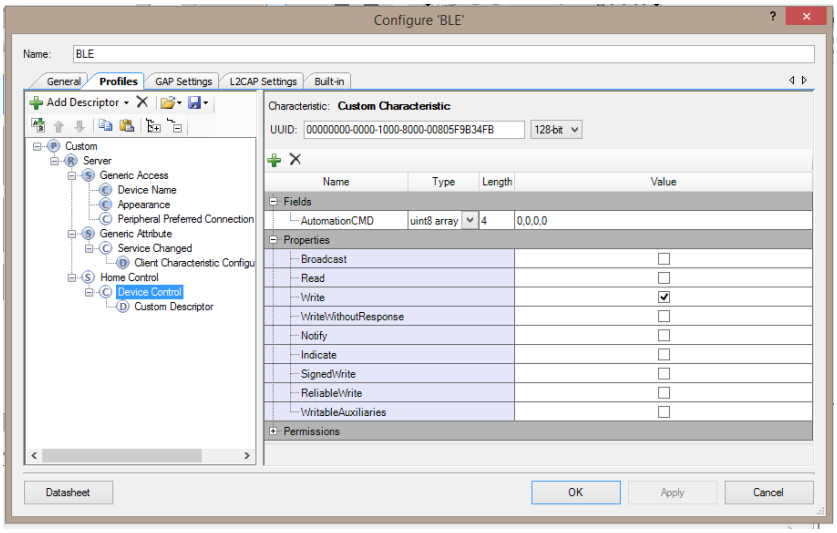
Security : We assigned the security level to No authentication and No Encryption.



In Profiles Settings,
We added a Custom service and selected Service type as primary.

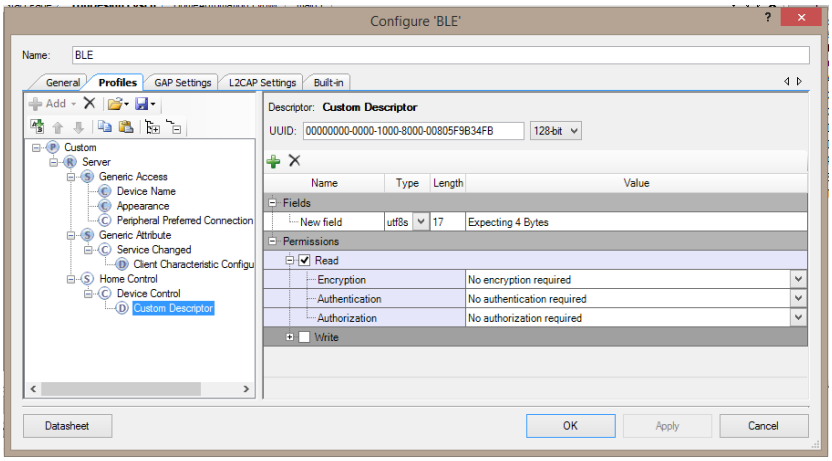


We also added Custom Characteristics and named as Device Control. We assigned fields as uint8 array and a length of 4. Since we are going to write the data on the board so we selected properties as write.



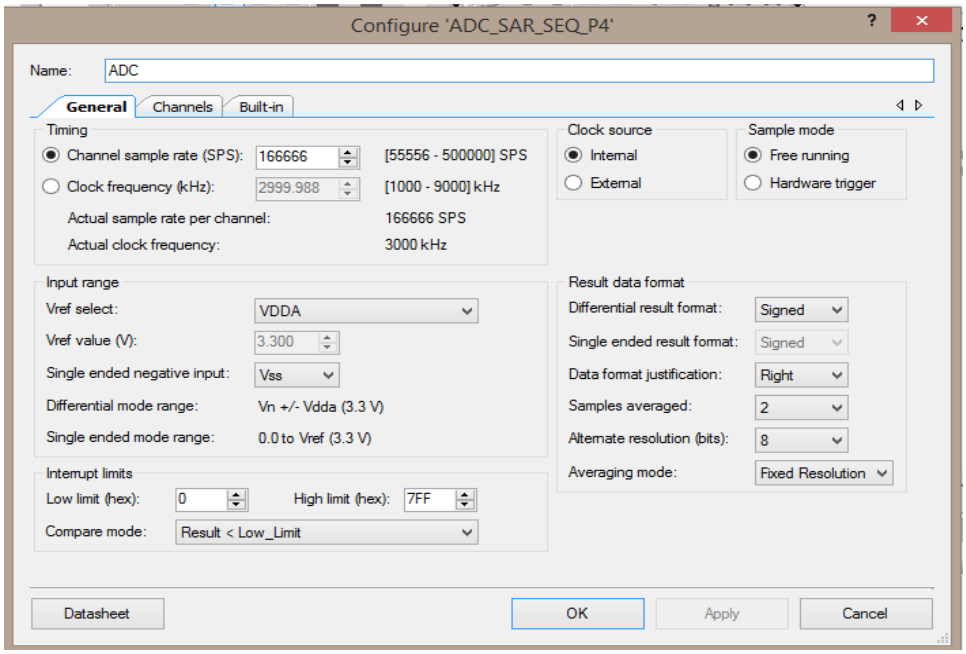
We also added Custom Descriptor. We assigned fields as utf8s and a length of 17. In

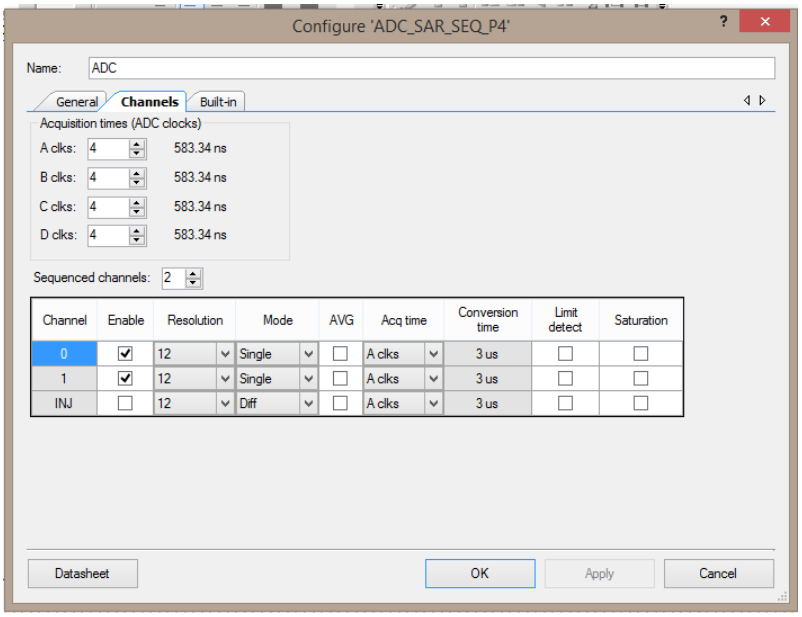
permissions we are going to read the data from user.

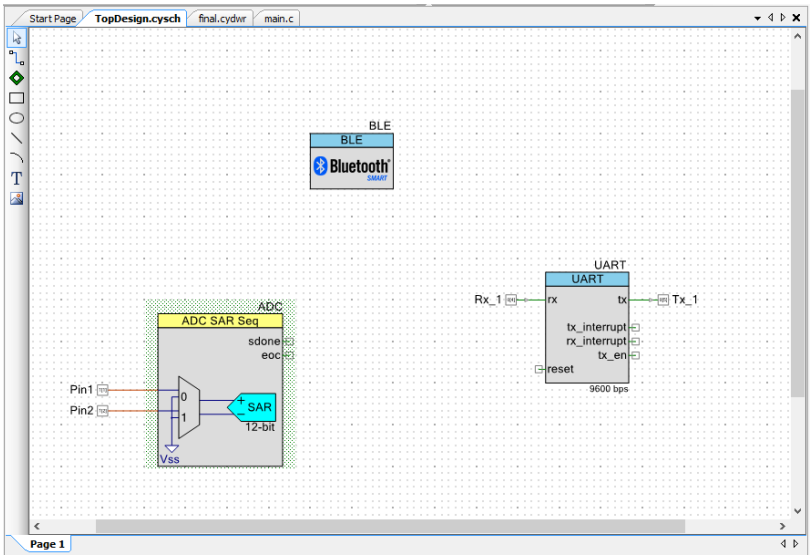


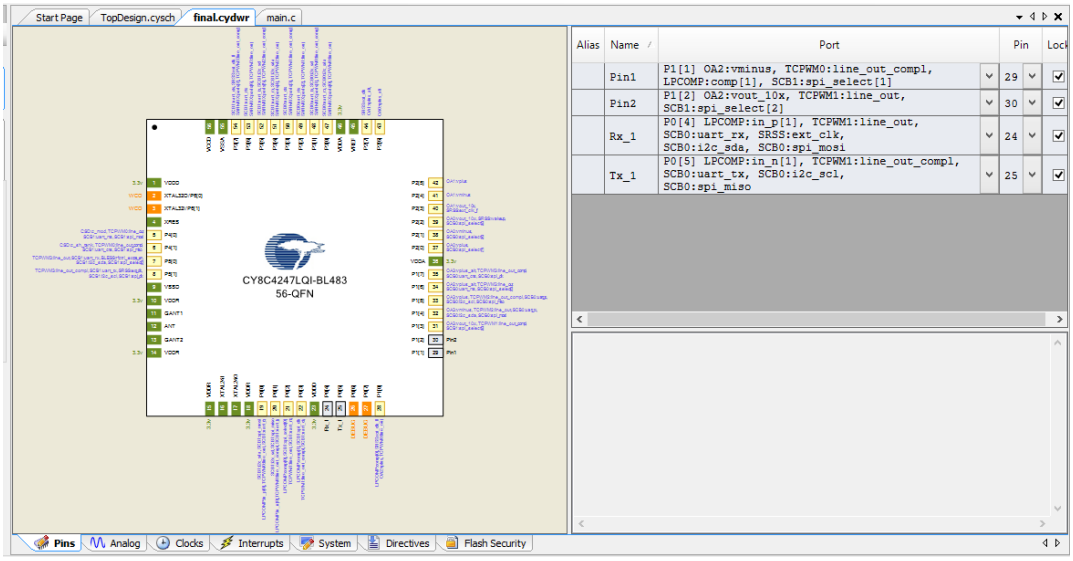
3) Sequencing SAR ADC:

We changed the name to ADC. We selected Vref to VDDA. We have selected free running mode and also reduced the sequenced channel to 2 since 2 channel is required for LDR and Temperature.




Configure 'ADC_SAR_SEQ_P4'
Name: ADC
General
Channels
Built-in
Acquisition times (ADC clocks)
A clks: 4 583.34 ns
B clks: 4 583.34 ns
C clks: 4 583.34 ns
D clks: 4 583.34 ns
Sequenced channels: 2
Channel | Enable | Resolution | Mode | AVG | Acq time | Conversion time | Limit detect | Saturation
0 | 12 | Single | A clks | 3 us
1 | 12 | Single | A clks | 3 us
INJ | 12 | Diff | A clks | 3 us
Datasheet
OK
Apply
Cancel


Start Page
TopDesign.cysch
final.cydwr
main.c
BLE
BLE
Bluetooth
ADC
ADC SAR Seq
sdone
eoc
Pin1
Pin2
0
1
SAR
12-bit
Vss
UART
UART
Rx_1
rx
tx
Tx_1
tx_interrupt
rx_interrupt
tx_en
reset
9600 bps
Page 1



API Description:

API stands for Application Programming Interface. It is a higher level code to complete some operation by a single function. The following are the API's used in this project.

1) ADC_Start();
This instruction is used to perform all required initialization and it enables power.

2) ADC_StartConvert();
This instruction starts converting analog data to digital data continuously.

3) ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
This instruction does not return the result until the ADC of all sequence channel are completed.

4) UART_Start();
This instruction is used to initialize the all UART operations.

5) s = ADC_GetResult16(0);
This instruction gets the result from channel 0 and places it on s. It is a 16 bit data.

6) t = ADC_GetResult16(1);
This instruction gets the result from channel 1 and places it on t. It is a 16 bit data.

7) CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
This instruction starts advertising once the device is disconnected.

8) CYBLE_EVT_GATT_CONNECT_IND:
This instruction indicates when a device is connected.

9) UART_PutChar('X');
This instruction sends the character X on serially on UART.

10) UART_PutChar('Y');
This instruction sends the character Y on serially on UART.

11) UART_PutChar('Z');
This instruction sends the character Z on serially on UART.

Output Observed:

We have control the devices by simply controlling through the CySmart application. In that we gave the command to only send the data of LDR to IoT (0x01 0x01) and for sending the reading of temperature to IoT we gave (0x01 0x02) command and for controlling the accelerometer we gave (0x02 0x01) for X axis, (0x02 0x02) for Y axis and (0x02 0x03) for Z axis.

APPENDIX A

Code in PSoC Creator:

```
#include <project.h>
#include <stdio.h>

CYBLE_CONN_HANDLE_T connectionHandle;
CYBLE_GATTS_WRITE_REQ_PARAM_T *wrReqParam;
uint8 DevType,DevCode,DevParam,Reserve,s,t,y,z;
char buffer[20];
char buffer1[20];

void HandleDevice(){
if (DevType==0x01)
   {
      switch(DevCode)
      {
         case 0x01:
         ADC_StartConvert();
         ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
         s = ADC_GetResult16(0);

         sprintf(buffer,"LDR = %d\n\r",s);
         UART_PutString(buffer);

         /*y = ADC_CountsTo_Volts(0,s);
         sprintf(buffer,"LDR = %d\n\r",y);
         UART_PutString(buffer);
         CyDelay(1000);*/

         break;

         case 0x02:
         ADC_StartConvert();
```

```
            ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
            t=ADC_GetResult16(1);

            sprintf(buffer1,"Temperature = %d\n\r",t);
            UART_PutString(buffer1);

            /*z = ADC_CountsTo_mVolts(1,t);
            sprintf(buffer,"Temperature = %d\n\r",z);
            UART_PutString(buffer1);*/

            CyDelay(1000);
            break;

            case 0x03:
            UART_PutChar('X');
            break;

            case 0x04:
            UART_PutChar('Y');
            break;

            case 0x05:
            UART_PutChar('Z');

            break;

            default:

            break;
    }
    }
}




void CustomEventHandler(uint32 event, void * eventParam)
{
   switch(event)
   {
      case CYBLE_EVT_STACK_ON:   //event called when BLE stack starts

     CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);  //Start advertising
      break;
```

```
        case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
        break;

        case CYBLE_EVT_GATT_CONNECT_IND:
        connectionHandle = *(CYBLE_CONN_HANDLE_T *)eventParam;
        break;

        case CYBLE_EVT_GATTS_WRITE_REQ:    //writing request
        wrReqParam=(CYBLE_GATTS_WRITE_REQ_PARAM_T *) eventParam;
        if(CYBLE_HOME_CONTROL_DEVICE_CONTROL_CHAR_HANDLE==
wrReqParam->handleValPair.attrHandle)
        {
           DevType = (uint8)wrReqParam->handleValPair.value.val[0];
           DevCode = (uint8)wrReqParam->handleValPair.value.val[1];
           DevParam = (uint8)wrReqParam->handleValPair.value.val[2];
           Reserve = (uint8)wrReqParam->handleValPair.value.val[3];
           HandleDevice();
           CyBle_GattsWriteRsp(connectionHandle);
        }

        break;

        default: break;



    }

}

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    ADC_Start();
    UART_Start();
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    CyBle_Start(CustomEventHandler);

    for(;;)
    {
        CyBle_ProcessEvents();
        /* Place your application code here. */

    }
```

```
}

/* [] END OF FILE */
```

Code in mbed :

```
#include "mbed.h"
#include "stdio.h"
#include "MMA8451Q.h"
#define MMA8451_I2C_ADDRESS (0x1d<<1)

MMA8451Q acc(PTE25, PTE24, MMA8451_I2C_ADDRESS);


Serial pc(PTE22,PTE23);
Serial pc1(USBTX,USBRX);//USBTX,USBRX
int main()
{
   int16_t x1=0,z1=0;
   int16_t y1=0;
   int s=10;

   char choice,lux,temp;
   pc1.printf("enter \n\r 1-ldr \n\r 2-lm35 \n\r 3-x \n\r 4-y \n\r 5-z:\n\r");

   while(true)
   {
      if(pc.readable())
      {
         choice=pc.getc();
         switch(choice)
         {
          case 'X':
            x1=-s*(acc.getAccY());
            pc1.printf("x=%d\n\r",x1);
            break;
          case 'Y':
          y1=-s*(acc.getAccX());
            pc1.printf("y=%d\n\r",y1);
          break;
          case 'Z':
          z1=s*(acc.getAccZ());
            pc1.printf("z=%d\n\r",z1);
          break;
          default:
            pc1.putc(choice);
            pc1.printf("\n\r");
```

```
                break;
            }
        }
    }
}
```