PSoC 4 BLE Lab 5: General Security Design of a Blood Glucose Monitor

Group: Gloria Bauman and Kervins Nicolas

Due: December 12th, 2016

Link to Github: https://github.com/kervins/Psoc-Glucose-Meter-

Introduction:

Creating a Glucose Meter to work with a BLE and UART connection. User must be able to use BLE Pioneer kit to create, build, and debug program. Using the Psoc software provided, individual must be able to configure the design figures and code to work in the Glucose Meter lab without errors. Applying general security to the device, Glucose Meter becomes a specified version of student's knowledge. User will understand the aspects of the Psoc software by applying their own modifications to configure the project to their personal objectives.

Description

The purpose of this lab is to understand the embedded architecture of a blood glucose monitor and simulate its function on the PSoC 4 BLE device. You will use the PSoC program and BLE device to design a blood glucose monitor, that simulates blood glucose readings, and connects to a peripheral device.

Objectives

1. Measure simulated blood glucose level reading using an analog to DC signal converter, and implementing the UART on the PSoC device in order to transmit information sent over BLE.

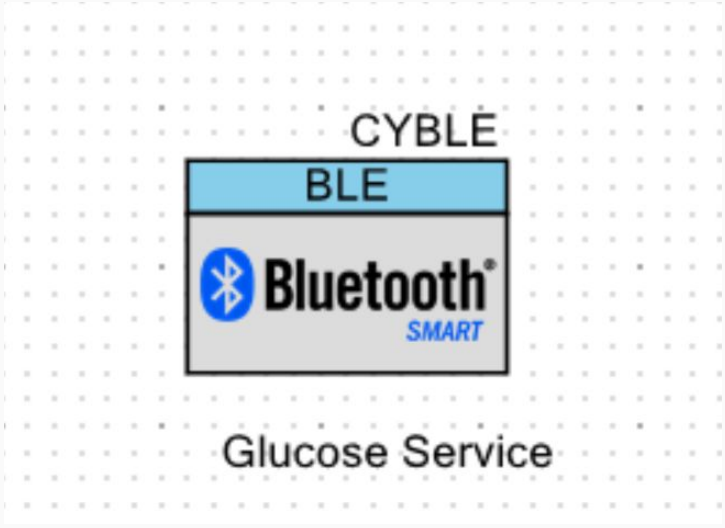
2. Implement a Blood Glucose Profile and send the data over BLE to a peripheral device, such as an IPhone using the cysmart app.

3. Design the system for low power consumption using Sleep, Deep-Sleep and Hibernate modes, which are indicated by various LED lights.

4. Implement a passkey authentication option for the data being sent over BLE in order to simulate protecting medical data that could be very sensitive for diabetics.

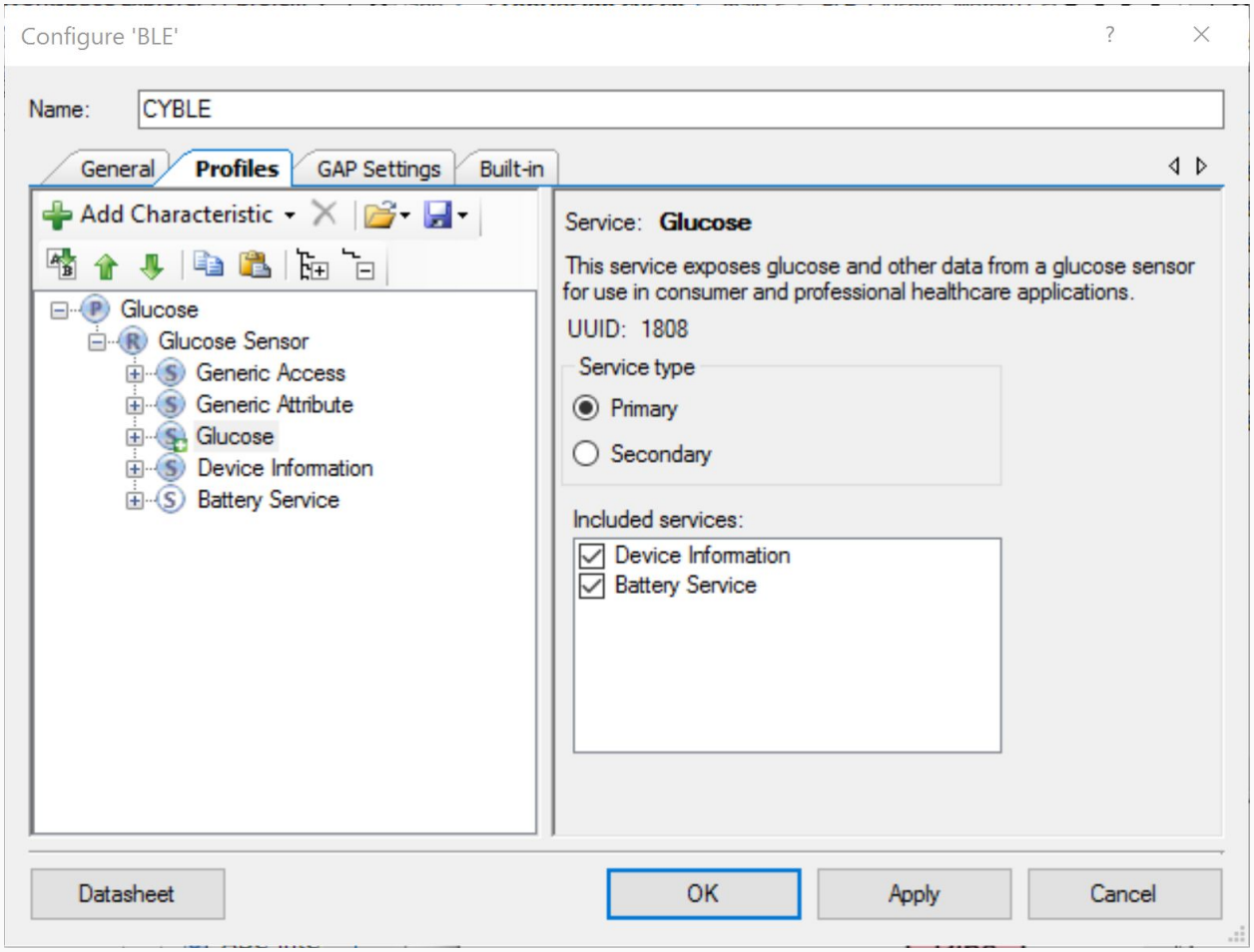
Process: Configure BLE Component

1) Initiate the PSoc Creator program and open the BLE Blood Glucose Example project
2) Under the “TopDesign.cysch” from the “Workspace Explorer” drag and place a “Bluetooth Low Energy” Component from the “Component Catalog”.

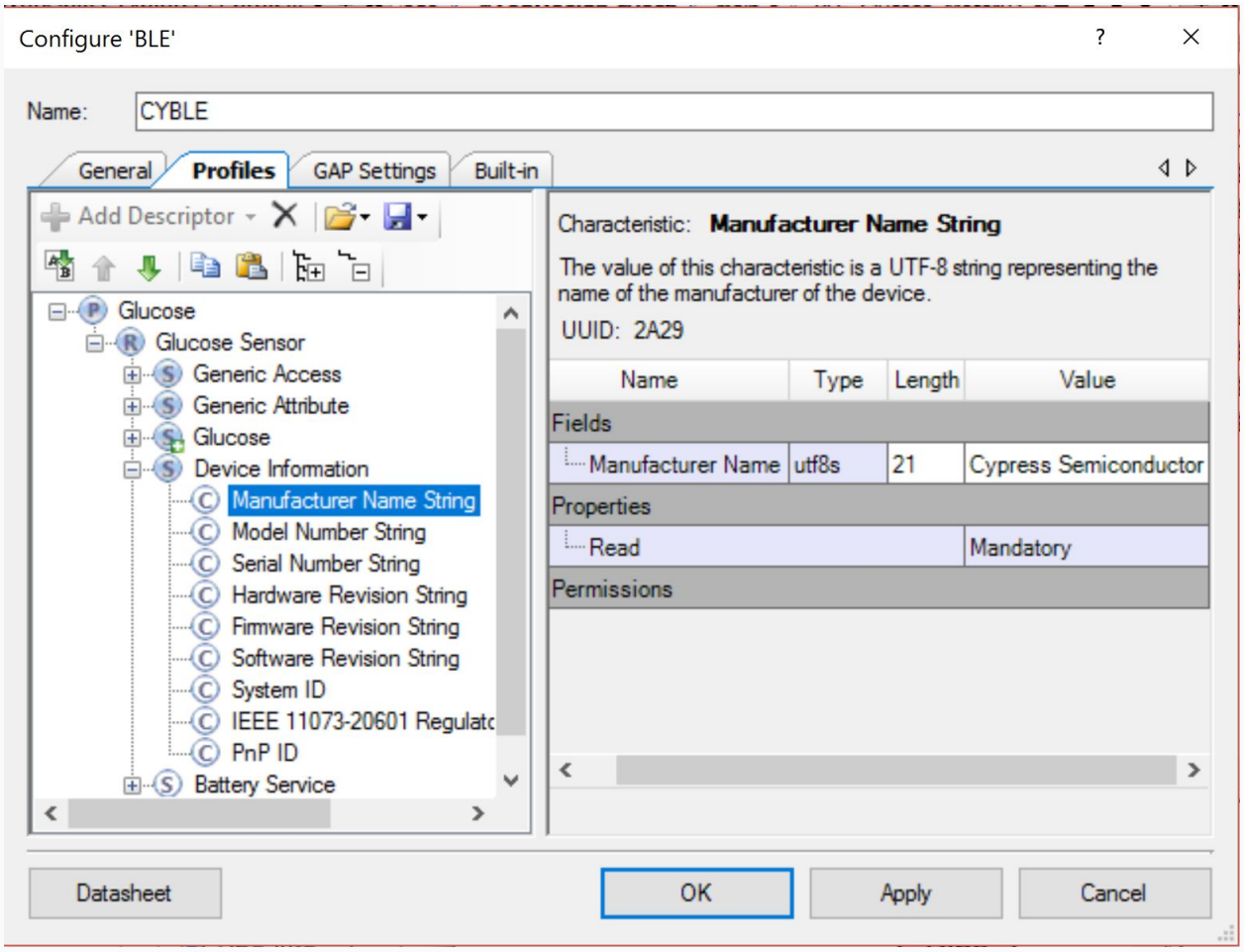


3) Configure the component to the appropriate parameters and settings.
    a) Under the “General Tab”
        i) Profile: “Glucose”
        ii) Profile Role: “Glucose Sensor Server (GATT Server)”
        iii) Check “Use Deep Sleep”

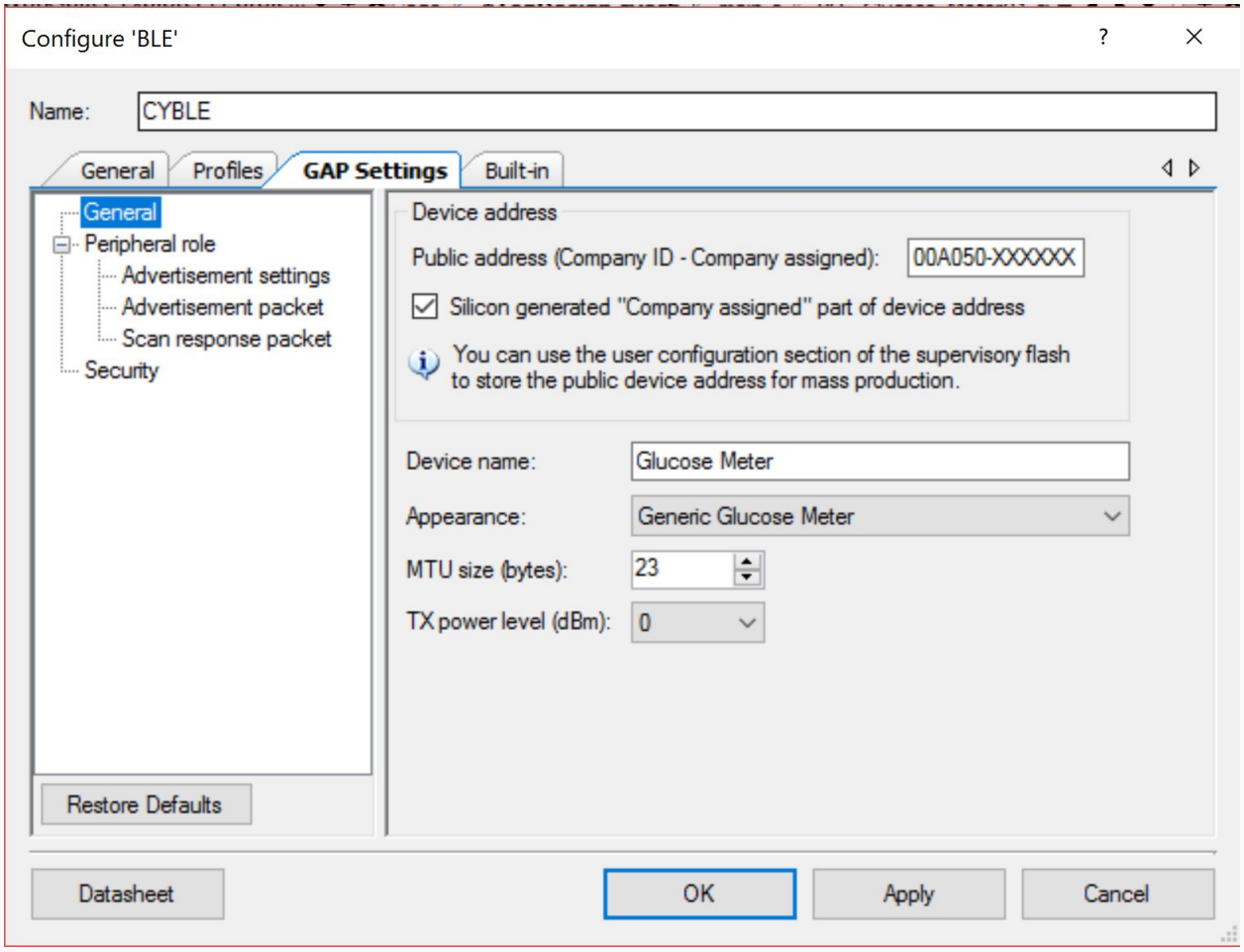
b) Under the “Profiles Tab”
    i) “Glucose” Section
        (1) Service Type: “Primary”
            (a) Included Services:
                (i) Check “Glucose”
                (ii) Uncheck “Battery Service"



    ii) “Device Information” Section
        (1) “Manufacturer Name String” Sub-Section
            (a) Type in “Cypress Semiconductor” in the “Manufacture Name” value



c) Under the “GAP Settings” tab
    i) “General” Section
        (1) Check off “Silicon generated ‘Company assigned’ part of the address (00A050-XXXXXX)
        (2) Device Name: “Glucose Meter”
        (3) Appearance: “Generic Glucose Meter”
        (4) MTU size (bytes): “23”
        (5) TX power Level (dBM): “0”



ii) "Peripheral Role" Section

(1) "Advertisement Settings" Sub-section

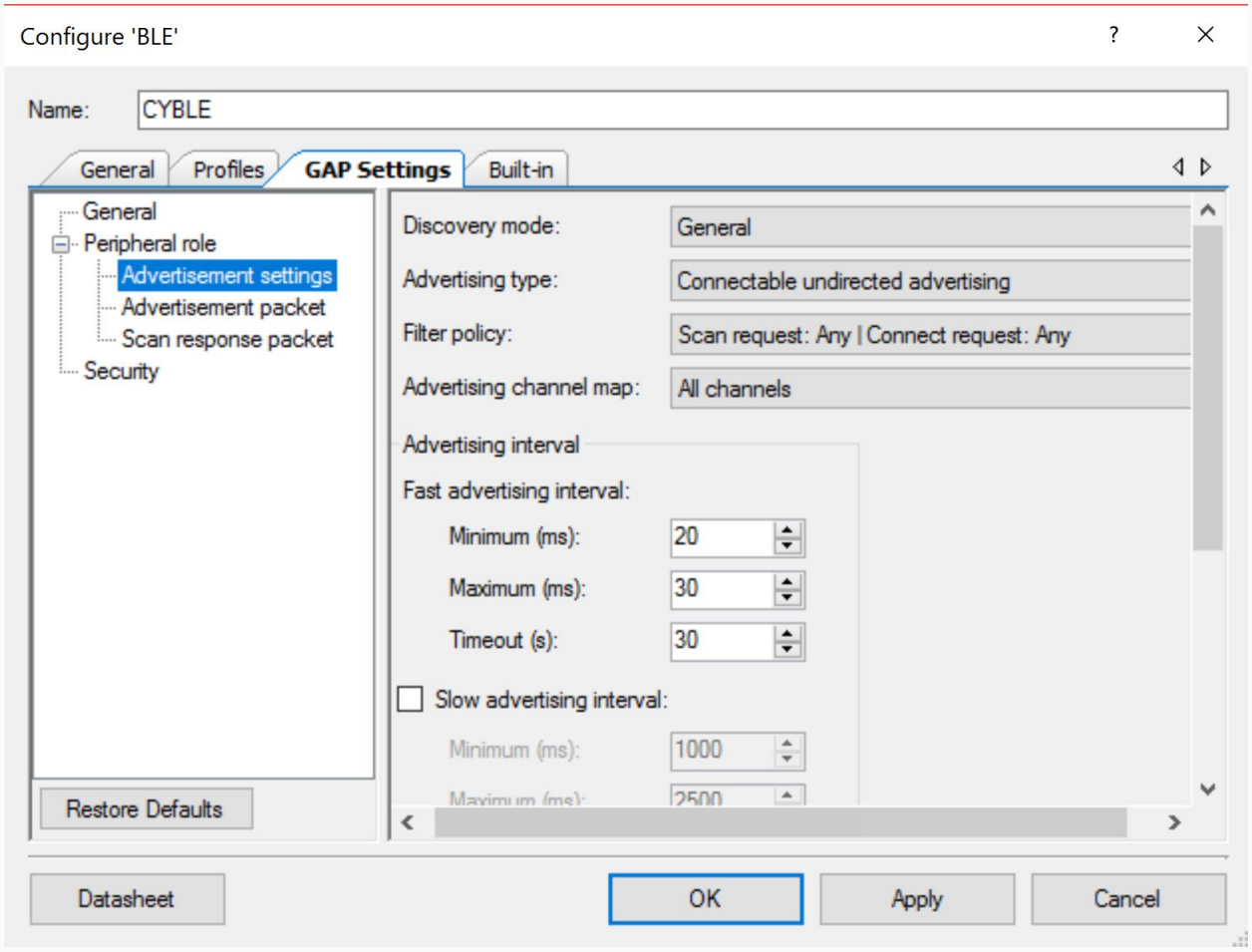
(a) Discovery Mode: "General"

(b) Advertising Type: "Connectable undirected advertising"

(c) Filter policy: "Scan request Any | Connect request Any"

(d) Advertising channel map: "Any"

(e) Uncheck "Slow Advertising Interval"



(2) “Advertisement Packet” sub-section

(a) Check “Local name”

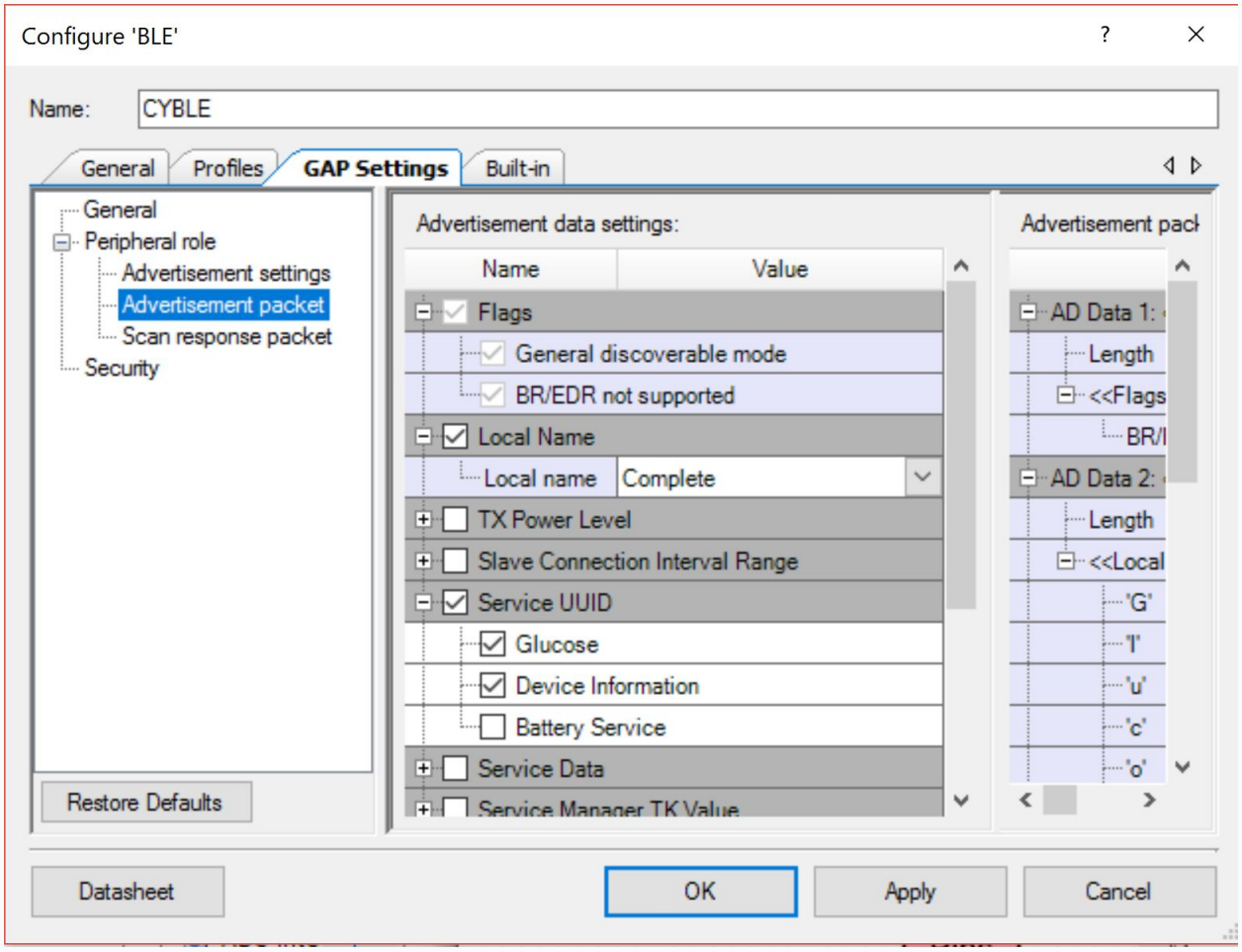
(i) Local Name: Complete

(b) Check “Service UUID”

(i) Check “Glucose”

(ii) Check “Device Information”

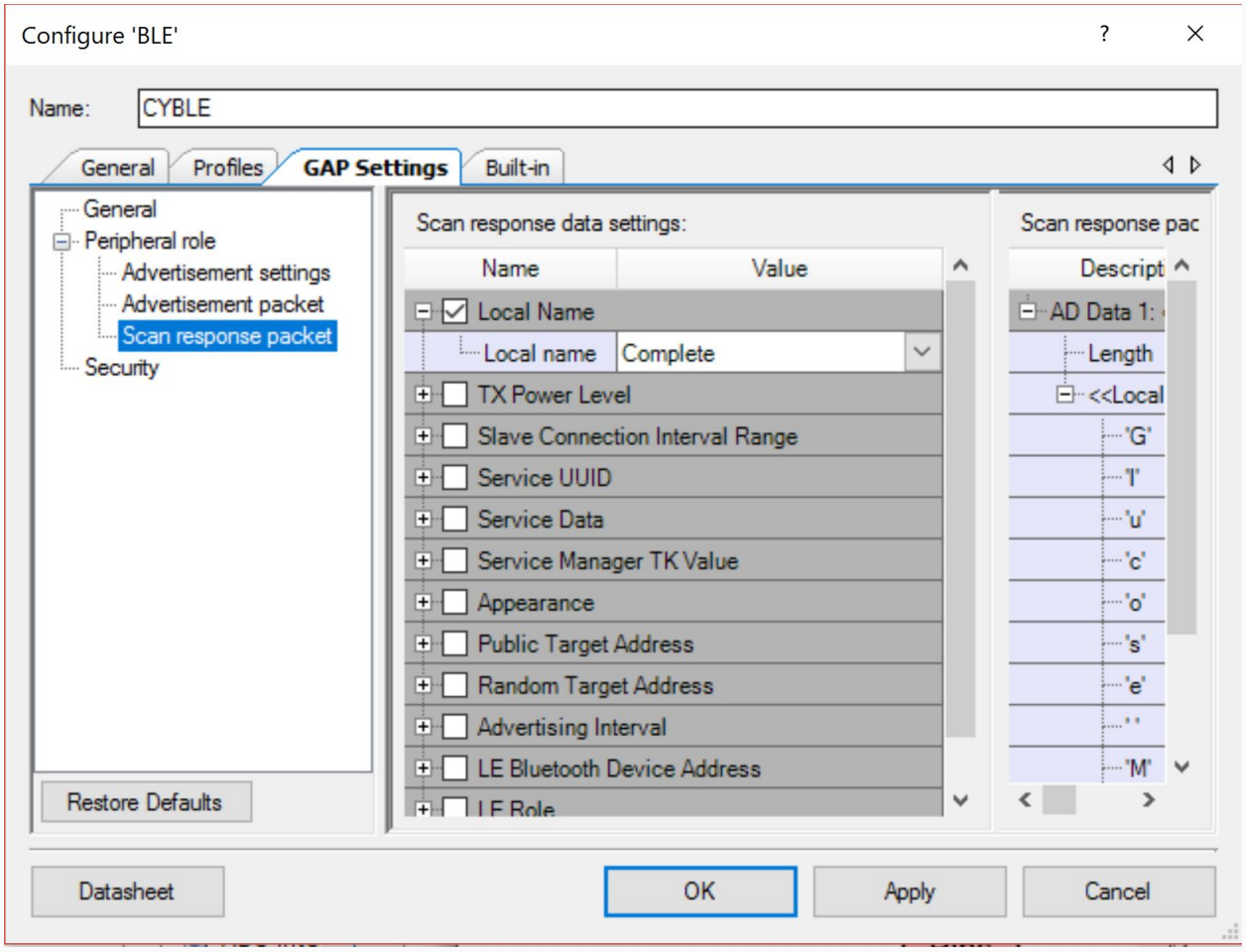
(iii) Uncheck “Battery Service”



(3) “Scan Response Packet” sub-section

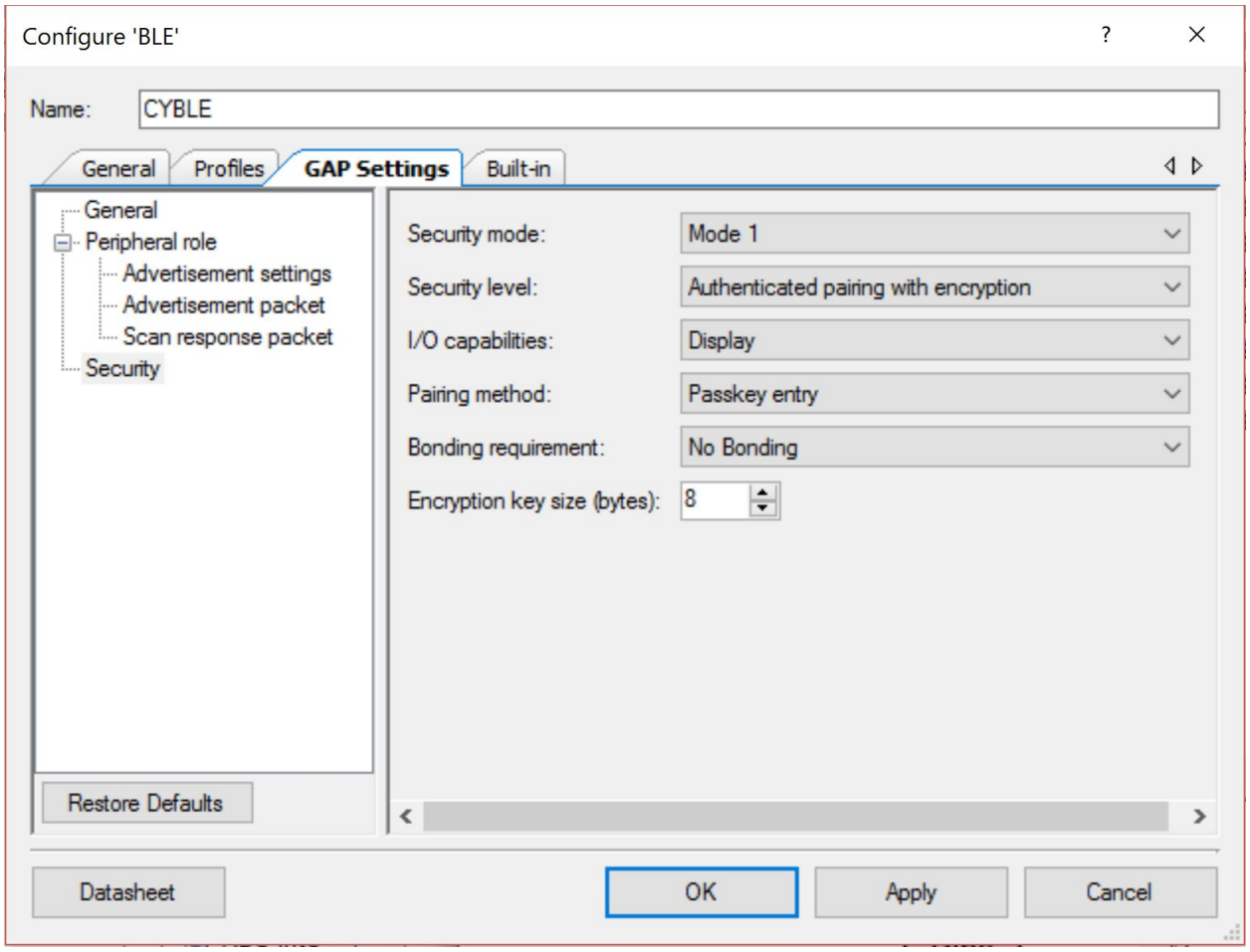
(a) Check “Local Name”

(i) Local Name: Complete



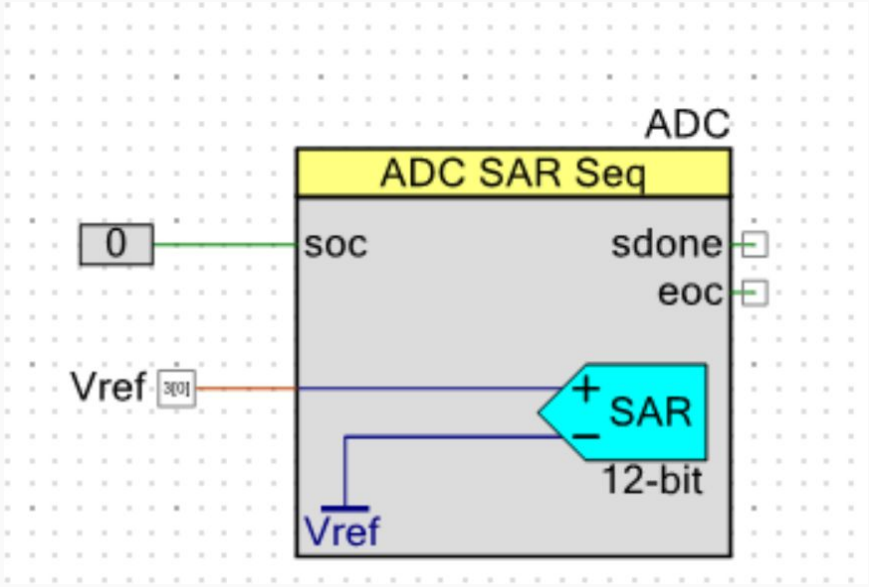
iii) "Security" Section

(1) Security Mode: "Mode 1"
(2) Security Level: "Authenticated paring with encryption"
(3) I/O Capabilities: "Display"
(4) Paring Method: "Passkey entry"
(5) Bonding Requirement: "No Bonding"
(6) Encryption Key Size (bytes): "8"

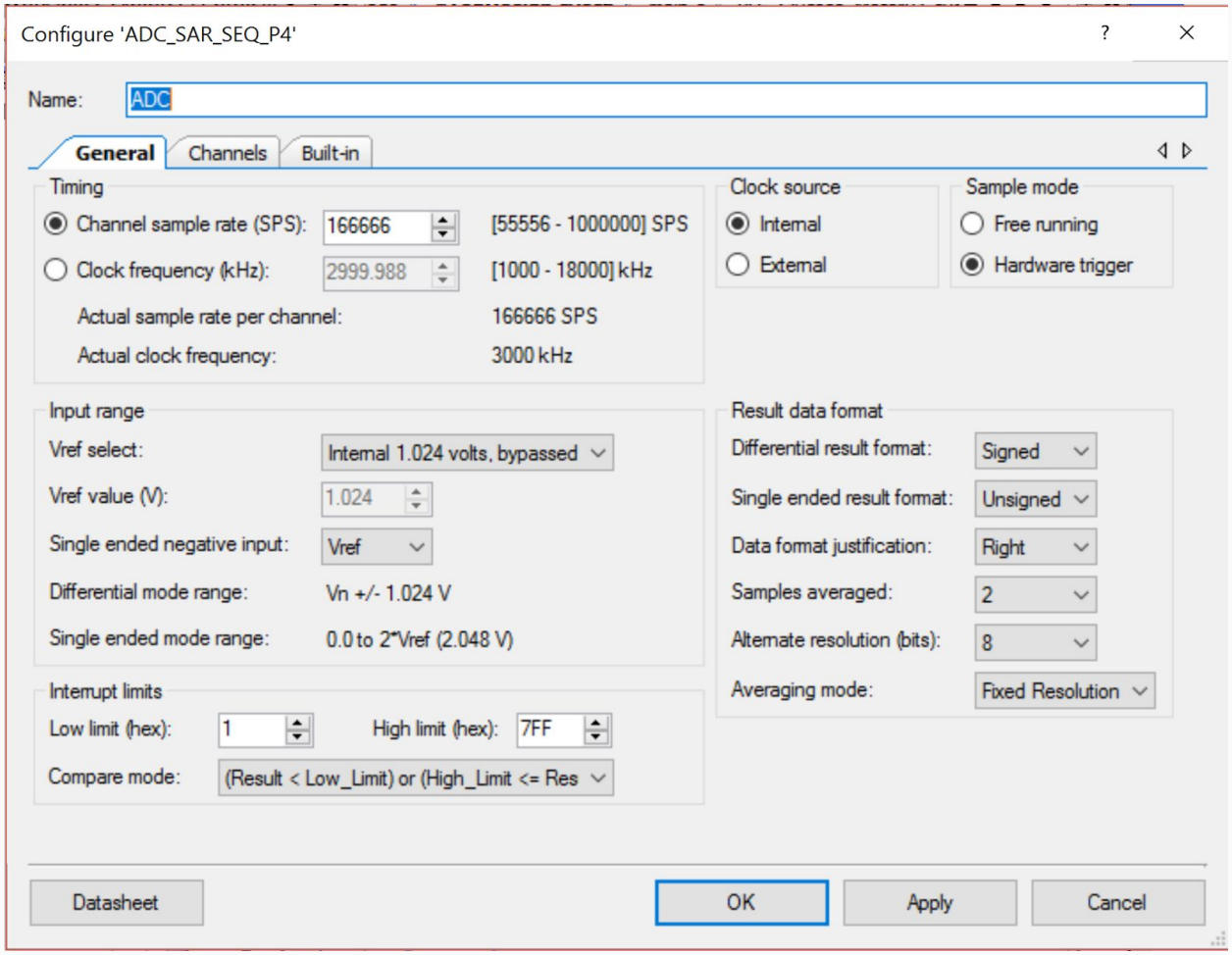

Configure 'BLE'
Name:
CYBLE
General
Profiles
GAP Settings
Built-in
General
Peripheral role
Advertisement settings
Advertisement packet
Scan response packet
Security
Restore Defaults
Security mode:
Mode 1
Security level:
Authenticated pairing with encryption
I/O capabilities:
Display
Pairing method:
Passkey entry
Bonding requirement:
No Bonding
Encryption key size (bytes):
8
Datasheet
OK
Apply
Cancel

Process: Configure ADC SAR Seq Component

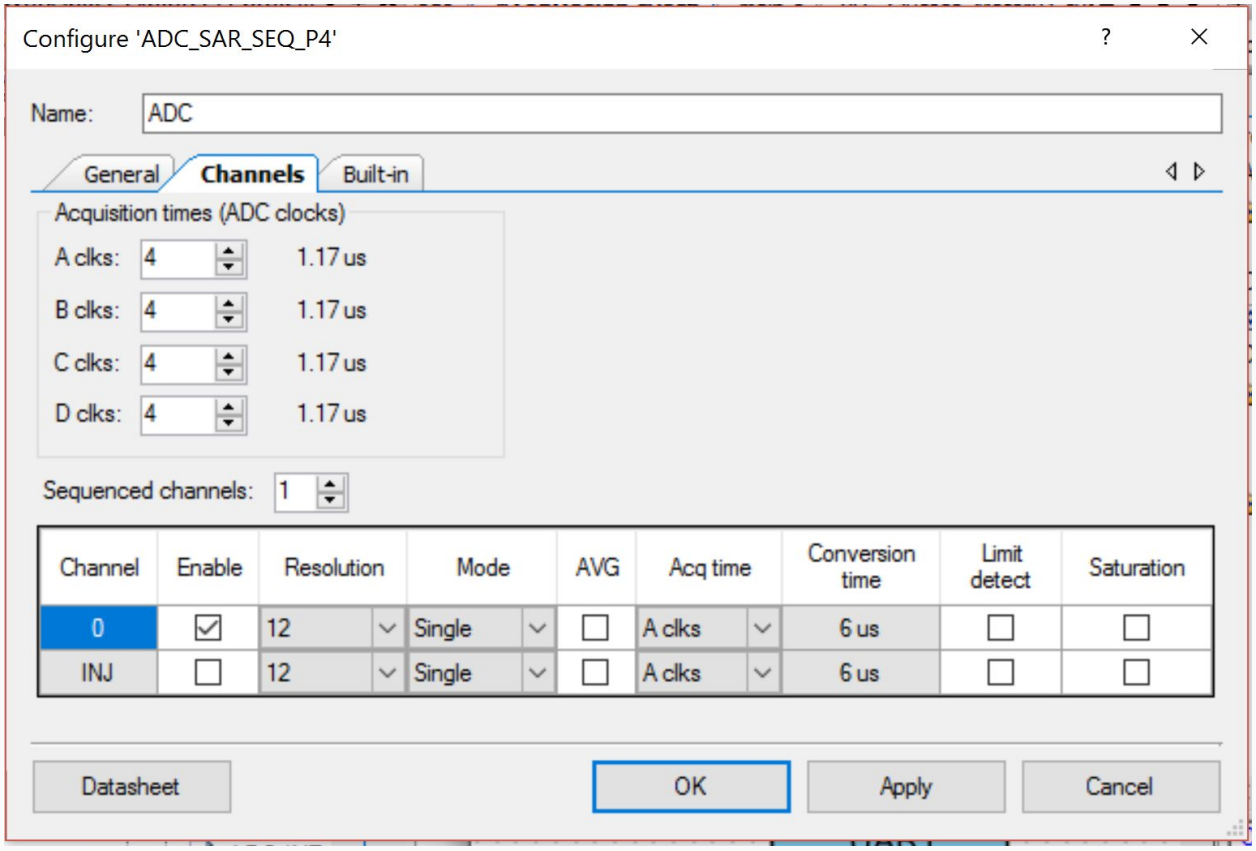
1) Connect the “soc” input to a “Low Logic ‘0’ [v1.0]” component from the “Component Catalog”



2) Configure the component to the appropriate parameters and settings
    a) “General Tab”
        i) Clock Source: Internal
        ii) Sample Mode: Hardware Triggered
        iii) Result Data Format
            (1) Differential Result Format: Signed
            (2) Data Format Justification: Right
            (3) Samples Averaged: 2
            (4) Alternate resolution (bits): 8
            (5) Averaging Mode: Fixed Resolution
        iv) Interrupt limits
            (1) Low Limit (hex): 1
            (2) High limit (hex): 7FF
            (3) Compare Mode: (Result<Low_Limit) or (High_Limit <= Result)
        v) Input Range
            (1) Vref Select: Internal 1.024 volts, bypassed
            (2) Single ended negative input: Vref
        vi) Timing
            (1) Channel sample rate (SPS): 166666



b) “Channels Tab”
   i) Sequenced Channels: 1
   ii) Channel 0
      (1) Check: enable
      (2) Resolution: 12
      (3) Mode: single
      (4) Uncheck: AVG
      (5) Acq Time: A clks
      (6) Conversion Time: 6us


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
Single
A clks
6 us
Datasheet
OK
Apply
Cancel

Build and Program

1) Bootloading PSoC 5: this will simulate a blood glucose level reading
    a) While holding down “SW1 (reset)” plug in the kit’s usb connector to the PC,
    b) allowing the kit to enter the bootloader mode.

Bootloader

- MCU (Multipoint Control Unit) - Hardware used as a bridge terminal involved in the connection of multiple systems
- Firmware is implemented in a MCU’s cache/flash memory
- Firmware must be able to communicate in the operation and flash
- Bootloader allows updates possible

```
}

/* Store bonding data to flash only when all debug information has been sent *
/* if((cyBle_pendingFlashWrite != 0u) &&
    ((UART_DEB_SpiUartGetTxBufferSize() + UART_DEB_GET_TX_FIFO_SR_VALID) == 0u)
{
    apiResult = CyBle_StoreBondingData(0u);
    printf("Store bonding data, status: %x \r\n", apiResult);
}

/*******************************************************************
*   Process GLS RACP requests
*******************************************************************/
GlsProcess();


/*******************************************************************
*  Process all pending BLE events in the stack
*******************************************************************/
CyBle_ProcessEvents();
```

Fig 1. Displays Code for the battery

- Battery is must be commented out by implementing “//” to code
- Project not connected to User’s Glucose Meter
- If not commented out, project will not build
    - Contains Errors

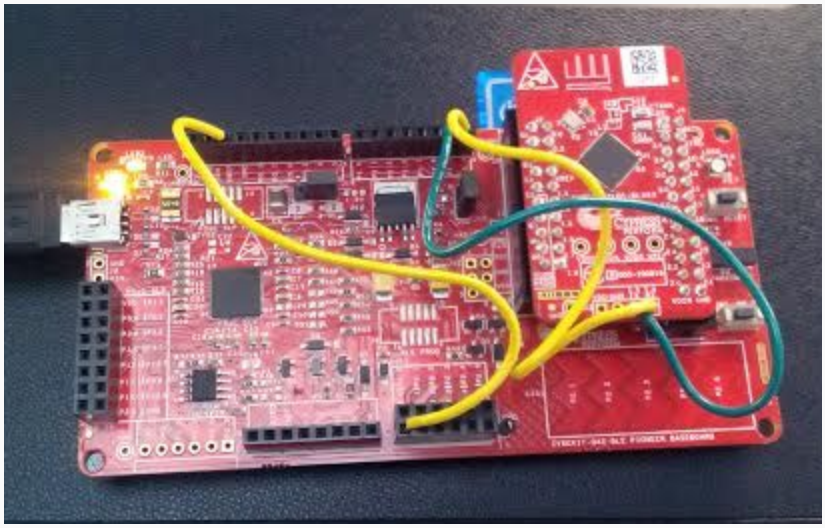


Fig 2. Displays wire connections

=

- Connection

UART RX pin = port 1 pin 4
UART TX pin = port 1 pin 5
VREF pin(J3) = port 3 pin 0 to battery voltage measurement
Port 1 pin 7 = Mechanical button
Used to wake up and start re-advertising of device
Port 2 pin 6
Indicates BLE Disconnection
Port 3 pin 6
Indicates advertising state
Port 3 pin 7
Indicates Battery Discharge(low power)

Testing with CySmart BLE Test and Debug Tool

1) Press “SW1” to start advertising

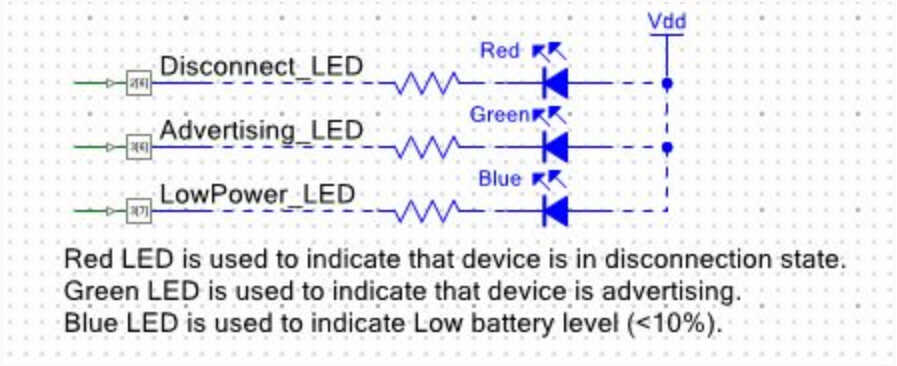


Fig 3. Displays LED States

2) Open “CySmart Mobile App” on your mobile Bluetooth capable device
3) Find your device listed under the home screen as “Glucose Meter” and tap on it.

4) A “Bluetooth Pairing Request” prompt should appear on device

```
129  int main()
130  {
131      CYBLE_LP_MODE_T lpMode;
132      CYBLE_BLESS_STATE_T blessState;
133
134      CyGlobalIntEnable;
135      UART_DEB_Start();                  /* Start communication component */
136      CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
137              //central is able to disconnect after disconnection
138
139      printf("\r\nBLE Glucose Meter Example Project\r\n");
140
```

Fig 4. Displays Code to enable passkey

Cyble_GapStartAdvertisement(CYBLE_ADVERTISING_FAST) from sub code

```
36  void StartAdvertisement(void)
37  {
38      uint16 i;
39      CYBLE_GAP_BD_ADDR_T localAddr;
40      apiResult = CyBle_GappStartAdvertisement(CYBLE_ADVERT
41      if(apiResult != CYBLE_ERROR_OK)
42      {
43          printf("StartAdvertisement API Error: %x \r\n", (
44      }
45      else
46      {
47          printf("Start Advertisement with addr: ");
48          CyBle_GetDeviceAddress(&localAddr);
49          for(i = CYBLE_GAP_BD_ADDR_SIZE; i > 0u; i--)
50          {
51              printf("%2.2x", localAddr.bdAddr[i-1]);
52          }
```

Fig 5. Displays sub-code for device to start advertising

a) Enter in the predefined passkey “000000” to gain access to the medical records

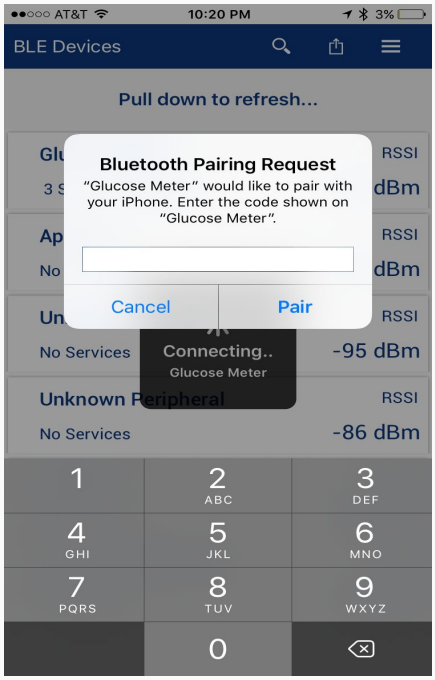


Fig 6. Displays Authentication code that allows Bluetooth Pairing

5) Go to the “Profile” Screen and tap on the “Glucose ” service.
    a) To get glucose reading tap on “Read Last” or “Read All”
        i) Read Last: only shows the readings from day 10 which is the last day
        ii) Read All: Creates a drop-down list of 10 days of various blood glucose level readings

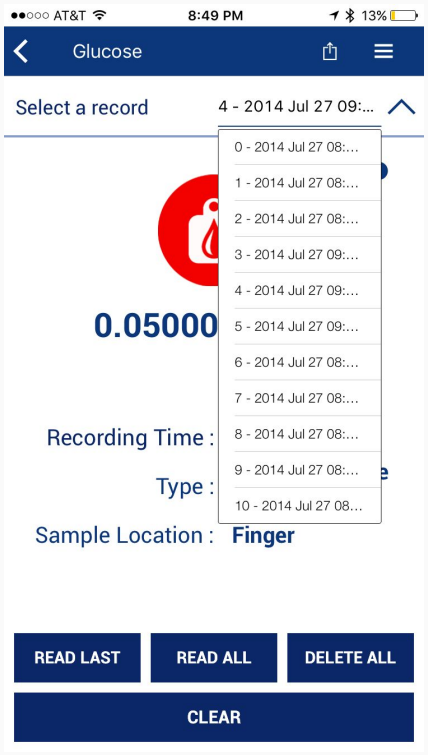


Fig 7. Displays Glucose readings of different days

b) Every day has a unique profile that consists of:
   i) Blood glucose level
   ii) Date/Time
   iii) Type of reading (ex. Blood)
   iv) Sample Location (ex. Finger Prick)

Fig 8. Displays Glucose Meter Level

Fig 9. Displays Glucose Meter Level

c) Each Glucose reading can be analyzed further by tapping on the tiny "*i*" in the top right corner. This sub-profile contains various data such as:

i) Carbohydrate ID, Carbohydrate, Meal, Tester, Health, Exercise Duration, Exercise Intensity, Medication ID, Medication, and Hb1c

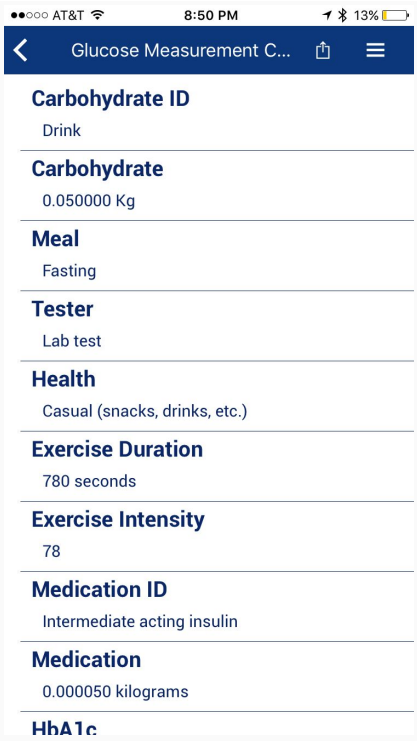


Fig 10. Displays Specified Glucose Options

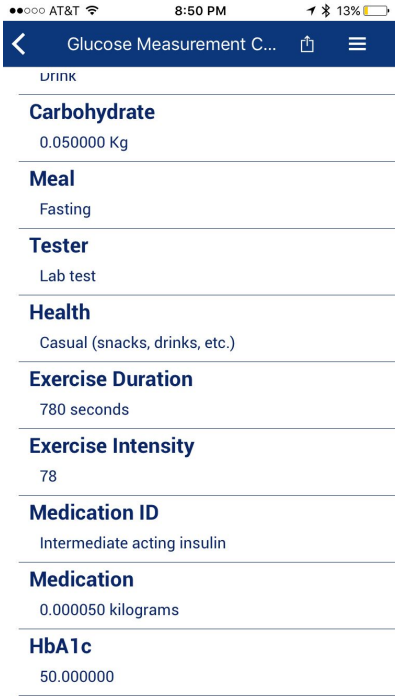


Fig. 11. Displays Specified Glucose Options

Tera Term Application

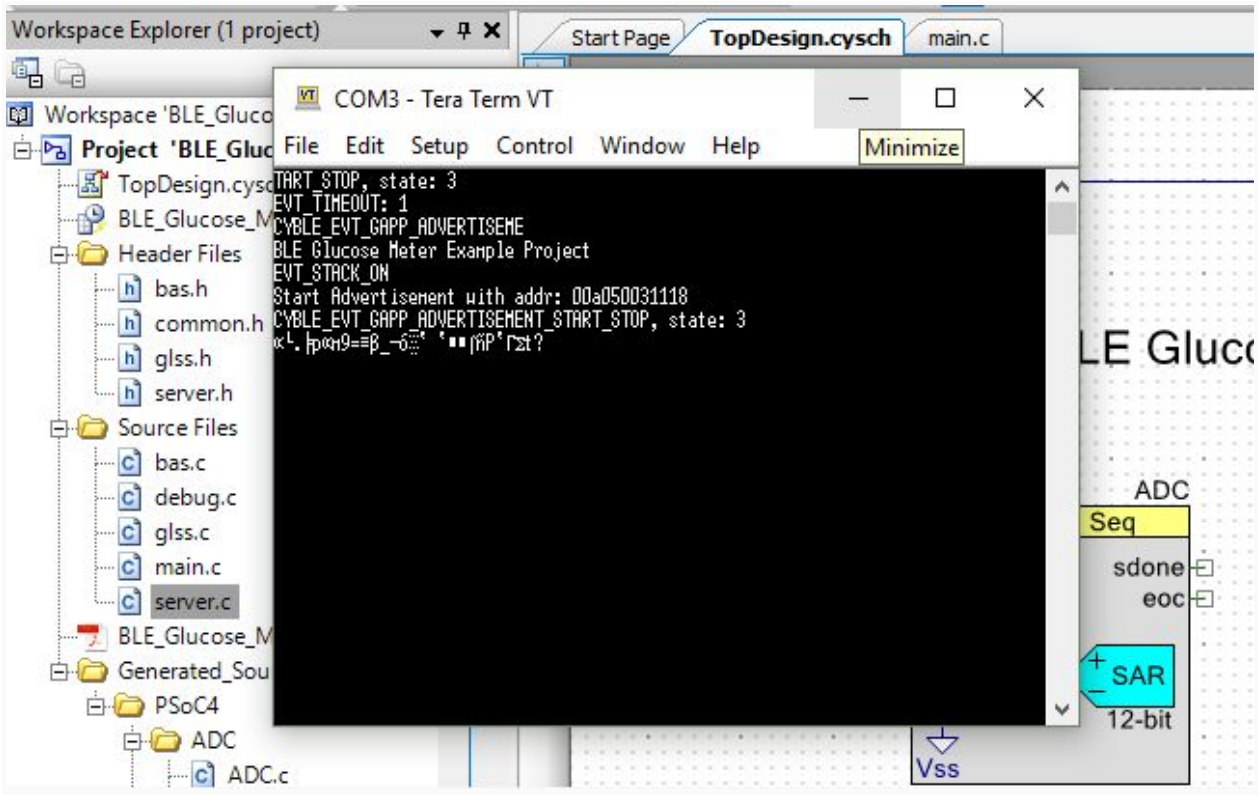


Fig 12. Displays Glucose Meter Program

Communication port in the KitProg

- Defines state of Kit
- Whether it is advertising a signal or not
- Displays address
- Defines part of code that allows advertising to be implemented

Problems

Changing the Passkey

- While loop
    - Once in a while loop, cannot leave
    - Program call program in while loop
        - locates and applies it in the program
        - Goes through all functions in the while loop

```
162     while(1)
163     {
164         /***********************************
165     *Stack Event Handler
166     ***********************************/
167     int8 passkeyInsert = 2;
168     int32 passKey = 0;
169     int8 passDigit[10];
170     int8 i = 0;
171      /***********************************
172     *Global Variables
173     ***********************************/
174     char8 command;  //input for UART
175
176     {
177      if(passkeyInsert!= -1)
178     passkeyInsert = -1;
179
180     CyBle_GapAuthPassKeyReply(cyBle_connHandle.bdHandle,passKey,CYBLE_
181                 passKey = 0;
182
183     {
184         if((command <= '0') && (command >= '9'))
```

Fig 13. Displays User's code to change PassKey

Code

- Is statement
- Must apply call statement
    - CyBle_GapAuthPassKeyReply(..)
- Stack Handler
    - Routine or specialized functions that must be defined
    - Input handler
- Global Variables
    - State variable defined in code

Concept

- Must apply certain code to work in the authorization between the pairing of the Program and BLE Devices

Conclusion

This Lab gave the user's knowledge about important aspects of building and programing a PSOC project. Users were able to apply own methodology on an existing project in order to better understand PSOC. Using coding in the main.c file, code was configured to apply various changes to project such as security authentication passkey. Allowing for the errors and warning, knowledge was gained. Students were given the ability to change the LED displays of the device due to any changes being made to device. Bluetooth connections, UART, and Psoc understanding was obtained while implementing different aspects to the Glucose Meter project.